Edge-enhancing filters with negative weights

Knyazev, A.

TR2015-142  December 2015

Abstract
In [doi:10.1109/ICMEW.2014.6890711], a graph-based denoising is performed by projecting the noisy image to a lower dimensional Krylov subspace of the graph Laplacian, constructed using non-negative weights determined by distances between image data corresponding to image pixels. We extend the construction of the graph Laplacian to the case, where some graph weights can be negative. Removing the positivity constraint provides a more accurate inference of a graph model behind the data, and thus can improve quality of filters for graph-based signal processing, e.g., denoising, compared to the standard construction, without affecting the computational costs.

2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)
Abstract—In [doi:10.1109/ICMEW.2014.6890711], a graph-based denoising is performed by projecting the noisy image to
a lower dimensional Krylov subspace of the graph Laplacian, constructed using non-negative weights determined by distances
between image data corresponding to image pixels. We extend the construction of the graph Laplacian to the case, where some
graph weights can be negative. Removing the positivity constraint
provides a more accurate inference of a graph model behind the data, and thus can improve quality of filters for graph-based
signal processing, e.g., denoising, compared to the standard construction, without affecting the computational costs.

I. INTRODUCTION

Constructing efficient signal filters is a fundamental problem
in signal processing with a vast literature; see, e.g., recent
papers [1], [2], [3], [4], [5], [6] and references there. A filter
can be described by a transformation $F$, often non-linear, of
an input signal, represented by a vector $x$, into a filtered
signal, represented by a vector $F(x)$. We revisit some classical
constructions of filters aimed at signal noise reduction, with
the emphasis on bilateral filter, popular in image denoising
[7], [8], [9], [10]. Reducing a high oscillatory additive noise,
the goal of the filter is, on the one hand, signal smoothing.
The smoothing can be achieved by averaging, which can
typically be interpreted as a low-pass filter, minimizing the
contribution of the filtered signal of highly oscillatory modes,
treated as eigenvectors of a graph Laplacian; see, e.g., [11].

On the other hand, it is desirable to preserve edges in the
ideal noise-free signal, even at the costs of an increased PSNR,
especially in imaging. Edge-conscious filters detect, often
implicitly, the locations of the edges and attempt using less
aggressive or anisotropic averaging at these locations. Fully
automatic edge detection in a noisy signal is difficult, typically
resulting in non-linear filters, i.e. where the filtered vector
$F(x)$ depends non-linearly on the input vector $x$. However,
it can be assisted by a guiding signal, having the edges in the
same locations as in the ideal signal; see, e.g., [3], [12], [13].

Graph signal processing, introducing eigenvectors of the
graph Laplacian as natural extensions of the Fourier bases,
sheds new light at image processing; see, e.g., [14], [15],
[16], [17]. In [18], graph-based filtering of noisy images is
performed by directly computing a projection of the image to
be filtered onto a lower dimensional Krylov subspace of the
normalized graph Laplacian, constructed using non-negative
graph weights determined by distances between image data
corresponding to image pixels. We extend the construction of
the graph Laplacian to the case, where some weights can be
negative, radically departing from the traditional assumption.

II. PRELIMINARIES

Let us for simplicity first assume that the guiding signal,
denoted by $y$, is available and can be used to reliably detect
the locations of the edges and, most importantly, to determine
the edge-conscious linear transformation (matrix) $F_y$ such that
the action of the filter $F(x)$ is given by the following matrix-vector product $F_y x = F(x)$. Having a specific construction
of the guided filter matrix $F_y$ as a function of $y$, one can define a
self-guided non-linear filter, e.g., as $F_\delta x$, which can be applied
iteratively, starting with the input signal vector $x_0$ as follows,
$x_{i+1} = F(x_i), \ i = 0, 1, \ldots , m; \text{ cf., e.g., [19]}.$

Similarly, an iterative application of the linear guided filter
can be used, mathematically equivalent to applying the powers
of the square matrix $F_y$, i.e. $x_m = (F_y)^m x_0$, thus naturally
called the power method, which is an iterative form of kernel
PCA; see, e.g., [20], [21]. To avoid a re-normalization of the
filtered signal, it is convenient to construct the matrix $F_y$
in the form $F_y = D_y^{-1} W_y$, where entries of the square matrix
$W_y$ are called weights. The matrix $D_y$ is diagonal, made of
row-sums of the matrix $W_y$, which are assumed to be non-zero.
Thus, $D_y^{-1} W_y$ multiplied by a column-vector of ones,
gives again the column-vector of ones.

Let us further assume that the matrix $W_y$ is symmetric and
that all the entries (weights) in $W_y$ are non-negative. For the
purpose of the signal denoising, the following observations
are the most important. The right eigenvector $v_1$ of the matrix
$D_y^{-1} W_y$ with the eigenvalue $\mu_1 = 1$ is trivial, just made of
ones, only affecting the signal offset. Since the iterative matrix
$F_y = D_y^{-1} W_y$ is diagonalizable, the power method gives

$$x_m = (F_y)^m x_0 = \sum_j \mu_j^m (v_j^T D x_0) v_j,$$

(1)

where $1 = |\mu_1| \geq |\mu_2| \geq \ldots$ are the eigenvalues of the
matrix $D_y^{-1} W_y$ corresponding to the eigenvectors $v_j$
scaled such that $v_j^T D v_j = \delta_{j1}$. The power method, according to (1), suppresses contributions of the eigenvectors corresponding to
the smallest eigenvalues. Thus, the matrix $W_y$ needs to be
constructed in such a way that these eigenvectors represent
the noisy part of the input signal, while the other eigenvectors
are edge-conscious; cf. anisotropic diffusion [22], [23], [24].

Let us introduce the guiding Laplacian $L_y = D_y - W_y$
and normalized Laplacian $D_y^{-1} L_y = I - D_y^{-1} W_y$ matrices.
In [18], the power method (1) is replaced with a projection of
the image vector $x$ to be denoised onto a lower dimensional
Krylov subspace of the guiding normalized graph Laplacian
$D_y^{-1} L_y$ and implemented, e.g., using the Conjugate Gradient
(CG) method; see, e.g., [25], [26], [27].
III. MOTIVATION

Taking aside algorithmic issues and related computational costs, the ultimate quality of denoising is first of all determined by the choice of the weights. One of the most popular edge-preserving denoising filters is the bilateral filter (BF), see, e.g., [28], [29] and references there, which takes the weighted average of the nearby pixels. The weights $w_{ij}$ may depend on spatial distances and signal data similarity, e.g.,

$$w_{ij} = \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_d^2}\right) \exp\left(-\frac{\|y[i] - y[j]\|^2}{2\sigma_r^2}\right),$$ (2)

where $p_i$ denotes the position of the pixel $i$, the value $y[i]$ is the signal intensity, and $\sigma_d$ and $\sigma_r$ are filter parameters.

To simplify the presentation and our arguments, we further assume that the signal is scalar on a one-dimensional uniform grid, setting without loss of generality the first multiplier in (2) to be 1, and that the weights $w_{ij}$ are computed only for the nearest neighbors and set to zero otherwise.

Let us start with a constant signal, where $y[i] = y[j] = 0$. Then, $w_{i-1} = w_{i} = w_{i+1} = 1$ and the graph Laplacian $L_y = D_y - W_y$ is a tridiagonal matrix that has nonzero entries 1 and −1 in the first row, −1 and 1 in the last row, and $[−1 \ 2 \ −1]$ in every other row. This graph Laplacian $L_y$ is a standard three-point-stencil finite-difference approximation of the negative second point-wise derivative of functions with homogeneous Neumann boundary conditions, i.e., vanishing first derivatives at the end points of the interval. Its eigenvectors are the basis vectors of the discrete cosine transform; see the first five low frequency eigenmodes (the eigenvectors corresponding to the smallest eigenvalues) of $L_y$ in Figure 1. As can be seen in Figure 1, all smooth low frequency eigenmodes turn flat at the end points of the interval, due to the Neumann conditions.

The key observation is that the Laplacian row sums in the first and last rows vanish for any signal, according to the standard construction of the graph Laplacian, no matter what formulas for the weights are being used! Thus, any low pass filter based on low frequency eigenmodes of the graph Laplacian flattens the signal at the end points.

Let us now use formula (2) for a piece-wise constant guiding signal $y$ with the jump large enough to result in a small value $w_{i+1} = w_{i+1}$ for some index $i$. The first five vectors of the corresponding Laplacian are shown in Figure 2. All the plotted in Figure 2 vectors are aware of the jump, representing an edge in our one-dimensional signal $y$, but they are also all flat on both sides of the edge! Such a flatness is expected to appear for any guiding signal $y$ giving a small value $w_{i+1} = w_{i+1}$.

The presence of the flatness in the low frequency modes of the graph Laplacian $L_y$ on both sides of the edge in the guiding signal $y$ is easy to explain. When the value $w_{i+1} = w_{i+1}$ is small relative to other entries, the matrix $L_y$ becomes nearly block diagonal, with two blocks, which approximate graph Laplacian matrices of the signal $y$ restricted to sub-intervals of the signal domain to the left and to the right of the edge.

The low frequency eigenmodes of the graph Laplacian $L_y$ approximate combinations of the low frequency eigenmodes of the graph Laplacians on the sub-intervals. But each of the low frequency eigenmodes of the graph Laplacian on the sub-interval suffers from the flattening effect on both ends of the sub-interval, as explained above. Combined, it results in the flatness in the low frequency modes of the graph Laplacian $L_y$ on both sides of the edge. For denoising, the flatness of the vectors determining the low-pass filter may have a negative effect for self-guided denoising even of piece-wise constant signals, if the noise is large enough relative to the jump in the signal, as we demonstrate numerically in Section V.

The attentive reader could notice that the power method (1) is based on $D_y^{-1}W_y$, related to the normalized graph Laplacian $D_y^{-1}L_y$, not the Laplacian $L_y$ used in our arguments above. Although the diagonal matrix $D_y$ is not a scalar identity, and so the eigenvectors of $D_y^{-1}L_y$, not plotted here, and of $L_y$ are different, the difference is not qualitative enough to noticeably change the figures and invalidate our explanation.
IV. NEGATIVE WEIGHTS IN SPECTRAL GRAPH PARTITIONING AND FOR SIGNAL EDGE ENHANCING

The low frequency eigenmodes of the graph Laplacian play a fundamental role in spectral graph partitioning, which is one of the most popular tools for data clustering; see, e.g., [30], [31], [32]. A limitation of the conventional spectral clustering approach is embedded in its definition based on the weights of graph, which must be nonnegative, e.g., based on a distance measuring relative similarities of each pair of points in the dataset. For the dataset representing values of a signal, e.g., pixel values of an image, formula (2) is a typical example of determining the nonnegative weights, leading to the graph adjacency matrix $W$ with nonnegative entries, as assumed in Section II and in all existing literature.

In many practical problems, data points represent feature vectors or functions, allowing the use of correlation for their pairwise comparison. However, the correlation can be negative, or, more generally, points in the dataset can be dissimilar, contrasting each other. In the conventional spectral clustering, the only available possibility to handle such a case is to replace the anticorrelation, i.e. negative correlation, of the data points with the uncorrelation, i.e. zero correlation. The replacement changes the corresponding negative entry in the graph adjacency matrix to zero, to enable the conventional spectral clustering to proceed, but nullifies a valid comparison.

A common motivation of spectral clustering comes from analyzing a mechanical vibration model in a spring-mass system, where the masses that are tightly connected have a tendency to move synchronically in low-frequency free vibrations; e.g., [33]. Analyzing the signs of the components corresponding to different masses of the low-frequency vibration modes of the system allows one to determine the clusters. The mechanical vibration model may describe conventional clustering when all the springs are pre-tensed to create an attracting force between the masses. However, one can also pre-tense some of the springs to create repulsive forces!

In the context of data clustering formulated as graph partitioning, that corresponds to negative entries in the adjacency matrix. The negative entries in the adjacency matrix are not allowed in conventional graph spectral clustering. Nevertheless, the model of mechanical vibrations of the spring-mass system with repulsive springs remains valid, motivating us to consider the effects of having negative graph weights.

In the spring-mass system, the masses, which are attracted, have the tendency to move together synchronically in the same direction in low-frequency free vibrations, while the masses, which are repulsed, have the tendency to move synchronically in the opposite direction. Using negative, rather than zero, weights at the edge of the guiding signal $y$ for the purposes of the low-pass filters thus is expected to repulse the flatness of low frequency eigenmodes of the graph Laplacian $L_y$ on the opposite sides of the edge of the signal $y$, making the low frequency eigenmodes to be edge-enhancing, rather than just edge-preserving; cf. [34] on sharpening.

Figures 3 and 4 demonstrate the effect of edge-enhancing, as a proof of concept. Both Figures 3 and 4 display the five eigenvectors for the five smallest eigenvalues of the same tridiagonal graph Laplacian as that corresponding to Figure 2 except that the small positive entry of the weights $w_{i,i+1} = w_{i+1,i}$ for the same $i$ is substituted by $-0.05$ in Figure 3 and by $-0.2$ in Figure 4. The previously flat around the edge eigenmodes in Figure 2 are repelled in opposite directions on the opposite sides of the edge in Figures 3 and 4.

Negative weights require caution, since even small changes dramatically alter the behaviors of the low frequency eigenmodes around the edge, as seen in Figures 3 and 4. Making the negative value more negative, we observe by comparing Figure 3 to Figure 4 that the leading eigenmode, displayed using the blue color in both figures, corresponding to the smallest nonzero eigenvalue (which can turn negative!) forms a narrowing layer around the signal edge, while other eigenmodes become less affected by the change in the negative value.
V. EDGE-ENHANCING FILTERS

In this section, as a proof of concept, we numerically test the proposed edge-enhancing filters on a toy one-dimensional example using the classical nonlinear self-guiding BF and a guided (by a noiseless signal) BF accelerated with a conjugate gradient (CG-BF) method, as suggested in [18]. The specific CG algorithm used in our tests is as described in Algorithm 1.

Algorithm 1: Conjugate Gradient Guided Filter

```
1 Input: signal vector to be filtered \(x_0\), matrices \(D_y\) and \(L_y\)
2 \(r_0 = -L_yx_0\)
3 for \(k = 0, 1, \ldots, m - 1\) do
4     \(s_k = D_y^{-1}r_k\)
5     if \(k = 0\) then
6         \(p_0 = s_0\)
7     else
8         \(p_k = s_k + \beta_k p_{k-1}\), where
9         \(\beta_k = \frac{(s_k, r_k)}{(s_{k-1}, r_{k-1})}\)
10     end
11     \(q_k = L_y p_k\)
12     \(\alpha_k = \frac{(s_k, q_k)}{(p_k, q_k)}\)
13     \(x_{k+1} = x_k + \alpha_k p_k\)
14     \(r_{k+1} = r_k - \alpha_k q_k\)
15 end
16 Output: filtered vector \(x_m\)
```

The noise is additive Gaussian, and the noisy signal is displayed using grey dots. The nonzero weights are computed by (2) with \(\sigma_2 = 0.5\) and \(\sigma_1 = 0.1\) only for \(j = i-1, i, i+1\), resulting in tridiagonal matrices \(W\) and \(L\). BF is self-guided, with \(W\) and \(L\) recomputed on every iteration using the current approximation \(x_k\) to the final filtered signal \(x_m\). CG-BF uses the fixed nonzero weights computed also by (2), but for the noiseless signal \(y\) resulting in the fixed tridiagonal matrices \(W_y\) and \(L_y\). The number of iterations in BF, 100, and CG-BF, 15, is tuned to match the errors. We note that formula (2) puts ones on the main diagonal of \(W\), so for small positive or even negative \(w_{i+1} = w_{i+1}\), the matrix \(D\) is well conditioned.

Figure 5 demonstrates the traditional approach, with all weighs non-negative. We observe, as discussed in Section III, flattening at the end points. Most importantly, there is noticeable edge smoothing in all corners, larger in self-guided BF and smaller in guided CG-BF, due to a large level of noise and relatively small number of signal samples, despite of the use of the edge-preserving formula (2). We set tuned negative graph weights \(-2 \times 10^{-3}, -10^{-3}, -10^{-8}\) for \(i = 100, 250,\) and 350 correspondingly, without changing anything else, to obtain Figure 6, which shows dramatic improvements both in terms of PSNR and edge matching, compared to Figure 5.

VI. CONCLUSION

The proposed novel technology of negative graph weights allows designing edge enhancing filters, as explained theoretically and shown numerically for a simple synthetic example. Our future work concerns testing the concept for image filtering and exploring its advantages in spectral data clustering.