On Convergence of the Method of Relaxations, an Application to Learning on a Graph.

Sylvain Marié, Mark Herbster

DRAFT – September 14, 2006– DRAFT
On Convergence of the Method of Relaxations, an Application to Learning on a Graph.
DO NOT DISTRIBUTE

Sylvain Marié, Mark Herbster
Department of Computer Science
University College London
Gower Street,
London WC1E 6BT, UK
m.herbster {at} cs.ucl.ac.uk
sylvain.marie {at} ingenieurs-supelec.org

Abstract

We present a semi-supervised algorithm to learn an interpolating function with minimal Laplacian semi-norm on a graph. This algorithm is directly inspired by the classical 'method of relaxations' used in physics to get discrete approximate solutions to the Dirichlet problem. We argue that this method is actually a gradient descent minimizing the energy of an interpolating function on the graph. Besides it converges linearly in the number of steps, with a convergence constant bounded with respect to simple characteristics of the graph. As opposed to inversion-based kernel methods on graph, this algorithm can be efficiently implemented to deal with large, complex graphs.

Keywords: Semi-supervised learning on graphs, method of relaxations, relaxation algorithms.

1 Introduction

We consider learning over a graph. Learning is a process involving *training* an algorithm (the learner) on a number of given (pattern, label) pairs so that the learner can afterwards *predict* the labels of unknown (unlabelled) patterns. Performance can be assessed by comparing the predicted labels with the true labels and counting the number of mistakes, the goal being to minimize this number. These settings however have to to take into account the reality of available data for classification applications. Indeed in many real world datasets only a few labelled samples are available, whereas unlabelled patterns are easier to get: labelling medical data, for example, often requires a long and costly expertise. Intuitively we see that unlabelled data probably contains relevant information about the underlying distribution of input samples, therefore it could be of interest to exploit it together with labelled data samples. This framework is that of *semi-supervised learning*.

In order to exploit the inherent geometry of their underlying distribution, patterns can be represented as vertices of a graph G which may either be given or be constructed from a similarity metric on the objects [BN04, ZGL03]. Our approach stands within the frame of *semi-supervised learning on a graph*, which has been studied from many perspectives; a common point of view in the literature is to represent functions defined on a graph by a Hilbert space associated to the Laplacian of the graph.

In this paper we study graph learning from an optimization point of view rather than a learning performance one. Our *relaxation* algorithm is a connectionist model of a proven semi-supervised technique. Although based on a classical descent algorithm, its update steps are surprisingly very

simple and intuitive. Therefore it is reasonable to think that something similar might already be change implemented in nature; indeed the consensus concept can be seen in group behaviors for example.

The maximum number of steps to perform to reach the solution within a tolerance ϵ is of order $O(n^4)$, whereas for the regularized version of the algorithm, it is $O(n^2)$. The study of convergence, in addition to providing bounds on the maximum error after a certain number of steps, leads to a better understanding of the system.

The network/graph model has several implementation implications. Memory used for the computations at each step is O(d) for the consensus algorithm, whereas it is O($n - l$) for standard gradient descent, and O($(n - l)^2$) for inversion-based methods (d denotes the degree of the updated vertex, l is the number of known vertices, and n is the total number of vertices). Since the simple update rule for the consensus algorithm only requires the values of the vertices of one neighborhood, it can be implemented in a a wholly parallel fashion, for example by assigning a processor to a vertex or a group of vertices and having the parallel processors communicate locally with their neighbors. An efficient implementation for very large graphs can be done on sequential architectures, using a *heap* data structure to keep track of the steepest direction. Finally, if after having run the algorithm the graph is slightly modified and we run the algorithm again, convergence might be very fast: indeed the impact of a slight modification would in most cases be significant in a small area surrounding it and thus we expect our function to be still very close to the optimal point.

We quickly review some concepts from graph theory in Sect. 4 and the Hilbert space of functions on graphs in Sect. 5. In Sect. 6 we develop an analysis of the l_1-norm steepest descent algorithm used to minimize a strongly convex function in the general case. These considerations are used in Sect. 7 to develop and analyze learning algorithms based on sequential simple updates of an interpolating function. We propose simple bounds for the convergence in Sect. 7.5, depending on characteristic values of the graph G. Finally, although these algorithms are better fit to parallel architectures, we discuss an efficient implementation on sequential architectures using a *heap* data structure for very large graphs in Sect. A.7.

2 The Discrete Dirichlet Problem

Let us consider an electric network of resistors of equal resistance. We connect some points of this network (the '*boundary*') to either a positive or a negative voltage source, and would like to guess the voltage at every other point of the network (*interior points*, see Fig. 1). By Kirchoff's and Ohm's laws the solution is given by the following *averaging property*

$$v_i = \left(\sum_{(i,j)\, connected} R_{ij}^{-1} \right)^{-1} \sum_{(i,j)\, connected} \frac{v_j}{R_{ij}} = \frac{\sum_{(i,j)\, connected} v_j}{\sum_{(i,j)\, connected}} \ .$$

Such a voltage distribution is called *harmonic*. The *Uniqueness Principle* for this Dirichlet problem asserts that there can not be two different harmonic voltage distributions corresponding to one setting of *boundary values*. Other interesting properties are described in details in [DS00]. In particular this harmonic function minimizes *energy dissipation* through the resistors,

$$E = \sum_{(i,j)\, connected} \frac{1}{R_{ij}} \left(v_i - v_j \right)^2 \ .$$

A general Dirichlet Problem is the problem of finding a harmonic (continuous) function given its boundary values, for example finding a distribution of temperature across a piece of metal with known boundary temperature. However continuous cases can often be approximated by an appropriate discrete array of lattice points. In this context a well-known method used by physicists to find a good approximate of the solution is the so-called *method of relaxations*. It consists into iteratively picking a point among *interior points* and update it with the weighted average of its neighbors, thus locally meeting the *averaging property* above. Repeating this averaging process, going through interior points again and again, is resulting in a better and better estimate of the harmonic solution.

In the following we will consider algorithms similar to the method of relaxations, running on graphs such as the one in Fig. 2 (presented here with the harmonic solution to the Dirichlet Problem).

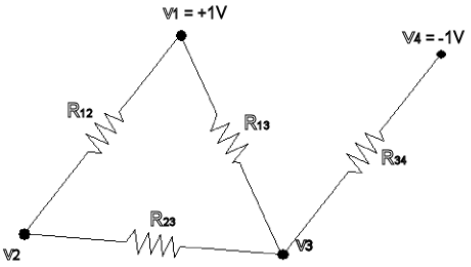

Fig. 1. Electrical network

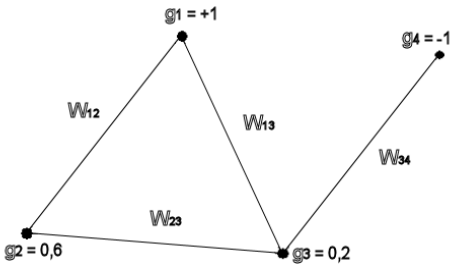

Fig. 2. Equivalent graph and harmonic solution

3 Notation

Let us define the nomenclature used throughout this paper: we denote vectors by bold letters, e.g. $\mathbf{x}$ and vector coordinates using an index, e.g. x_i for coordinate i of vector $\mathbf{x}$. Matrices are capital bold letters, e.g. $\mathbf{A}$, and matrix entries use a double indexation, e.g. A_{ij} for entry (i, j) of matrix $\mathbf{A}$. The transposition operator is noted $^\top$. The $k \times k$ identity matrix is $\mathbf{I}_k$, the $k \times 1$ vector of ones (zeros) is $\mathbf{1}_k$ ($\mathbf{0}_k$), and the $k \times l$ matrix of ones (zeros) is $\mathbf{1}_{k,l}$ ($\mathbf{0}_{k,l}$).

The dimensionality of the space we consider is n, except when stated. The Euclidian base of $\mathbb{R}^n$ is made of vectors $\mathbf{e_1} = [1, 0, \ldots, 0]$, $\mathbf{e_2} = [0, 1, \ldots, 0]$, ..., $\mathbf{e_n}$. In this space, l_1,l_2 and infinite norm are respectively denoted by $\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|$, $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^n x_i^2}$,and $\|\mathbf{x}\|_\infty := \max_{i\in\{1,\ldots n\}} x_i$. The dimensionality (number of elements) of a discrete set S is noted $|S|$.

4 Graphs and Laplacian Spectrum

In this article we consider undirected simple graphs, connected, without loops (self-edges). That means, there can not be more than one edge between two vertices, there exists a path between any two vertices, and no edge is linking one vertex to itself. Let G be such a graph with vertex set $V = \{1, \ldots n\}$, edge set $E(G) := E \subseteq \{(i, j)\}_{i,j\in V,\, i<j}$ and $n \times n$ weight matrix $\mathbf{A}$ such that $A_{ij} = 1$ if $(i, j) \in E$ and zero otherwise. This setting corresponds to an unweighted graph, but most results presented in this paper can easily be extended to weighted graphs. The graph Laplacian $\mathbf{L}(G)$, which we denote $\mathbf{L}$ when it is not ambiguous, is the $n \times n$ matrix defined as $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}(d_1, \ldots, d_n)$ and d_i is the degree of vertex i, $d_i = \sum_{j=1}^n A_{ij}$. In Fig. 2 we have shown a simple graph composed of 4 vertices and 4 edges. Its weight matrix and Laplacian matrix are respectively

$$A = \begin{bmatrix} 0&1&1&0 \\ 1&0&1&0 \\ 1&1&0&1 \\ 0&0&1&0 \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 2&-1&-1&0 \\ -1&2&-1&0 \\ -1&-1&3&-1 \\ 0&0&-1&1 \end{bmatrix} .$$

$\mathbf{L}$ is symmetric positive semi-definite with real eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$.A number of properties have been proven, linking eigenvalues of the Laplacian matrix and properties of the graph (see [Moh91] for a detailed survey). Among them, the following give a good intuition of the role played by the smallest non-zero and the largest eigenvalue.

1. The smallest eigenvalue of $\mathbf{L}$ is $\lambda_1 = 0$ and correspond to eigenvectors which are piecewise constant on the components of G. The multiplicity of the 0 eigenvalue is equal to the number of components of G so if G is connected (1 component), $\lambda_2 > 0$.
2. $\lambda_2 \geq \dfrac{4}{n\, diam(G)}$ where $diam(G)$ denotes the diameter of G, that is the largest *distance* between any two vertices of G.
3. if G is connected, $\lambda_2 \geq 2\mu(G)(1 - \cos\frac{\pi}{n})$ where $\mu(G)$ is the *edge-connectivity* of G, that means, the number of edges whose deletion would disconnect G.
4. $\lambda_n \leq \max\{d_u + d_v, (u, v) \in E(G)\}$,

5. if G is a simple graph then $\lambda_n \leq n$ with equality if and only if the complement of G is not connected.

As we can guess from the lower bounds, λ_2 (often called *algebraic connectivity* of G, or Fiedler constant) represents how 'connected' G is. The higher the value of λ_2 is, the more connected G is. On the other hand, the lower λ_2 is, the better partitioned G can be. For example, the path graph with n vertices has an algebraic connectivity of $\lambda_2 = O(\frac{1}{n^2})$, a n-dimensional hypercube has a $\lambda_2 = 2$, whereas for a fully connected graph with n vertices (often called a *complete* graph), $\lambda_2 = n$.

5 Hilbert Space of Functions on a Graph

Let $\mathcal{R}(G)$ be the linear space of real-valued functions defined on the graph G, i.e., an n-dimensional vector space whose elements are the real vectors $\mathbf{g} = (g_1, \ldots, g_n)^\top$. As seen in [HPW05], we can define a linear subspace $\mathcal{H}(G)$ of $\mathcal{R}(G)$ which is orthogonal to the eigenvectors of $\mathbf{L}$ with zero eigenvalue, that is, $\mathcal{H}(G) := \{\mathbf{g} : \mathbf{g}^\top \mathbf{u_i} = 0, i = 1, \ldots, r\}$. Since we consider a G that is connected, $\mathbf{L}$ has only one eigenvector with eigenvalue zero (the constant vector) and therefore

$$\mathcal{H}(G) := \{\mathbf{g} : \sum_{i=1}^{n} g_i = 0\}. \tag{1}$$

On $\mathcal{R}(G)$ we define a semi-inner product $\langle \mathbf{f}, \mathbf{g} \rangle_\mathbf{L} := \mathbf{f}^\top \mathbf{L} \mathbf{g} = \sum_{(i,j)\in E(G)} A_{ij}(f_i - f_j)(g_i - g_j)$ with its associated semi-norm

$$\|\mathbf{g}\|_\mathbf{L}^2 = \mathbf{g}^\top \mathbf{L} \mathbf{g} = \sum_{(i,j)\in E(G)} A_{ij}(g_i - g_j)^2. \tag{2}$$

On $\mathcal{H}(G)$, $< ., . >$ is an inner-product and $\|.\|_\mathbf{L}$ is a norm. An obvious analogy with the *electrical energy dissipation* can be noticed, where the edges are replaced with resistors $R_{ij} = A_{ij}^{-1}$.

6 Minimization Using Coordinate Descent

6.1 Properties of Strongly Convex Quadratic Forms

We now consider strongly convex quadratic forms of $\mathbb{R}^n$. This means that such functions can be written

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f(\mathbf{x}) = \mathbf{x}^\top \mathbf{H} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \tag{3}$$

where the Hessian $\mathbf{H}$ is a positive definite real matrix, with positive eigenvalues. *Note:* we only consider functions which are not identically equal to $+\infty$. *Strongly convex* functions have some remarkable properties, described in detail in [BV04]. In particular, [BV04, §9.1.2] and well-known results of algebra yields:

Theorem 1. *A strongly convex quadratic form f with domain **dom** $f = \mathbb{R}^n$ has got one unique minimum $f(\mathbf{x}^*)$ on $\mathbb{R}^n$, satisfying $\|\nabla f(\mathbf{x}^*)\|_2 = 0$.*

Theorem 2. *Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(\mathbf{x})$ be a strongly quadratic form, and let $0 < \mu_1 \leq \ldots \leq \mu_n$ be the eigenvalues of its Hessian matrix $\mathbf{H}$. Then for all $(\mathbf{x}, \mathbf{y})$ in $\mathbb{R}^n \times \mathbb{R}^n$*

$$\frac{\mu_1}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) \leq \frac{\mu_n}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \ . \tag{4}$$

Insert Comments about these inequalities ?

This can be used to bound the distance between an input $\mathbf{x}$ and the optimal input $\mathbf{x}^*$ minimizing f:

Corollary 1. *Under the same assumptions as Theorem 2*

$$f(\mathbf{x}) - \frac{1}{2\mu_1}\|\nabla f(\mathbf{x})\|_2^2 \leq f(\mathbf{x}^*) \leq f(\mathbf{x}) - \frac{1}{2\mu_n}\|\nabla f(\mathbf{x})\|_2^2$$

and $$\frac{\mu_1}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \leq f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{\mu_n}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2$$

6.2 Steepest Coordinate Descent Algorithm

We now want to find the unique solution $\mathbf{x}^*$ to the minimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) := \mathbf{x}^\top \mathbf{H}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \tag{5}$$

where f is a strongly convex quadratic form as described previously. We use a steepest coordinate descent algorithm producing a minimizing sequence $\mathbf{x}^{(k)}$, $k = 1, 2, \ldots$, recursively obtained from the starting point $\mathbf{x}^{(0)}$ by the update rule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)} \Delta\mathbf{x}^{(k)} \tag{6}$$

where the *search direction* $\Delta\mathbf{x}^{(k)}$ is chosen to be the steepest descent direction corresponding to l_1-norm, that is, the coordinate direction along which the gradient of f is the biggest:

$$\Delta\mathbf{x}(i) = -\frac{\partial f(\mathbf{x})}{\partial x_i}\mathbf{e_i} \quad \text{where} \quad i = \arg\max_{i=1,\ldots,n} \left|\frac{\partial f(\mathbf{x})}{\partial x_i}\right| \tag{7}$$

We assume that an *exact line search* is done, that means, at every update, $t^{(k)}$ is chosen to minimize f along the ray $\{\mathbf{x} + t\Delta\mathbf{x}(i) \mid t \geq 0\}$.

Theorem 3. *The steepest coordinate descent algorithm with exact line search converges* linearly *in the number of steps k:*

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq c^k \left[f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right] \tag{8}$$

where $c = 1 - \frac{m}{n\,M}$ and m (resp. M) is any lower (upper) bound of the smallest (largest) eigenvalue of the Hessian matrix $\nabla^2 f(.) = \mathbf{H}$. Besides the maximum squared error made on a coordinate of $\mathbf{x}^{(k)}$ is bounded:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_\infty^2 \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_2^2 \leq \frac{M}{m} c^k \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 \leq \frac{\sqrt{n}M}{m} c^k \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_\infty^2$$

Proof. The proof of this theorem can be derived similarly to the proof for simple gradient descent in [BV04, §9.3.1], by noticing that the norms $\|.\|_2$ and $\|.\|_\infty$ are equivalent and in particular that for all $\mathbf{x}$ in $\mathbb{R}^n$, $\|\mathbf{x}\|_2 \geq \|\mathbf{x}\|_\infty \geq \frac{1}{\sqrt{n}}\|\mathbf{x}\|_2$. □

Corollary 2. *The maximum number of steps to perform in order to reach $f(\mathbf{x}^*)$ within a tolerance ϵ (that means, so that $f(\mathbf{x}^{(k)}) \leq f(\mathbf{x}^*) + \epsilon$) is:*

$$k_{max} = \log\left[\frac{f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)}{\epsilon}\right] \log\left(1/c\right)^{-1}$$

Similarly, the number of steps to perform in order for the maximum error on a node to be less than $\epsilon' := \eta\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2$ is

$$k'_{max} = \log\left[\frac{M\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2}{m\epsilon'^2}\right] \log\left(1/c\right)^{-1} = \log\left[\frac{m\eta^2}{M}\right] \log\left(c\right)^{-1}$$

The expression of k_{max} in both formulae suggests that the number of iterations depends on how close to $\mathbf{x}^*$ the initial point is, and what the final required accuracy is. It also depends of M/m – which is an upper bound of the condition number of $\nabla^2 f(\mathbf{x}) = H$ over $\mathbb{R}^n$ – and of the dimensionality n of the space.

7 Relaxation Algorithms

Now that we have seen some remarkable convergence properties of steepest coordinate descent algorithm when used to minimize strongly convex quadratic forms of $\mathbb{R}^n$, we use these results on three learning problems on a graph G.

7.1 Learning on a Graph

We consider a graph G built from a collection of n samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Each vertex represents a sample, and edges represent some *similarity* concept between two vertices. We assume that some of the samples are *labelled*, say the first l, with labels $\{y_1, \ldots, y_l\}$, and that other samples come without labels. Let $G^{(K)}$ be the subgraph of G containing only labelled vertices, and $G^{(U)}$ the subgraph containing only unlabelled vertices; we assume that $G^{(U)}$ is connected. We define $\mathbf{L}^{(K)}$ and $\mathbf{L}^{(U)}$ to be their respective Laplacians. Finally let $E(K,U)$ be the set of edges between $G^{(K)}$ and $G^{(U)}$ in G. We consider the problem of learning a function g on the graph G, such that $g_i = y_i, \forall i = 1, \ldots, l$.

Batch algorithms have been proposed to learn functions on a graph [BN04, ZGL03]. These methods compute $\mathbf{g}$ by solving a squared linear system of n and $n-l$ equations respectively. Labels can then be obtained by traditional methods such as harmonic threshold or class mass normalization. An online implementation using the Laplacian kernel has also been proposed in [HPW05], and involves computing the pseudo-inverse of $\mathbf{L}$ offline and solving a linear system of l equations. In this paper we use an algorithm based on steepest coordinate gradient descent.

We consider the problem of learning a function $\mathbf{g}^*$ on G, solution to one of the following constrained optimization problems:

$$(\mathbf{P1}) \quad \min_{\mathbf{g}\in\mathbb{R}^n} \quad \{\mathbf{g}^\top \mathbf{L}\mathbf{g} : g_i = y_i, i = 1, \ldots, l\}$$

$$(\mathbf{P2}) \quad \min_{\mathbf{g}\in\mathbb{R}^n} \quad \{\mathbf{g}^\top (\mathbf{L} + a\,\mathbf{I}_n)\mathbf{g} : g_i = y_i, i = 1, \ldots, l, a > 0\}$$

$$(\mathbf{P3}) \quad \min_{\mathbf{g}\in\mathcal{H}(G)} \quad \{\mathbf{g}^\top \mathbf{L}\mathbf{g} : g_i = y_i, i = 1, \ldots, l\}$$

These problems all correspond to finding a smooth functional $\mathbf{g}$ realizing an exact interpolation on known vertices $1, \ldots, l$. The smoothness is defined for problem $(\mathbf{P1})$ and $(\mathbf{P3})$ by the Laplacian semi-norm of $\mathbf{g}$ on G ; for $(\mathbf{P2})$ it is a combination of the Laplacian semi-norm and the standard L_2-norm with a factor a. Finally, recent papers have underlined the interest of finding a function compatible with the Hilbert space $\mathcal{H}(G)$ defined on the graph: this supplementary constraint is present here in problem $(\mathbf{P3})$. We now prove in the following that these problems are equivalent to minimization of quadratic forms with positive definite Hessian. In 7.4 we derive a steepest coordinate descent algorithm as a series of simple updates 'on the vertices of G'.

explain better

7.2 Problems $(\mathbf{P1})$ and $(\mathbf{P2})$

Equivalent Unconstrained Problems.

For problems $(\mathbf{P1})$ and $(\mathbf{P2})$ we prove that F is equivalent to a quadratic form $\mathbb{R}^{n-l}$. To achieve this we express the equality constraints in a matrix form: $\mathbf{C}\mathbf{g} = \mathbf{y}$ where $\mathbf{y}$ is a $l \times 1$ vector $\mathbf{y} = [y_1 \ldots y_l]^\top$ and $\mathbf{C}$ is a $l \times n$ matrix $\mathbf{C} = [\mathbf{I}_l | \mathbf{0}_{l,n-l}]$. A particular solution of this equation is $\mathbf{q} = [y_1 \ldots y_l\, 0 \ldots 0]^\top$. Constrained problems $(\mathbf{P1})$ and $(\mathbf{P2})$ can then be written as *unconstrained* problems:

$$(\mathbf{P1}') \quad \min_{\mathbf{z}\in\mathbb{R}^{n-l}} \{F_R(\mathbf{z}) := (\mathbf{q} + \mathbf{R}\mathbf{z})^\top \mathbf{L}(\mathbf{q} + \mathbf{R}\mathbf{z})\} \tag{9}$$

$$(\mathbf{P2}') \quad \min_{\mathbf{z}\in\mathbb{R}^{n-l}} \{F_{aR}(\mathbf{z}) := (\mathbf{q} + \mathbf{R}\mathbf{z})^\top (\mathbf{L} + a\,\mathbf{I}_n)(\mathbf{q} + \mathbf{R}\mathbf{z}), a > 0\} \tag{10}$$

where $\mathbf{R} \in \mathbb{R}^{n\times(n-l)}$ is any matrix whose range is the nullspace of $\mathbf{C}$. We choose $\mathbf{R} = [\mathbf{0}_{l,n-l} | \mathbf{I}_{n-l}]^\top$. The functions F_R and F_{aR} are quadratic forms of $\mathbb{R}^{(n-l)}$. We show in the next paragraph that their respective Hessian matrices are positive definite.

Eigenvalues of the Hessian Matrix.

Let $\mathbf{L}|_U := \mathbf{R}^\top \mathbf{L}\mathbf{R}$ be the $(n-l) \times (n-l)$ matrix containing the entries of L corresponding to unknown samples. Let $\lambda_1 \leq \ldots \leq \lambda_n$ and $\mu_1 \leq \ldots \leq \mu_{n-l}$ be the eigenvalues of $\mathbf{L}$ and $\mathbf{L}|_U$ respectively. $\mathbf{L}|_U$ is diagonally dominant and positive semi-definite so $\mu_1 \geq 0$. $\mathbf{L}$ is a real $n \times n$ symmetric matrix and $\mathbf{R}$ is a $n \times (n-l)$ matrix with orthonormal columns. We recall the following well-known theorem:

Theorem 4. ***(Interleaving eigenvalues theorem.)*** *Let $\mathbf{A}$ be a real $p \times p$ symmetric matrix with eigenvalues $a_1 \leq \ldots \leq a_p$. Let $\mathbf{X}$ be a $p \times q$ matrix with orthonormal columns ($q \leq p$), and let the $q \times q$ matrix $\mathbf{B} = \mathbf{X}^\top \mathbf{A}\mathbf{X}$ have eigenvalues $b_1 \leq \ldots \leq b_q$. Then $\forall i = 1, \ldots, q$, $a_i \leq b_i \leq a_{i+p-q}$.*

We are now able to characterize eigenvalues of $\mathbf{L}|_U$ with respect to the Laplacian eigenvalues:

Lemma 1. *The extreme eigenvalues of $\mathbf{L}|_U$ are bounded according to $\mu_1 \geq \frac{l}{n+l}\lambda_2$ and $\mu_{n-l} \leq \lambda_n$.*

Proof. Let $\mathbf{z_1}$ be a non-zero eigenvector of $\mathbf{L}|_U$ associated with the smallest eigenvalue μ_1. Let $\mathbf{g_1} := R\mathbf{z_1} - \tilde{z}\mathbf{1}_n$ be the orthogonal projection of $R\mathbf{z_1}$ on $\mathcal{H}(G)$. $\mathbf{g_1}$ is found simply by removing the mean $\tilde{z} := \frac{1}{n}(\mathbf{1}_n^\top R\mathbf{z_1})$ of $R\mathbf{z_1}$ from every coordinate of $R\mathbf{z_1}$. Since $(R\mathbf{z_1})_i = 0$ for $i = 1, \ldots, l$,

$$\mathbf{g_1}^\top \mathbf{g_1} = \sum_{i=1}^{n} (g_{1i})^2 = l\tilde{z}^2 + \sum_{i=l+1}^{n} (g_{1i})^2 \geq l\tilde{z}^2 \tag{11}$$

Besides Pythagorean theorem states that $(R\mathbf{z_1})^\top (R\mathbf{z_1}) = \mathbf{g_1}^\top \mathbf{g_1} + \frac{1}{n}(\mathbf{u_1}^\top R\mathbf{z_1})^2$, so from inequality (11) we get

$$\mathbf{z}_1^\top \mathbf{z}_1 = (R\mathbf{z_1})^\top (R\mathbf{z_1}) \leq \left(1 + \frac{n}{l}\right) \mathbf{g_1}^\top \mathbf{g_1} \tag{12}$$

And therefore

$$\mu_1 = \frac{\mathbf{z_1}^\top \mathbf{L}|_U \mathbf{z_1}}{\mathbf{z_1}^\top \mathbf{z_1}} = \frac{\mathbf{g_1}^\top \mathbf{L}\mathbf{g_1}}{(R\mathbf{z_1})^\top (R\mathbf{z_1})} \geq \frac{l}{n+l} \frac{\mathbf{g_1}^\top \mathbf{L}\mathbf{g_1}}{\mathbf{g_1}^\top \mathbf{g_1}} \tag{13}$$

The Rayleigh-Ritz characterization of eigenvalues of $\mathbf{L}$ states that for all $\mathbf{g}$ in $\mathcal{H}(G)$ non-zero,

$$\lambda_n \geq \frac{\mathbf{g}^\top \mathbf{L}\mathbf{g}}{\mathbf{g}^\top \mathbf{g}} \geq \lambda_2 \tag{14}$$

Because $\mathbf{z}_1$ is non-zero, it is straightforward to see that $\mathbf{g_1}$ is non-zero. Therefore, we can combine (13) and (14) to obtain

$$\mu_1 \geq \frac{l}{n+l}\lambda_2 \tag{15}$$

The second part of the proof is directly obtained from Theorem 4. □

Note: we can prove that this bound is semi-tight by considering the case where G is a complete graph ($\lambda_2 = n$). In that case $G^{(U)}$ is a complete graph as well (eigenvalues $0, (n-l), \ldots, (n-l)$), and $\mathbf{L}|_U$ can be decomposed in $\mathbf{L}^{(U)} + l\,\mathbf{I}_{n-l}$.The eigenvalues of $\mathbf{L}|_U$ are $0 + l, (n-l) + l, \ldots, (n-l) + l$: $\mu_1 = l$. The bound is then $l \geq l\frac{n}{n+l}$ so we can immediately notice that we can make the right hand side as close as we want to the left hand side by using increasing values of n.

Finally we prove that both problems $(\mathbf{P1})$ and $(\mathbf{P2})$ are quadratic forms of $\mathbb{R}^{(n-l)}$ with positive definite Hessians. Indeed the respective Hessian matrices are $\nabla^2 F_R(\mathbf{z}) = 2\mathbf{L}|_U$ with eigenvalues $\{2\mu_1, \ldots, 2\mu_{n-l}\}$ and $\nabla^2 F_{aR}(\mathbf{z}) = 2(\mathbf{L}|_U + a\,\mathbf{I}_{n-l})$ with eigenvalues $\{2(\mu_1+a), \ldots, 2(\mu_{n-l}+a)\}$.

7.3 Problem $(\mathbf{P3})$

Equivalent Unconstrained Problem.

We express problem $(\mathbf{P3})$ as an unconstrained problem. To achieve this we express the equality in a matrix form: $\mathbf{C}^+\mathbf{g} = [y_1 \ldots y_l\, 0]^\top$ where $\mathbf{C}^+$ is the $(l+1) \times n$ matrix $\mathbf{C}^+ = \left[\frac{\mathbf{I}_l | \mathbf{0}_{l,n-l}}{\mathbf{1}_n^\top}\right]$. Let $s \in \mathbb{R}^n$ be a solution to this constraint and $\mathbf{Q} \in \mathbb{R}^{n \times (n-l)}$ any matrix whose range is the nullspace of $\mathbf{C}^+$. Problem $(\mathbf{P3})$ can then be written as an *unconstrained* problem:

$$(\mathbf{P3'}) \quad \min_{\mathbf{z} \in \mathbb{R}^{n-l}} \{F_H(\mathbf{z}) := (\mathbf{s} + \mathbf{Q}\mathbf{z})^\top \mathbf{L}(\mathbf{s} + \mathbf{Q}\mathbf{z})\} \tag{16}$$

Let $\bar{y} := \frac{1}{n-l}\sum_{k=1}^{l} y_k$. For the following we choose $\mathbf{s} = \left[\mathbf{y}^\top | -\bar{y}, \ldots, -\bar{y}\right]^\top$, and $\mathbf{Q} = \left[\frac{\mathbf{0}_{l,n-l}}{\mathbf{I}_{n-l} - \frac{1}{n-l}\mathbf{1}_{n-l,n-l}}\right]$. The function F_H is a quadratic form of $\mathbb{R}^{(n-l)}$.

Since the dimensionality of the problem is $n-l-1$, an even simpler form can be found in $\mathbb{R}^{n-l-1}$, with a $\mathbf{Q} \in \mathbb{R}^{n\times(n-l-1)}$. However in order to keep the balance between all vertices and in particular to have a nicer coordinate descent step we have chosen to add an useless dimension to the problem. Hence the reader might guess that the resulting function's Hessian is NOT positive definite and that we might not be able to use previous results. Nevertheless we remark that $F_H(\mathbf{z}+\alpha\mathbf{1}_{n-l}) = F_H(\mathbf{z})$ for any $\mathbf{z} \in \mathbb{R}^{n-l}$ and $\alpha \in \mathbb{R}$, so we can equivalently study the restriction of F_H to the lower dimensional space $\mathcal{H}^{(U)} := H(G^{(U)}) = \{\mathbf{z} \in \mathbb{R}^{n-l} : \mathbf{z}^\top \mathbf{1}_{n-l} = 0\}$. We show in the next paragraph that its Hessian matrix has only positive eigenvalues on this subspace.

We immediately notice that a coordinate descent on F_H will necessarily end up outside of $\mathcal{H}^{(U)}$. As we have seen previously the problem is invariant by translation along the direction $\mathbf{1}_{n-l}$, so we are going to use a projection onto $\mathcal{H}^{(U)}$ along this direction after each step of the descent algorithm. We find out that the resulting method is behaving as expected in terms of speed (see Sect. 7.5).

Eigenvalues of the Hessian Matrix.

Note: For readability issues we keep the notation μ_i to denote eigenvalues of the matrix we consider.

Lemma 2. *The first and last eigenvalues of $\mathbf{Q}^\top\mathbf{L}\mathbf{Q}$ associated with eigenvectors in $\mathcal{H}^{(U)}$ are bounded according to $\mu_2 \geq \lambda_2$ and $\mu_{n-l} \leq \lambda_n$.*

Proof. For any $\mathbf{z}$ in $\left[\mathcal{H}^{(U)}\right]^\perp$, $\mathbf{Q}^\top\mathbf{L}\mathbf{Q}\mathbf{z} = \mathbf{0}_n$ so the smallest eigenvalue of $\mathbf{Q}^\top\mathbf{L}\mathbf{Q}$ is $\mu_1 = 0$, and the associated eigenvectors are constant vectors. We observe that $(\mathbf{Q}^\top\mathbf{Q}) = I_{n-l} - \frac{1}{n-l}1_{n-l,n-l}$ so for any $\mathbf{z} \in \mathcal{H}^{(U)}$, $(\mathbf{Q}^\top\mathbf{Q})\mathbf{z} = \mathbf{z}$ and

$$\frac{\mathbf{z}^\top(\mathbf{Q}^\top\mathbf{L}\mathbf{Q})\mathbf{z}}{\mathbf{z}^\top\mathbf{z}} = \frac{(\mathbf{Q}\mathbf{z})^\top\mathbf{L}(\mathbf{Q}\mathbf{z})}{(\mathbf{Q}\mathbf{z})^\top(\mathbf{Q}\mathbf{z})} \tag{17}$$

Finally for all $\mathbf{z} \in \mathbb{R}^{n-l}$, $\mathbf{Q}\mathbf{z}$ is in $\mathcal{H}(G)$ so from the Rayleigh-Ritz inequalities (14) we obtain $\mu_2 \geq 1 \times \lambda_2$ and $\mu_{n-l} \leq 1 \times \lambda_n$. □

Note: we can prove that this bound is semi-tight. We consider a complete graph; the eigenvalues of $\mathbf{L}$ are 0 with multiplicity 1, and n with multiplicity $(n-1)$. Then for any $\mathbf{z}$ in $\mathcal{H}^{(U)}$, $\mathbf{Q}\mathbf{z} \in \mathcal{H}(G)$ so necessarily $\mathbf{Q}\mathbf{z}_2$ is an eigenvector of $\mathbf{L}$ of eigenvalue n.

Finally the Hessian matrix $\nabla^2 F_H(\mathbf{z}) = 2\mathbf{Q}^\top\mathbf{L}\mathbf{Q}$ has only positive eigenvalues $\{2\mu_2, \dots, 2\mu_{n-l}\}$ on $\mathcal{H}^{(U)}$. To conclude this Section we proved that all three problems $(\mathbf{P1})$, $(\mathbf{P2})$ and $(\mathbf{P3})$ are equivalent to minimization of strongly convex quadratic forms. We can now apply Theorem 1 for $(\mathbf{P1})$ and $(\mathbf{P2})$, and a similar theorem holding for strongly convex closed functions in the case of $(\mathbf{P3})$. Therefore they all have a unique optimal solution on their respective domains. This solution can be approached by using any classic descent method, e.g. steepest descent.

7.4 Update Rules of the Relaxation Algorithms

We now work out the value of the gradient for each problem, so that we can find the $\mathbf{g}$ which sets coordinate k of the gradient to zero (this is coordinate gradient descent with exact line search).

Problems $(\mathbf{P1})$ and $(\mathbf{P2})$.

The k^{th} coordinate of the gradient of the energy for the equivalent unconstrained problem $(\mathbf{P1}')$ is $(\nabla F_R)_k = 2(\mathbf{R}^\top\mathbf{L}(\mathbf{q}+\mathbf{R}\mathbf{z}))_k$.

$$(\nabla F_R)_k = 2d_{k+l}z_k - \sum_{j:(k+l,j)\in E(G^{(U)})} 2z_j - \sum_{\substack{j\in G-G^{(U)}\\ :(k+l,j)\in E(G)}} 2y_j \tag{18}$$

Setting the $(i-l)^{th}$ coordinate to zero and using $\mathbf{g} = [y_1 \dots y_l\ z_1 \dots z_{n-l}]^\top$, we find a simple update formula:

$$g_i = \frac{1}{d_i}\sum_{j:(i,j)\in E(G)} g_j \tag{19}$$

Similarly from problem $(\mathbf{P2}')$ we obtain

$$g_i = \frac{1}{d_i + a} \sum_{j:(i,j)\in E(G)} g_j \tag{20}$$

Hence one step of coordinate descent with exact line search consists in setting coordinate g_i of the current vector $\mathbf{g}$ to the average of its neighbors for problem $(\mathbf{P1})$, and to a regularized average of its neighbors for $(\mathbf{P2})$. We point out that the resulting algorithm for $(\mathbf{P1})$ is exactly the algorithm applied in the so-called *method of relaxations* to solve the Dirichlet problem.

Problem $(\mathbf{P3})$.

Let $\bar{y} = \frac{1}{n-l}\sum_{i=1}^{l} y_i$ and $\bar{z} = \frac{1}{n-l}\sum_{i=1}^{n-l} z_i$. Let $d_i^{(K)}$ when $i > l$ denote the number of existing edges between the unknown vertex i and known vertices $j \in \{1, \ldots, l\}$; let $\delta(K,U) = \sum_{i=l+1}^{n} d_i^{(K)} = |E(K,U)|$ be the total number of edges between known vertices and unknown vertices. Finally let $\Phi_i = 2d_i^{(K)} - \frac{\delta(U,K)}{n-l}$. Φ_i is the difference between two times the number of edges between the unknown vertex i and known vertices, and the average of this number across all unknown vertices. We assume that before performing the update, $\mathbf{z}$ is such that $\bar{z} = 0$, that is, $\mathbf{g} \in \mathcal{H}(G)$. For example we start with $\mathbf{g} = \mathbf{s}$. The following two-steps update rule can be derived similarly to the ones in 7.4:

- update g_i with

$$g_i^+ = \frac{\displaystyle\sum_{(i,j)\in E(G)} g_j - \frac{\Phi_i g_i}{n-l} - \frac{1}{n-l}\sum_{(p,q)\in E(K,U)} (g_p - g_q)}{\displaystyle d_i - \frac{\Phi_i}{n-l}} \tag{21}$$

- update all the unknown vertices j in $\{l+1, \ldots, n\}$ (including vertex i) with

$$g_j^+ = g_j - \frac{1}{n-l}\sum_{k=1}^{n} g_k \tag{22}$$

The first part of this update rule is quite close to what we have found in previous problems, except that here we take the average of the neighbors AND the vertex i itself – with a coefficient depending on its connectivity. A constant term is also present ; it depends on the continuity of the labels at the border between known vertices and unknown vertices. It is interesting to note that for a large number $(n-l)$ of unknown vertices we tend to the same update rule than for problem $(\mathbf{P1})$.

Conclusion: Finally, coordinate descent with exact line search consists for these three problems in setting coordinate g_i of the current vector $\mathbf{g}$ to the average of its neighbors, a regularized average, or a simple function of other vertices. Of course, $\mathbf{g}$ is only updated on the *unknown* part of the graph.

7.5 Convergence of the Relaxation Algorithms

Now that we have found how to implement steepest coordinate descent algorithms in order to solve these three learning problems, we provide simple bounds for their convergence.

Theorem 5. *The steepest coordinate descent algorithms with exact line search described in Sect. 7.4 converge linearly in the number of steps k:*

$$\|\mathbf{g}^+\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2 \leq c^k \left(\|\mathbf{g}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2\right)$$

where $c = 1 - \frac{l\,\lambda_2}{(n^2-l^2)\,\lambda_n}$ for $(\mathbf{P1})$, $c = 1 - \frac{\frac{l}{n+l}\,\lambda_2 + a}{(n-l)\,\lambda_n + a}$ for $(\mathbf{P2})$, and $c = 1 - \frac{\lambda_2}{(n-l-1)\,\lambda_n}$ for $(\mathbf{P3})$.

relaxation? replace c by (1-c) ev-ery-where ?

Corollary 3. *The maximum number of steps to perform in order to reach $\|\mathbf{g}^*\|_{\mathbf{L}}^2$ within a tolerance ϵ is:*

$$k_{max} = \frac{\log\left(\dfrac{\|\mathbf{g}^{(0)}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2}{\epsilon}\right)}{\log(1/c)} \tag{23}$$

Proof. **Problems (P1) and (P2)**. We apply the bounds found in Sect. 6 for the function $F_R : \mathbf{z} \mapsto F(q + R\mathbf{z})$, with $\mathbf{dom} F_R = \mathbb{R}^{n-l}$. From the Rayleigh-Ritz inequalities (14) applied to $\mathbf{L}|_U$ the following holds:

$$\forall \mathbf{z} \in S, \quad m\, \mathbf{z}^\top \mathbf{z} \leq \mathbf{z}^\top \nabla^2 F_R(\mathbf{z}) \mathbf{z} \leq M\, \mathbf{z}^\top \mathbf{z} \tag{24}$$

with $m = 2\mu_1$ and $M = 2\mu_n$. We then proceed as in the proof of Theorem 3 with $c = 1 - \dfrac{\mu_1}{(n-l)\,\mu_{n-l}}$. It is possible to iterate recursively – the coordinate descent is done only on coordinates $(l+1, \dots n)$ – to find the equivalent of equation (8):

$$\|\mathbf{g}^{(k)}\|^2_{\mathbf{L}} - \|\mathbf{g}^*\|^2_{\mathbf{L}} \leq c^k \left(\|\mathbf{g}^{(0)}\|^2_{\mathbf{L}} - \|\mathbf{g}^*\|^2_{\mathbf{L}} \right) \tag{25}$$

From Lemma 1 we finally derive an upper bound of c with respect to eigenvalues of L. The proof is similar for $(\mathbf{P2})$. □

Proof. **Problem (P3)**. The function F_H is not strongly convex, only its restriction to $\mathcal{H}^{(U)}$ is. We perform a steepest coordinate gradient descent with exact line search on the function F_H. Recall that the current point is such that $\bar{z} = 0$ (see Sect. 7.4, item 3.) so $\mathbf{z} \in \mathcal{H}^{(U)}$. From current point $\mathbf{z}$, we go to point $\mathbf{z}^+(t) = \mathbf{z} - t \frac{\partial F_H(\mathbf{z})}{\partial z_i}\, \mathbf{e_i}$. However the strong convexity result in Theorem 2 holds only if both $\mathbf{y}$ and $\mathbf{x}$ are in $\mathcal{H}^{(U)}$. In order to use it, we consider the projection $\mathbf{z}'(t)$ of $\mathbf{z}^+(t)$ onto $\mathcal{H}^{(U)}$:

$$\mathbf{z}'(t) = \left(\mathbf{I}_{n-l} - \frac{1}{n-l} 1_{n-l,n-l} \right) \mathbf{z}^+(t) \tag{26}$$

This projection is done by translating $\mathbf{z}^+(t)$ along the invariance direction, so $F_H(\mathbf{z}'(t)) = F_H(\mathbf{z}^+(t))$. We now start from Theorem 2, replacing $\mathbf{y}$ with $\mathbf{z}'(t)$ and $\mathbf{x}$ with $\mathbf{z}$:

$$F_H(\mathbf{z}'(t)) \leq F_H(\mathbf{z}) + \nabla F_H(\mathbf{z})^\top (\mathbf{z}'(t) - \mathbf{z}) + \frac{M}{2} \|\mathbf{z}'(t) - \mathbf{z}\|_2^2 \tag{27}$$

The difference between $\mathbf{z}'$ and $\mathbf{z}$ is

$$\mathbf{z}'(t) - \mathbf{z} = t \frac{\partial F_H(\mathbf{z})}{\partial z_i} \left(\frac{1}{n-l} \mathbf{1}_{n-l} - \mathbf{e_i} \right) \tag{28}$$

and the squared norm of that difference is

$$\|\mathbf{z}'(t) - \mathbf{z}\|_2^2 = t^2 \|\nabla F_H(\mathbf{z})\|_\infty^2 \left(\frac{n-l-1}{n-l} \right) \quad . \tag{29}$$

The function F_H is invariant by translation along the direction $\mathbf{1}_{n-l}$ so its gradient is necessarily in $\mathcal{H}^{(U)}$. Therefore, the scalar product between $(\mathbf{z}'(t) - \mathbf{z})$ and the gradient of F_H is:

$$\nabla F_H(\mathbf{z})^\top (\mathbf{z}'(t) - \mathbf{z}) = -t \frac{\partial F_H(\mathbf{z})}{\partial z_i} \nabla F_H(\mathbf{z})^\top \mathbf{e_i} = -t \|\nabla F_H(\mathbf{z})\|_\infty^2 \tag{30}$$

so combining with (27) and writing $\tilde{F_H}(t, i)$ for $F_H(\mathbf{z}'(t))$ we obtain

$$\tilde{F_H}(t, i) \leq F_H(\mathbf{z}) - t\, \|\nabla F_H(\mathbf{z})\|_\infty^2 + \frac{M t^2}{2} \|\nabla F_H(\mathbf{z})\|_\infty^2 \left(\frac{n-l-1}{n-l} \right) \tag{31}$$

the following is similar to the standard proof, with $M \left(\frac{n-l-1}{n-l} \right)^2$ instead of M, see [BV04]. So the convergence results are still valid and in particular Theorem 5 holds for problem $(\mathbf{P3})$, with the following simpler upper bound for c: $c = 1 - \frac{(n-l)\lambda_2}{(n-l-1)^2\,\lambda_n} \leq 1 - \frac{\lambda_2}{(n-l-1)\,\lambda_n}$.

□

To conclude, in this Section we proved that all three minimization problems $(\mathbf{P1})$, $(\mathbf{P2})$, and $(\mathbf{P3})$ can be solved by a simple steepest coordinate descent algorithm, converging linearly with a rate depending on characteristic eigenvalues of $\mathbf{L}$.

Table 1. Relaxation Algorithms

Pb.	Objective Function	Constraints (dimensionality)	Update Rule ($i \in \{l+1, \dots, n\}$)	Ref.
$\mathbf{P1}$	$\mathbf{g}^\top \mathbf{L} \mathbf{g}$	$C\mathbf{g} = \mathbf{y}$ $(n-l)$	$g_i = \dfrac{S(i)}{d_i}$	(19)
$\mathbf{P2}$	$\mathbf{g}^\top (\mathbf{L} + a\,\mathbf{I}_n)\mathbf{g}$	$\mathbf{C}\mathbf{g} = \mathbf{y}$ $(n-l)$	$g_i = \dfrac{S(i)}{d_i + a}$	(20)
$\mathbf{P3}$	$\mathbf{g}^\top \mathbf{L} \mathbf{g}$	$\mathcal{H}(G)$,$\mathbf{C}\mathbf{g} = \mathbf{y}$ $(n-l-1)$	$g_i^+ = \dfrac{S(i) - \dfrac{\Phi_i g_i}{n-l} - \dfrac{1}{n-l} \displaystyle\sum_{(p,q)\in E(K,U)} (g_p - g_q)}{d_i - \dfrac{\Phi_i}{n-l}}$ $\forall j \in (l+1, \dots, n),\ g_j^+ = g_j - \frac{1}{n-l} \sum_{k=1}^{n} g_k$	(21)

Table 2. Convergence Constants

Pb.	Convergence Constant (c)	Ref.	Upper bound of c (1)	Upper bound of c (2)
$\mathbf{P1}$	$1 - \dfrac{l\,\lambda_2}{(n^2 - l^2)\,\lambda_n}$	Th. 5	$1 - \dfrac{4l}{(n^2 - l^2)\,n^2 D}$	$1 - \dfrac{4l}{(n^2 - l^2)\,n^2 (n-1)}$
$\mathbf{P2}$	$1 - \dfrac{\frac{l}{n+l}\,\lambda_2 + a}{(n-l)\,\lambda_n + a}$	Th. 5	$1 - \dfrac{\frac{4l}{(n+l)nD} + a}{(n-l)\,n + a}$	$1 - \dfrac{\frac{4l}{(n+l)(n-1)n} + a}{(n-l)\,n + a}$
$\mathbf{P3}$	$1 - \dfrac{\lambda_2}{(n-l-1)\,\lambda_n}$	Th. 5	$1 - \dfrac{4}{(n-l-1)\,n^2 D}$	$1 - \dfrac{4}{(n-l-1)\,n^2 (n-1)}$

7.6 Summary

The update rules and bounds of the four algorithms are summarized in tables 1 and 2. *Note*: $\mathbf{C}\mathbf{g} = \mathbf{y}$ denotes the constraint on labelled vertices $g_i = y_i$, $i = 1, \dots, l$; $S(i) := \sum_{(i,j)\in E(G)} g_j$ denotes the sum of values of the neighbors of vertex i. D denotes the diameter of G, i.e. the largest distance between any two vertices of G. Finally, $\Phi_i = 2d_i^{(K)} - \frac{\delta(U,K)}{n-l}$.

8 Discussion and Remarks

Complexity. The expressions for the convergence constants in Sect. 7.6 seem to be consistent with the meaning of λ_2: we can see that c decreases when λ_2 increases. This is not surprising because the more connected a graph is, the easier the information can flow from known vertices to unknown vertices, and the faster the algorithm should converge. From Theorem 5, the maximum number of steps, k_{max}, is of order $\frac{1}{1-c}$. It is interesting to note that if we keep the ratio l/n (quota of labelled vertices) constant, the worst case for problem $(\mathbf{P1})$ is $k_{max} = O(n^4)$, whereas for problem $(\mathbf{P2})$ it is only $k_{max} = O(n^2)$: it seems that the presence of an other smoothness measure prevents the algorithm to be too slow when n is large. For problem $(\mathbf{P3})$, the worst case is $k_{max} = O(n^4)$.

Bounding the Maximum Value of a Vertex. In problems $(\mathbf{P1})$ and $(\mathbf{P2})$ we choose to start the execution of the algorithm with $\mathbf{z}^{(0)}$ such that $\|\mathbf{z}^{(0)}\|_\infty \leq \|\mathbf{y}\|_\infty$. In such case the maximum value of $\mathbf{g}^{(k)}$ on the vertices is bounded: at any step $k > 0$:

$$\|\mathbf{g}^{(k)}\|_\infty \leq \max_{i=1,\dots,l} |y_i| = \|\mathbf{y}\|_\infty \tag{32}$$

Indeed, every update on a vertex is an average of the neighbors, and therefore will result in smaller absolute value than the one of the maximum known vertex.

Bounding the Maximum Squared Error. The application of Theorem 3 in Sect. 6.2 also provides us with a bound on the maximum squared error made on a vertex:

$$\|\mathbf{g}^{(k)} - \mathbf{g}^*\|_\infty^2 \leq \|\mathbf{g}^{(k)} - \mathbf{g}^*\|_2^2 \leq \frac{\sqrt{n}\, c^k}{d(1-c)} \|\mathbf{g}^{(0)} - \mathbf{g}^*\|_\infty^2 \tag{33}$$

where d stands for the dimensionality of the problem (resp. $n-l$, $n-l$ and $n-l-1$ for problems $(\mathbf{P1})$, $(\mathbf{P2})$ and $(\mathbf{P3})$). To prove this we note that $M/m = \frac{1}{d(1-c)}$ when $c = 1 - \frac{m}{d\,M}$, so if we use a bigger c (cf. the values of c using the λs) we obtain $M/m \leq \frac{1}{d(1-c)}$. We then can use (32) to eliminate $\mathbf{g}^*$ for problems $(\mathbf{P1})$ and $(\mathbf{P2})$:

$$\|\mathbf{g}^{(k)} - \mathbf{g}^*\|_\infty^2 \leq \|\mathbf{g}^{(k)} - \mathbf{g}^*\|_2^2 \leq \frac{4\sqrt{n}\, c^k}{d(1-c)} \|\mathbf{y}\|_\infty^2 \tag{34}$$

POTENTIALLY INSERT EQUIVALENT OF 34 FOR P3

Generalization to Non-connected Graphs. Most results in this paper can easily be extended to the general case where G is not necessarily connected. The minimum requirement is that there is a path between every vertex in $G^{(U)}$ and at least one labelled vertex in $G^{(K)}$ In other words if every unknown vertex has at least one way to 'receive' the information coming from the known vertices. In particular, λ_2 is replaced by the first non-zero eigenvalue of $\mathbf{L}$ in Lemmas 1, and 2.

9 Conclusion

We have presented and analyzed a new semi-supervised algorithm on graphs that fits labelled data exactly. The simplicity of the consensus algorithm makes it a natural and intuitive way of learning a function on large graphs, specially with a highly parallel architecture. On a sequential machine, an efficient implementation using a heap might outperform inverse-based and gradient methods for very large graphs.

References

[BN04] Mikhail Belkin and Partha Niyogi. Semi-supervised learning on riemann manifolds. *Machine Learning*, 56:209–239, 2004.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. chapters 9 and 10.

[DS00] Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks*. Mathematical Assoc. of America, 2000.

[HPW05] Mark Herbster, Massimiliano Pontil, and Lisa Wainer. Online learning over graphs. In *22nd Int. Conf. Machine Learning (ICML' 05)*, 2005.

[HUL93] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and Minimization Algorithms*. Springer-Verlag, 1993. chapter IV.

[Moh91] Bojan Mohar. *The laplacian spectrum of graphs*, volume 2, pages 871–898. Y. Alavi, G. Chartrand, O. R. Oellermann, A. J. Schwenk, Wiley, 1991.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

A Discussion and Remarks

A.1 Bounding the Maximum Value of a Vertex

Theorem 6. *For problem* $(\mathbf{P3})$*,*

$$\|\mathbf{g}\|_\infty \leq \|\mathbf{y}\|_\infty \sqrt{\frac{2l\overline{d}}{\lambda_2}} \tag{35}$$

where $\overline{d} = \frac{1}{l}\sum_{i=1}^{l} d_i$ *is the average degree of the known vertices.*

Proof. All unknown vertices are initially assigned to value $-\bar{y}$. Using the Rayleigh-Ritz characterization of eigenvalues we get

$$\|\mathbf{g}\|_{\mathbf{L}}^2 \geq \lambda_2 \|\mathbf{g}\|_2^2 \geq \lambda_2 \|\mathbf{g}\|_\infty^2 \tag{36}$$

Besides $\mathbf{g}^{(0)} = \mathbf{s} = \left[\mathbf{y}^\top | -\bar{y}, \ldots, -\bar{y}\right]^\top$ so, recalling that $d_i^{(U)}$ is the number of edges between vertex i and unknown vertices, we have

$$\begin{aligned}
\|\mathbf{g}^{(0)}\|_{\mathbf{L}}^2 &= \sum_{(i,j)\in E(G)} (g_i - g_j)^2 = \sum_{(i,j)\in E(G)\setminus E(G^{(U)})} (g_i - g_j)^2 \\
&= \sum_{(i,j)\in E(G^{(K)})} (y_i - y_j)^2 + \sum_{i=1}^{l} d_i^{(U)} (y_i + \bar{y})^2 \\
&\leq \left[\#(E(G)\setminus E(G^{(U)}))\right] \left(2\|\mathbf{y}\|_\infty\right)^2 && (37) \\
&\leq \left[\frac{1}{2}\sum_{i=1}^{n} d_i - \frac{1}{2}\sum_{i=l+1}^{n} d_i\right] \left(2\|\mathbf{y}\|_\infty\right)^2 && (38) \\
&\leq 2\left(\sum_{i=1}^{l} d_i\right) \|\mathbf{y}\|_\infty^2 = 2l\overline{d}\|\mathbf{y}\|_\infty^2 && (39) \\
& && (40)
\end{aligned}$$

MISSING : prove that $\|g^{(k+1)}\|_\infty \leq \|g^{(k)}\|_\infty$
so finally

$$\begin{aligned}
\|\mathbf{g}\|_\infty &\leq \frac{1}{\sqrt{\lambda_2}}\|\mathbf{g}\|_{\mathbf{L}} \leq \frac{1}{\sqrt{\lambda_2}}\|\mathbf{g}^{(0)}\|_{\mathbf{L}} \\
\|\mathbf{g}\|_\infty &\leq \|\mathbf{y}\|_\infty \sqrt{\frac{2l\overline{d}}{\lambda_2}} && (41)
\end{aligned}$$

□

Claim 1 *(not proven) For problem* $(\mathbf{P3})$*, if* $l < \frac{n}{2}$

$$\|\mathbf{g}\|_\infty \leq \|\mathbf{y}\|_\infty \tag{42}$$

Counter-example when $l \geq \frac{n}{2}$: the path graph with 5 vertices, vertices 1, 4 and 5 being labelled with the same label x. In that case the solution to $(P3)$ is labels 2 and 3 labelled both with $\frac{-3x}{2}$.

A.2 Performance of a Pass Through Each Vertex

For problem $(\mathbf{P1})$, surprisingly enough, if we update every unknown vertex once (1 pass in any order, without taking into account which vertex is the 'steepest'–we denote this vertex by s), the progress is not necessarily bigger than one single step of steepest descent.

However, the following holds (for algorithms $(\mathbf{P1})$ and $(\mathbf{P2})$):

Claim 2 *If we sequentially update every unknown vertex once, in any particular order, either one of these updates will correspond to a steepest step (vertex with maximum tension), or at the end of the pass through the $n-l$ vertices we will have made more progress that the progress that we would have made with one steepest step. The tension of vertex i denotes the absolute value of the ith coordinate of the gradient of the objective function F_R; $T_i := |d_i g_i - S(i)|$ where $S(i) = \sum_{(i,j)\in E(G)} g_j$ is the sum over the neighbors of vertex i.*

Proof. to be rewritten more formally We prove this by showing that during the update process, either the vertex that we update now is the steepest, or the steepest (we denote this vertex by s) keeps being in the set of vertices that we have not updated yet, or we have already made more progress than the steepest descent progress. In other words we prove that the steepest vertex (s) can not become one of the vertices that we have already updated (i) unless we have made more progress than the one we would make by updating i again. We will use upper scripts $^{(k-)}$ and $^{(k+)}$ to refer to the value of a quantity just before and just after updating vertex k (this makes sense only if k has already been updated).
The vertex that we just updated is noted j. Let us consider a vertex i which has already been updated

$$g_i^{(j+)} = \frac{S(i)^{(i-)}}{d_i} \tag{43}$$

Now let us compute the tension of vertex i after updating j:

$$T_i^{(j+)} = \left| d_i g_i^{(j+)} - S(i)^{(j+)} \right| \tag{44}$$

but some neighbors of i might have been updated since we updated i:

$$S(i)^{(j+)} = S(i)^{(i-)} - \sum_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} \left(g_k^{(k-)} - \frac{S(k)^{(k-)}}{d_k} \right) \tag{45}$$

so finally

$$T_i^{(j+)} = \left| \sum_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} \left(g_k^{(k-)} - \frac{S(k)^{(k-)}}{d_k} \right) \right| \tag{46}$$

and

$$T_i^{(j+)} \leq \left(\sum_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} \frac{T_k^{(k-)}}{d_k} \right) \leq \sum_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} T_k^{(k-)} \tag{47}$$

So the tension of vertex i, that we have already updated and can not update any more in this pass, is less than the sum of the tensions of all its neighbors that we have updated since. Now let us look at the progress that each update is making. For one (not necessarily steepest) step along direction p we have a similar inequality than (88):

$$\|\mathbf{g}^{(p+)}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2 \leq \left(1 - \frac{m}{M} \frac{(T_p^{(p-)})^2}{\|\nabla F_R(\mathbf{g}^{(p-)})\|_2^2} \right) \left(\|\mathbf{g}^{(p-)}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2 \right) \tag{48}$$

so finally

$$\begin{aligned}
\frac{\|\mathbf{g}^{(j+)}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2}{\|\mathbf{g}^{(i+)}\|_{\mathbf{L}}^2 - \|\mathbf{g}^*\|_{\mathbf{L}}^2} &\leq \prod_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} \left(1 - \frac{m}{M} \frac{{T_k^{(k-)}}^2}{\|\nabla F_R(\mathbf{g}^{(k-)})\|_2^2} \right) \\
&\leq 1 - \frac{m}{M} \frac{\left(\sum_{\substack{k\in U_p(i\to j)\\(k,i)\in E(G)}} T_k^{(k-)} \right)^2}{\|\nabla F_R(\mathbf{g}^{(i+)})\|_2^2} \\
&\leq 1 - \frac{m}{M} \frac{{T_i^{(j+)}}^2}{\|\nabla F_R(\mathbf{g}^{(i+)})\|_2^2}
\end{aligned} \tag{49}$$

Therefore, if the new steepest descent vertex becomes one that we already updated (e.g. i in this proof), then the progress that we can not make by updating the steepest has necessarily been already done on the neighbors of i.

□

An alternative theorem based on probabilities:

Theorem 7. *Consider a coordinate gradient descent algorithm, where at each step the vertex to be updated is chosen randomly. This algorithm converges linearly in expectation, with a convergence constant $c^{1/n}$ where c is the convergence constant of the corresponding steepest coordinate descent algorithm (can be $(\mathbf{P1})$ or $(\mathbf{P2})$).*

Proof. (sketch) At each step there is a probability of $\frac{1}{n}$ of choosing the direction of steepest descent, therefore every n steps we expect to choose the steepest direction. □

A.3 Combining the Laplacian Distance with Another Distance Measure

Using a kernel matrix $\mathbf{K}$ instead of $\mathbf{I}_n$ in $\mathbf{g}^\top(\mathbf{L} + a\,\mathbf{I}_n)\mathbf{g}$ (problem $(\mathbf{P2})$) allows to use a combination of smoothness along the graph and with respect to the corresponding kernel. However, the corresponding update step becomes more complex since not only the neighborhood of considered vertex is used but also every other vertex:

$$g_i = \frac{1}{d_i + aK_{ii}} \left(\sum_{j:(i,j)\in E(G)} g_j - a \sum_{\substack{k=1 \\ k\neq i}}^{n} K_{ik} g_k \right) \tag{50}$$

A.4 Minimum Norm Interpolation (MNI)

Problem $(\mathbf{P3})$ can be solved directly by *Minimum norm interpolation* (MNI). The reproducing kernel of $\mathcal{H}(G)$ is the pseudo-inverse of the Laplacian $\mathbf{K} = \mathbf{L}^+$. With the representer theorem, we express the coordinates of $\mathbf{g}$ as

$$g_i = \sum_{j=1}^{l} K_{ij} c_j \tag{51}$$

The solution of $(\mathbf{P3})$ is given by $\mathbf{c} = \tilde{K}^+\mathbf{y}$ where $\tilde{K} = (K_{ij})_{i,j=1}^{l}$, see [HPW05]. However we need to compute $\mathbf{K} = \mathbf{L}^+$ first so the complexity is $O(n^3)$ and the amount of memory used is $O(n^2)$. **CHECK**

A.5 Adding a Second Smoothness Measure to $(\mathbf{P3})$

Similarly to problem $(\mathbf{P2})$ we can come up with a problem adding a second smoothness measure to $(\mathbf{P3})$:

$$\min_{\mathbf{g}\in\mathcal{H}(G)} \{\mathbf{g}^\top(\mathbf{L} + a\,\mathbf{I}_n)\mathbf{g} : g_i = y_i, i = 1, \ldots, l\} \tag{52}$$

The corresponding update rule is similar, with

$$g_i^+ = \frac{\displaystyle\sum_{(i,j)\in E(G)} g_j - \frac{(\Phi_i + a)g_i}{n-l} - \frac{1}{n-l} \sum_{(p,q)\in E(K,U)} (g_p - g_q) - a\bar{y}}{\displaystyle d_i + a - \frac{\Phi_i + a}{n-l}} \tag{53}$$

instead of (21). In order to find a convergence bound, we replace all the λ by $\lambda + a$ in Lemma 2, and the convergence constant c becomes:

$$c = 1 - \frac{\lambda_2 + a}{(n - l - 1)(\lambda_n + a)} \tag{54}$$

$$c \leq 1 - \frac{\frac{4}{nD} + a}{(n - l - 1)(n + a)} \tag{55}$$

$$c \leq 1 - \frac{\frac{4}{n(n-1)} + a}{(n - l - 1)(n + a)} \tag{56}$$

so we notice that the maximum number of steps, k_{max}, to perform in order to reach a chosen maximum error ϵ, is $O(n^2)$, like the algorithm for $(\mathbf{P2})$.

A.6 Regularization and Vertex Duplication

We consider *vertex duplication* on the graph G: through this process, a chosen known vertex i in G, with label y_i, is linked to a new vertex with the same label, with weight $\frac{1}{\alpha}$. The latter, called the duplicate, is considered as fixed (known), whereas the original vertex i becomes free (unknown) (see Fig. 3).

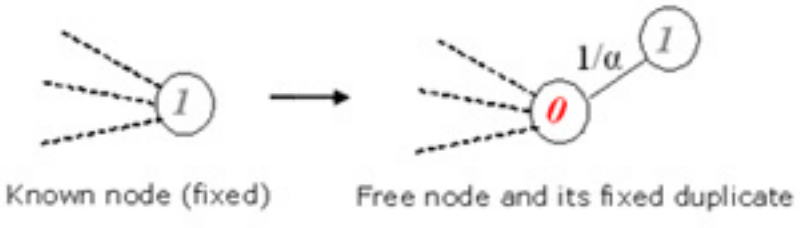

Fig. 3. Vertex duplication

α tunes how strongly the prior (label of the duplicate) is taken into account. $\alpha \to \infty$ is equivalent to not taking into account duplicate vertices at all, whereas $\alpha \to 0$ means the same than no duplication (i.e. the original node stays fixed). By extension we also consider the case where we associate a duplicate to an unknown node ($i > l$): in that case the label of the duplicate is chosen to match the one of the initial guess $\mathbf{g}^{(0)}$ (remember that for $(\mathbf{P1})$ and $(\mathbf{P2})$, $\mathbf{g}^{(0)} = \mathbf{q} = \left[\frac{\mathbf{y}}{0_{n-l,1}}\right]$, whereas for $(\mathbf{P3})$ it is $\mathbf{g}^{(0)} = \mathbf{s} = \left[\mathbf{y}^\top | -\bar{y}, \ldots, -\bar{y}\right]^\top$).

Now we consider the following scenarios :

1. Only unknown nodes in G are duplicated, hence the total number of nodes is now $2n - l$. We define $\mathbf{g''_D}$ a function on the new graph G''_D. $\mathbf{g''_D}^\top = [\mathbf{g}^\top | \mathbf{y_D}^{(U)\top}]$ where $\mathbf{y_D^{(U)}}$ is the vector of initial guesses for unknown nodes. The expression of the energy becomes:

$$\|\mathbf{g''_D}\|^2_{\mathbf{L''_D}} = \frac{1}{\alpha} \sum_{i=l+1}^{n} (y_D^{(U)}{}_i - g_i)^2 + \mathbf{g}^\top \mathbf{L} \mathbf{g} \tag{57}$$

We consider the minimum energy problem $(\mathbf{P1})$:

$$\min_{\mathbf{g''_D} \in \mathbb{R}^{2n-l}} \left\{ \|\mathbf{g''_D}\|^2_{L''_D} \,\middle|\, \begin{array}{c} g''_{D_i} = y_i,\ i = 1, \ldots, l \\ g''_{D_{i+n}} = y_D^{(U)}{}_i,\ i = 1, \ldots, (n-l) \end{array} \right\} \tag{58}$$

If we take the same initial guess as we did previously for $(\mathbf{P1})$ (that is, $\mathbf{y_D^{(U)}} = \mathbf{q} = \mathbf{0}_{n-l}$), the energy becomes

$$\|\mathbf{g''_D}\|^2_{L''_D} = \frac{1}{\alpha} \sum_{i=l+1}^{n} g_i^2 + \mathbf{g}^\top L \mathbf{g} \tag{59}$$

Hence the minimum energy problem is equivalent to

$$\min_{\mathbf{g}\in\mathbb{R}^n} \left\{ \mathbf{g}^\top \left(L + \alpha^{-1} I_n\right) \mathbf{g} \mid g_i = y_i,\ i = 1, \ldots, l \right\} \tag{60}$$

We can easily recognize problem $(\mathbf{P2})$ with $a = \alpha^{-1}$ here. It is a minimum Laplacian semi-norm problem regularized by the standard l_2 norm.

Finally, duplicating only unknown nodes in G and solving the minimum energy problem $(\mathbf{P1})$ is equivalent to solving $(\mathbf{P2})$ without any node duplication.

2. Every vertex in G is duplicated, hence the total number of nodes is now $2n$. We define $\mathbf{g_D}$ a function on the new graph G_D. $\mathbf{g_D}^\top = [\mathbf{g}^\top | \mathbf{y_D}^\top]$ where $\mathbf{g} = \mathbf{g_{D\,1,\ldots,n}}$ is the part of $\mathbf{g_D}$ corresponding to the original graph G, and $\mathbf{y_D} = \mathbf{g}^{(0)}$ is the vector of labels given to the duplicates. Then the expression of the energy becomes:

$$\|\mathbf{g_D}\|^2_{L_D} = \mathbf{g_D}^\top L_D \mathbf{g_D} = \frac{1}{\alpha} \sum_{i=1}^{n} (y_{D_i} - g_i)^2 + \mathbf{g}^\top L \mathbf{g} \tag{61}$$

Hence the minimum energy problem $(\mathbf{P1})$:

$$\min_{\mathbf{g_D}\in\mathbb{R}^{2n}} \left\{ \|\mathbf{g_D}\|^2_{L_D} | g_{D\,i+n} = y_{D_i},\ i = 1, \ldots, n \right\} \tag{62}$$

is equivalent to

$$\min_{\mathbf{g}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{n} (y_{D_i} - g_i)^2 + \alpha \mathbf{g}^\top L \mathbf{g} \right\} \tag{63}$$

This is a least squares problem, regularized by the Laplacian semi-norm.

3. Only known nodes in G are duplicated, hence the total number of nodes is now $n + l$. We define $\mathbf{g'_D}$ a function on the new graph G'_D. $\mathbf{g'_D}^\top = [\mathbf{g}^\top | \mathbf{y}^\top]$ where $\mathbf{y}$ is the vector of known labels that we used throughout this paper. The expression of the energy becomes:

$$\|\mathbf{g'_D}\|^2_{L'_D} = \frac{1}{\alpha} \sum_{i=1}^{l} (y_i - g_i)^2 + \mathbf{g}^\top L \mathbf{g} \tag{64}$$

Hence the minimum energy problem $(\mathbf{P1})$:

$$\min_{\mathbf{g'_D}\in\mathbb{R}^{n+l}} \left\{ \|\mathbf{g'_D}\|^2_{L'_D} | g'_{D\,i+n} = y_i,\ i = 1, \ldots, l \right\} \tag{65}$$

is equivalent to

$$\min_{\mathbf{g}\in\mathbb{R}^n} \left\{ \sum_{i=1}^{l} (y_i - g_i)^2 + \alpha \mathbf{g}^\top L \mathbf{g} \right\} \tag{66}$$

This is a least squares problem as well, regularized by the Laplacian semi-norm. However, here the least squares term only takes into account the error made on previously known nodes $(1, \ldots, l)$ and not any more the error between unknown nodes $(l + 1, \ldots, n)$ and their initial guess.

Scenario 3 is of particular interest because in that case regularization provides us with an important noise tolerance. Actually if the labels in $\mathbf{y}$ are not error-free (that is, if the expert who gives us the labels beforehand has some non-zero probability of making mistakes), our standard consensus algorithms – presented in previous sections – forces an *exact interpolation* leading to a function with high Laplacian semi-norm. On the contrary, using regularization by node duplication we allow for a compromise between an exact interpolation and a small semi-norm, which lowers the impact of errors on the solution.

TO QUANTIFY

A.7 Implementation for Large Graphs

For large graphs we consider an implementation based on a 'heap' data structure. This structure is well-known in computer science and used to implement efficient priority queues. The 'heap' is a nearly complete binary tree where each vertex represents a vertex of G, and contains three attributes:

the index of the vertex (e.g. $i \in V(G)$), the value of the 'tension' at that vertex (absolute value of the partial derivative $|d_i g_i - \sum_{(i,j)\in E(G)} g_j|$) , and the value taken at that vertex by the current interpolating function g_i (see Fig. 4, the tension value is represented in green). The definition of the 'tension' chosen here is a measure of the progress that we could make when updating this particular vertex with the learning algorithm. Here it has been chosen to fit the algorithm for $(\mathbf{P1})$.

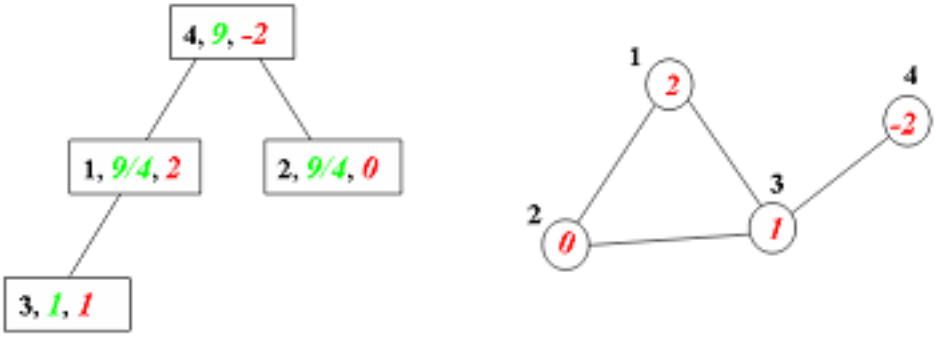

Fig. 4. Heap structure used for the simple graph

This structure allows us to implement steepest coordinate descent efficiently. Indeed, the heap is chosen to be a *max-heap* on the tension values, which means that every vertex of the heap has a tension higher or equal to any of its children. Then the steepest descent is done by updating the vertex (m) of the graph towards which the vertex at the top of the heap points. This vertex has maximum tension:

1. update vertex m indicated by the top vertex with the chosen update rule (cf. summary Sect. 7.6 for a comparison of the algorithms): its tension goes to zero and its value g_m changes.
2. Max-heapify the heap: this procedure reorganizes the heap so that the max-heap property is valid again (every vertex of the heap must have a tension higher or equal to any of its children). This operation is $O(\log_2 n)$.
3. For all the neighbors of vertex m, update their corresponding pointers in the heap, taking into account the new value g_m to compute the new tension value. Max-heapify the heap after each of these updates.

The total update step is $O(d_m \log_2 n)$ where d_m is the degree of vertex m. Therefore this implementation might be more efficient than an implementation cycling through all g_is (non-steepest descent) for large, sparse graphs.

Note: for problem $(\mathbf{P3})$, the tension measure would be different (it is the absolute value of the partial derivative), and besides we would have to add an extra step between steps 1. and 2. to update the pivot. Step 3. would act on both the neighborhood of vertex m and the pivot, since both changed values.

B Useful Results from Graph Theory

A number of properties have been proven in the field of Graph theory, linking eigenvalues of the Laplacian matrix and properties of the graph. Here is a summary of useful results (see [Moh91] for a detailed survey):

Theorem 8. *Let G be a graph of order n (n nodes). Then*

1. *$L(G)$ only has real eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$,*
2. *L(G) is positive semidefinite,*

3. *its smallest eigenvalue is* $\lambda_1 = 0$ *and correspond to eigenvectors which are piece-wise constant on the components of* G*. The multiplicity of the* 0 *eigenvalue is equal to the number of components of* G *so if* G *is connected (*1 *component),* $\lambda_2 > 0$*.*

4. $\lambda_2 \geq \dfrac{4}{n\, diam(G)}$ *where* $diam(G)$ *denotes the diameter of* G*, i.e. the largest* distance *between any two vertices of* G*. The distance* $d(u, v)$ *between two vertices* u *and* v *is defined as being the length of the shortest path on* G *between* u *and* v*.*

5. $\lambda_2 \geq \dfrac{2}{n-1}\left[\bar{\rho}(G) - \dfrac{n-2}{2(n-1)}\right]^{-1}$ *where* $\bar{\rho}(G)$ *is the mean distance, i.e. the average of all distances between distinct vertices of the graph.*

6. *if* G *is connected,* $\lambda_2 \geq 2\mu(G)(1 - \cos\frac{\pi}{n})$ *where* $\mu(G)$ *is the* edge-connectivity *of* G*, i.e. the number of edges whose deletion would disconnect* G*.*

7. $\lambda_2 \leq \dfrac{n}{n-1} \min\{d_v, v \in V(G)\}$,

8. $\lambda_n \leq \max\{d_u + d_v, (u, v) \in E(G)\}$,

9. *if* G *is a simple graph then* $\lambda_n \leq n$ *with equality if and only if the complement of* G *is not connected.*

Most of these results simply extend to the non connected case when replacing λ_2 with the first non-zero eigenvalue of the Laplacian matrix.

As we can guess from the lower bounds, λ_2 (often called *algebraic connectivity* of G, or Fiedler constant) represents how 'connected' G is. The higher the value of λ_2 is, the more connected G is. On the other hand, the lower λ_2 is, the better partitioned G can be. For example, the path graph with n vertices has an algebraic connectivity of $\lambda_2 = O(\frac{1}{n^2})$, a n-dimensional hypercube has a $\lambda_2 = 2$, whereas for a fully connected graph with n vertices (often called a *complete* graph), $\lambda_2 = n$.

C On Consensus and Social Networks

rather use the dirichlet problem example ?

One can see social networks as a set of pairwise relations between people: person A knows person B who does not know person C etc. We could represent this by a graph where vertices are people and there is an edge between two people if they know each other. When two people know each other they have the possibility to influence each other, up to a certain level (depending on the nature of the relation, etc.). Now, we consider the opinion of people about some topic. People can have any opinion, between strongly in favor(+1) and strongly against(-1)). If they do not have any opinion about that event, the corresponding value is zero (see Fig. 5).

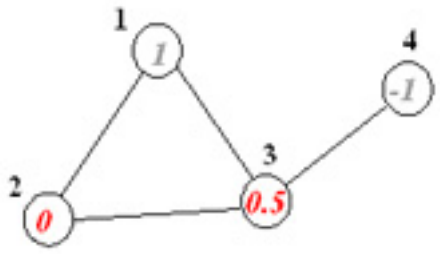

Fig. 5. Social network

On Fig. 5 for example person 2 does not have an opinion. Now let us consider that some people can be influenced by their neighbors (say, people 2 and 3), and that others (the 'leaders') cannot because their opinion is too strong (1 and 4). We expect that everybody who can be influenced will reach a consensus between all the opinions of the people he/she knows, in other words that the opinion of

everybody who can be influenced will tend to the average of the opinions of its neighbors (see Fig. 6).

group figures on 2 columns

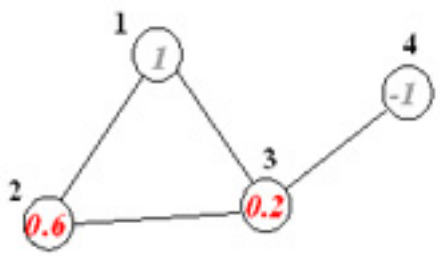

Fig. 6. Consensus

The algorithm presented in this paper implements such a consensus on graphs in order to learn an interpolating function.

intuitive and cf nature

D Strong Convexity and Implications

Note: The reader should refer to [BV04] for an interesting review on optimization methods and convergence, extensively used throughout this section.

First let us recall some notions on convexity of functions. Then we will see that strong convexity of a function f implies many interesting inequalities, which can be used to bound the distance between an input $\mathbf{x}$ and the optimal input $\mathbf{x}^*$ minimizing f. *Note:* in the following set of definitions and theorems, we consider only functions which are not identically equal to $+\infty$.

D.1 Basic Definitions

In the following we will use the following notation: a vector interval, $[\mathbf{x}, \mathbf{y}]$, is the set of all possible convex combination of $\mathbf{x}$ and $\mathbf{y}$:

$$[\mathbf{x}, \mathbf{y}] := \{\alpha\mathbf{x} + (1-\alpha)\mathbf{y}, \alpha \in [0,1]\}$$

Definition 1. *[BV04] A set C is convex if the line segment between any two points in C lies in C,* i.e.*, if for any $(\mathbf{x}_1, \mathbf{x}_2) \in C^2$ and any θ with $0 \leq \theta \leq 1$, we have*

$$\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in C$$

Definition 2. *[HUL93] Let U be a nonempty convex set in $\mathbb{R}^n$. A function $h : U \mapsto \mathbf{R}$ is said to be* convex *on U when for all pairs $(\mathbf{x}, \mathbf{y}) \in U^2$, and all $0 < \theta < 1$, there holds*

$$h(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta h(\mathbf{x}) + (1-\theta)h(\mathbf{y}) \tag{67}$$

We say that h is strictly convex *when strict inequality holds whenever $\mathbf{x} \neq \mathbf{y}$. An even stronger property is that there exists $c > 0$ such that for all $(\mathbf{x}, \mathbf{y}) \in U^2$ and $0 < \theta < 1$, there holds*

$$h(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta h(\mathbf{x}) + (1-\theta)h(\mathbf{y}) - \frac{c}{2}\theta(1-\theta)\|\mathbf{x}-\mathbf{y}\|_2^2 \tag{68}$$

In this case, h is said to be strongly convex *on U (with* modulus *of strong convexity c).*

So in other words, a function h is convex if every *chord* lies above the graph of h. *Strict convexity* is obtained when every non-trivial chord lies strictly above the graph of h. We note that some functions are strictly convex but not strongly convex, e.g. $x \mapsto e^x$ because the closer x gets to $-\infty$ the smaller a valid candidate for c has to be, so that we can not find a $c > 0$ verifying inequality (68).

Theorem 9. *[BV04] A function is convex if and only if it is convex when restricted to any line that intersects its domain.*

Note: a similar theorem holds for strict and strong convexity.

D.2 First-order Condition

Theorem 10. *[HUL93] Let U be a convex open subset of $\mathbb{R}^n$ and $h : U \mapsto \mathbf{R}$ a function continuously differentiable on U. Then h is strongly convex on U if and only if its gradient is strongly monotone on U, that is, if there $\exists c > 0$ such that*

$$\forall (\mathbf{x}, \mathbf{y}) \in U^2, \quad (\nabla h(\mathbf{y}) - \nabla h(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x}) \geq c\|\mathbf{y} - \mathbf{x}\|_2^2$$

D.3 Second-order Conditions

Theorem 11. *[HUL93] Let h be twice differentiable on an open convex set $U \subset \mathbb{R}^n$. Then*

1. *h is convex on U if and only if $\nabla^2 h(\mathbf{x}_0)$ is positive semi-definite for all $\mathbf{x}_0 \in U$;*
2. *if $\nabla^2 h(\mathbf{x}_0)$ is positive definite for all $\mathbf{x}_0 \in U$, then h is strictly convex on U.*
3. *h is strongly convex with modulus $c > 0$ on U if and only if the smallest eigenvalue of $\nabla^2 h(.)$ is minorized by c on U, hence for all $\mathbf{x}_0 \in U$ and all $d \in \mathbb{R}^n$,*

$$\mathbf{d}^\top \nabla^2 h(\mathbf{x}_0) \mathbf{d} \geq c\|\mathbf{d}\|^2$$

D.4 Closed Functions

Definition 3. *A set S is* closed *if it contains its boundary. It is* open *if it contains no boundary points.*

Definition 4. *[BV04] A function $h : \mathbb{R}^n \mapsto \mathbf{R}$ is said to be* closed *if, for each $\alpha \in \mathbb{R}$, the sublevel set $\{\mathbf{x} \in \textbf{\textit{dom}}\ h \ | h(\mathbf{x}) \leq \alpha\}$ is closed. If h is continuous, and **dom** h is closed, then h is closed. If h is continuous with **dom** h open, then h is closed if and only if h converges to ∞ along every sequence converging to a boundary point of **dom** h.*

Theorem 12. *[BV04] Continuous functions with **dom** $h = \mathbb{R}^n$ are closed.*

D.5 Minima

Theorem 13. *Let h be a convex function, continuous on $U \subset \mathbb{R}^n$. If h is strictly or strongly convex then it admits at most one minimum on U.*

Theorem 14. *Let h be a closed strongly convex function, C^1 on $\mathbb{R}^n$. Then h admits one unique minimum in $\mathbb{R}^n$.*

Proof. From Theorem 13, h admits at most one minimum on U. Now we consider a point $\mathbf{x}_0 \in \mathbb{R}^n$ and a direction $\mathbf{d} \in \mathbb{R}^n$, such that $\|\mathbf{d}\|_2^2 = 1$. From Theorem 10, for any $\alpha \in \mathbb{R}$,

$$(\nabla h(\mathbf{x}_0 + \alpha \mathbf{d}) - \nabla h(\mathbf{x}_0))^\top (\alpha \mathbf{d}) \geq c\|\alpha \mathbf{d}\|_2^2 \tag{69}$$

So

$$\alpha > 0 \quad (\nabla h(\mathbf{x}_0 + \alpha \mathbf{d}) - \nabla h(\mathbf{x}_0))^\top \mathbf{d} \geq c\alpha \tag{70}$$

$$\alpha < 0 \quad (\nabla h(\mathbf{x}_0 + \alpha \mathbf{d}) - \nabla h(\mathbf{x}_0))^\top \mathbf{d} \leq c\alpha \tag{71}$$

and we have the following limits:

$$\lim_{\alpha \to +\infty} \nabla h(\mathbf{x}_0 + \alpha \mathbf{d})^\top \mathbf{d} = +\infty \tag{72}$$

$$\lim_{\alpha \to -\infty} \nabla h(\mathbf{x}_0 + \alpha \mathbf{d})^\top \mathbf{d} = -\infty \tag{73}$$

Since the gradient is continuous, necessarily there exists a value of α for which it is zero, so h admits at least one minimum in direction $\mathbf{d}$. This is valid for any possible direction, therefore h admits at least one minimum on $\mathbf{R}^n$. □

D.6 Minimizing a Strongly Convex Function

Note: all results in this section are either taken directly or derived from results in [BV04].

We consider the problem (5) of minimizing a function $f : \mathbb{R}^n \to \mathbf{R}$ twice continuously differentiable on **dom** f (thus, **dom** f has to be open):

$$\min_{\mathbf{x} \in \mathbf{dom}\ f} f(\mathbf{x})$$

We assume that this problem has a solution $\mathbf{x}^*$. For $\mathbf{x}_0 \in dom(f)$, we define the sublevel set S:

$$S = \{\mathbf{x} \in \mathbf{dom}\ f | f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$$

We want to prove Theorem 1:

Theorem 1 1 *If f is twice continuously differentiable, strongly convex on the sublevel set S, and if S is closed, then if problem (5) has a solution $\mathbf{x}^*$ in S, it is unique and $\|\nabla f(\mathbf{x}^*)\|_2 = 0$, and besides $\exists (m, M) \in \mathbb{R}^2$ such that $\forall \mathbf{x} \in S$*

$$f(\mathbf{x}) - \frac{1}{2m}\|\nabla f(\mathbf{x})\|_2^2 \leq f(\mathbf{x}^*) \leq f(\mathbf{x}) - \frac{1}{2M}\|\nabla f(\mathbf{x})\|_2^2$$

$$\textit{and} \qquad \frac{m}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \leq f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{M}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2$$

m (resp. M) is any lower (upper) bound of the smallest (largest) eigenvalue of the Hessian $\nabla^2 f(.)$ on S.
Note*: if f is a closed function, any sublevel set S is closed so we don't need to check this condition. Besides in that case the existence of the solution is guaranteed by Theorem 14.*

Proof. We assume that f is twice continuously differentiable, strongly convex on the sublevel set S. From Theorem 14, f admits one unique minimum on S so problem (5) has a unique solution x^*. From Theorem 11,

$$\forall (\mathbf{x}, \mathbf{z}) \in S^2, \quad \mathbf{x}^\top \nabla^2 f(\mathbf{z}) \mathbf{x} \geq m \mathbf{x}^\top \mathbf{x} \tag{74}$$

Besides for $(\mathbf{x}, \mathbf{y}) \in S^2$ there exists some $\mathbf{z}$ in $[\mathbf{x}, \mathbf{y}]$ such that (Taylor expansion around $\mathbf{x}$)

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^\top \nabla^2 f(\mathbf{z})(\mathbf{y} - \mathbf{x}) \tag{75}$$

so from the strong convexity assumption (74),

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{m}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \tag{76}$$

This can be used to bound $f(\mathbf{x}) - f(\mathbf{x}^*)$, the suboptimality of point $\mathbf{x}$, in terms of $\|\nabla f(\mathbf{x})\|_2$: minimizing the right-hand side of (76) with respect to $\mathbf{y}$ is done for $\mathbf{y} = \mathbf{x} - \frac{1}{m}\nabla f(\mathbf{x})$ and then considering the particular case of $\mathbf{y} = \mathbf{x}^*$ in the left-hand side, we obtain

$$f(\mathbf{x}^*) \geq f(\mathbf{x}) - \frac{1}{2m}\|\nabla f(\mathbf{x})\|_2^2 \tag{77}$$

Inequality (76) implies that S is bounded. Therefore the maximum eigenvalue of $\nabla^2 f(\mathbf{x})$ is bounded above on S, i.e. $\exists M > 0$ such that

$$\forall \mathbf{x} \in S, \quad \mathbf{x}^\top \nabla^2 f(\mathbf{x}) \mathbf{x} \leq M \mathbf{x}^\top \mathbf{x} \tag{78}$$

Thus, for all $(\mathbf{x}, \mathbf{y}) \in S^2$,

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{M}{2}\|\mathbf{y} - \mathbf{x}\|_2^2 \tag{79}$$

which is the counterpart of inequality (76). Minimizing the right-hand side of (79) with respect to $\mathbf{y}$ is done for $\mathbf{y} = \mathbf{x} - \frac{1}{M}\nabla f(\mathbf{x})$ and then considering the particular case of $\mathbf{y} = \mathbf{x}^*$ in the left-hand side, we obtain

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) - \frac{1}{2M}\|\nabla f(\mathbf{x})\|_2^2 \tag{80}$$

from which, considering $\mathbf{x} = \mathbf{x}^*$, we get $\|\nabla f(\mathbf{x}^*)\|_2 = 0$: the gradient is necessarily zero at the optimal point $\mathbf{x}^*$. Replacing $\mathbf{y} = \mathbf{x}^*$ in equation (76) we have

$$\frac{m}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \leq f(\mathbf{x}^*) - f(\mathbf{x}) - \nabla f(\mathbf{x})^\top(\mathbf{x}^* - \mathbf{x})$$
$$\frac{m}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \leq f(\mathbf{x}^*) - f(\mathbf{x}) + \|\nabla f(\mathbf{x})\|_2\|\mathbf{x}^* - \mathbf{x}\|_2$$

We used Cauchy-Schwarz inequality here. Then, using $f(\mathbf{x}^*) - f(\mathbf{x}) \leq 0$

$$\|\mathbf{x}^* - \mathbf{x}\|_2 \leq \frac{2}{m}\|\nabla f(\mathbf{x})\|_2 \tag{81}$$

which relates how close a point is to the solution and how small the gradient at this point is. One consequence of (81) is that the optimal point x^* is unique. Finally, replacing $\mathbf{y} = \mathbf{x}$ and $\mathbf{x} = \mathbf{x}^*$ ($\nabla f(\mathbf{x}^*) = 0$) in equations (76) and (79) we also have

$$\frac{m}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \leq f(\mathbf{x}) - f(\mathbf{x}^*) \leq \frac{M}{2}\|\mathbf{x}^* - \mathbf{x}\|_2^2 \tag{82}$$

This last inequality expresses the relationship between how close an input is to the optimal input, and how close their corresponding values through f are. □

D.7 Steepest Coordinate Descent Algorithm

Note: the proof in this section is similar to the proof for simple gradient descent in [BV04].

We now use a steepest coordinate descent algorithm to find the unique solution $\mathbf{x}^*$ in the sublevel S to the optimization problem (5). We note that the strong convexity assumption, associated to the assumption that S is closed, ensures that there exists one unique minimum $f(\mathbf{x}^*)$. The algorithm produces a minimizing sequence $\mathbf{x}^{(k)}$, $k = 1, 2, \ldots$, recursively obtained from the starting point $\mathbf{x}^{(0)} = \mathbf{x}_0$ by the update rule

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)} \tag{83}$$

where the *search direction* $\Delta\mathbf{x}^{(k)}$ is chosen to be the steepest descent direction corresponding to l_1-norm, that is, the coordinate direction along which the gradient of f is the biggest:

$$\Delta\mathbf{x}(i) = -\frac{\partial f(\mathbf{x})}{\partial x_i}\mathbf{e_i} \quad \text{where} \quad i = \underset{i=1,\ldots,n}{\arg\max}\left|\frac{\partial f(\mathbf{x})}{\partial x_i}\right| \tag{84}$$

We assume that an *exact line search* is done, that means, at every update, $t^{(k)}$ is chosen to minimize f along the ray $\{\mathbf{x} + t\Delta\mathbf{x}(i) \mid t \geq 0\}$. Then, the following convergence result holds:

Theorem 15. *The steepest coordinate descent algorithm with exact line search converges* linearly *in the number of steps k:*

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq c^k\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right) \tag{85}$$

where $c = 1 - \dfrac{m}{n\,M}$ and m (resp. M) is any lower (upper) bound of the smallest (largest) eigenvalue of the Hessian $\nabla^2 f(.)$ on S. Thus, the maximum number of steps to perform in order to reach $f(\mathbf{x}^*)$ within a tolerance ϵ (i.e., so that $f(\mathbf{x}^{(k)}) \leq f(\mathbf{x}^*) + \epsilon$) is:

proof moved to appendix

$$k_{max} = \frac{\log\left(\dfrac{f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)}{\epsilon}\right)}{\log(1/c)}$$

Besides the maximum squared error made on a coordinate of $\mathbf{x}^{(k)}$ is bounded:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_\infty^2 \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_2^2 \leq \frac{M}{m}c^k\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 \leq \frac{\sqrt{n}M}{m}c^k\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_\infty^2$$

Proof. In the following we will use the notation $\mathbf{x}^+$ to denote the next value of $\mathbf{x}$ found by the algorithm. Replacing $\mathbf{y}$ with $\left(\mathbf{x} - t\frac{\partial f(\mathbf{x})}{\partial x_i}\,\mathbf{e_i}\right)$ in equation (79) and writing $\tilde{f}(t,i)$ for $f(\mathbf{x}+t\Delta\mathbf{x}(i))$ we obtain

$$\tilde{f}(t,i) \leq f(\mathbf{x}) - t\,\frac{\partial f(\mathbf{x})}{\partial x_i}\nabla f(\mathbf{x})^\top \mathbf{e_i} + \frac{Mt^2}{2}\left(\frac{\partial f(\mathbf{x})}{\partial x_i}\right)^2$$

$$\tilde{f}(t,i) \leq f(\mathbf{x}) - t\,\|\nabla f(\mathbf{x})\|_\infty^2 + \frac{Mt^2}{2}\|\nabla f(\mathbf{x})\|_\infty^2 \tag{86}$$

The left side is lower bounded by $\mathbf{x}^+ = \tilde{f}(t_{exact}, i)$; minimizing over t the right side of inequality (86) we find $t = 1/M$ and so

$$f(\mathbf{x}^+) \leq f(\mathbf{x}) - \frac{1}{2M}\|\nabla f(\mathbf{x})\|_\infty^2 \tag{87}$$

Subtracting $f(\mathbf{x}^*)$ from both sides, and combining with (77), we get

$$f(\mathbf{x}^+) - f(\mathbf{x}^*) \leq \left(1 - \frac{m}{M}\frac{\|\nabla f(\mathbf{x})\|_\infty^2}{\|\nabla f(\mathbf{x})\|_2^2}\right)(f(\mathbf{x}) - f(\mathbf{x}^*)) \tag{88}$$

But the norms $\|.\|_2$ and $\|.\|_\infty$ are equivalent and in particular $\forall \mathbf{z} \in \mathbb{R}^n$

$$\|\mathbf{z}\|_2 \geq \|\mathbf{z}\|_\infty \geq \frac{1}{\sqrt{n}}\|\mathbf{z}\|_2 \tag{89}$$

so finally using the right part of previous inequality,

$$f(\mathbf{x}^+) - f(\mathbf{x}^*) \leq \left(1 - \frac{m}{n\,M}\right)(f(\mathbf{x}) - f(\mathbf{x}^*)) \tag{90}$$

applying this inequality recursively we find that

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq c^k\left(f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)\right) \tag{91}$$

where $c = 1 - \frac{m}{n\,M}$. Besides, with inequality (82) and left part of inequality (89),

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_\infty^2 \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_2^2 \leq \frac{M}{m}c^k\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2 \leq \frac{\sqrt{n}M}{m}c^k\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_\infty^2 \tag{92}$$

□

D.8 Proof of Gradient for $(\mathbf{P3})$

Problem $(\mathbf{P3})$.

Let $\bar{y} = \frac{1}{n-l}\sum_{i=1}^{l} y_i$ and $\bar{z} = \frac{1}{n-l}\sum_{i=1}^{n-l} z_i$. Let $d_i^{(K)}$ when $i > l$ denote the number of existing edges between the unknown vertex i and known vertices $j \in \{1,\ldots,l\}$. Finally, let $\delta(K,U) = \sum_{i=l+1}^{n} d_i^{(K)} = |E(K,U)|$ be the total number of edges between known vertices and unknown vertices. We assume that before performing the update, $\mathbf{z}$ is such that $\bar{z} = 0$ (that is, $\mathbf{g} \in \mathcal{H}(G)$, e.g. we start with $\mathbf{g} = \mathbf{s}$). The k^{th} coordinate of the gradient of the energy function for the equivalent unconstrained problem $(\mathbf{P3}')$ is $(\nabla F_H)_k = 2\,(\mathbf{Q}^\top \mathbf{L}\,(\mathbf{s} + \mathbf{Q}\mathbf{z}))_k$. Setting this expression to zero, and making z_k^+ (the new value of z_k) appear, we obtain

$$0 = \left(d_{k+l} - \frac{2d_{k+l}^{(K)}}{n-l} + \frac{\delta(U,K)}{(n-l)^2}\right)(\mathbf{z}_k^+ - \bar{y}) - \sum_{(k+l,j)\in E(U,K)} \mathbf{y}_j - \sum_{(k+l,j)\in E(G^{(U)})} (\mathbf{z}_{j-l} - \bar{y})$$
$$+\frac{1}{n-l}\sum_{(i,j)\in E(K,U)} (\mathbf{y}_i - \mathbf{z}_{j-l} + \bar{y}) + \frac{1}{n-l}\left(2d_{k+l}^{(K)} - \frac{\delta(K,U)}{n-l}\right)(\mathbf{z}_k - \bar{y}) \tag{93}$$

The update rule then becomes

$$\mathbf{z}_k^+ = \left[\sum_{(k+l,j)\in E(U,K)} \mathbf{y}_j + \sum_{(k+l,j)\in E(G^{(U)})} (\mathbf{z}_{j-l} - \bar{y}) - \frac{1}{n-l}\sum_{(i,j)\in E(K,U)} (\mathbf{y}_i - \mathbf{z}_{j-l} + \bar{y})\right.$$
$$\left. -\frac{1}{n-l}\left(2d_{k+l}^{(K)} - \frac{\delta(K,U)}{n-l}\right)(\mathbf{z}_k - \bar{y})\right]\left(d_{k+l} - \frac{2d_{k+l}^{(K)}}{n-l} + \frac{\delta(U,K)}{(n-l)^2}\right)^{-1} + \bar{y} \tag{94}$$

Now we rewrite this using the g_is. Remember that $g_i = y_i$ if $i \leq l$, $g_i^+ = z_{i-l}^+ - \bar{z} - \bar{y} = z_{i-l}^+(\frac{n-l-1}{n-l}) - \frac{z_{i-l}}{n-l} - \bar{y}$ for the node to update, and $g_j = z_{i-l} - \bar{z} - \bar{y} = z_{i-l} - \bar{y}$ otherwise. We also use

$$\Phi_i = 2d_i^{(K)} - \frac{\delta(U,K)}{n-l} \tag{95}$$

Φ_i is the difference between two times the number of edges between the unknown vertex i and known vertices, and the average of this number across all unknown vertices. We finally obtain the following two-steps update rule:

- update g_i with

$$g_i^+ = \frac{\sum_{(i,j)\in E(G)} g_j - \frac{\Phi_i g_i}{n-l} - \frac{1}{n-l}\sum_{(p,q)\in E(K,U)}(g_p - g_q)}{d_i - \frac{\Phi_i}{n-l}} \tag{96}$$

- update all the unknown vertices (including g_i) with $\forall j \in (l+1,\ldots,n)$

$$g_j^+ = g_j - \frac{1}{n-l}\sum_{k=1}^{n} g_k \tag{97}$$

The first part of this update rule is quite close to what we have found in previous problems, except that here we take the average of the neighbors AND the vertex i itself – with a coefficient depending on its connectivity. A constant term is also present ; it depends on the continuity of the labels at the border between known vertices and unknown vertices. It is interesting to note that for a large number $(n-l)$ of unknown vertices we tend to the same update rule than for problem $(\mathbf{P1})$.