Generalised Blurring Mean-Shift Algorithms for Nonparametric Clustering

Miguel Á. Carreira-Perpiñán
EECS, School of Engineering, University of California, Merced
mcarreira-perpinan@ucmerced.edu

Abstract

Gaussian blurring mean-shift (GBMS) is a nonparametric clustering algorithm, having a single bandwidth parameter that controls the number of clusters. The algorithm iteratively shrinks the data set under the application of a mean-shift update, stops in just a few iterations and yields excellent clusterings. We propose several families of generalised GBMS (GGBMS) algorithms based on explicit, implicit and exponential updates, and depending on a step-size parameter. We give conditions on the step size for the convergence of these algorithms and show that the convergence rate for Gaussian clusters ranges from sublinear to linear, cubic and even higher order depending on the update and step size. We show that the algorithms are related to spectral clustering if using a random-walk matrix with modified eigenvalues and updated after each iteration, and show the relation with methods developed for surface smoothing in the computer graphics literature. Detailed experiments in toy problems and image segmentation show that, while all the GGBMS algorithms can achieve essentially the same result (for appropriate settings of the bandwidth and step size), they significantly differ in runtime, with slightly over-relaxed explicit updates being fastest in practice.

Gaussian mean shift (GMS) and Gaussian blurring mean shift (GBMS) are nonparametric clustering algorithms which have received considerable recent attention in image segmentation [7, 3, 5, 1]. Both are deterministic (no initial conditions to set, unlike e.g. k–means) and have a single user parameter, the scale or bandwidth $\sigma > 0$, that is intuitive (being measured in pixels) and indirectly determines the number of clusters (from many for small σ to 1 for large σ). These algorithms do not assume a prior model for the clusters' shape and are thus particularly suitable for segmentation (often a front-end for other tasks such as tracking). Both GMS and GBMS are based on a kernel density estimate of the dataset $\{\mathbf{x}_n\}_{n=1}^N \subset \mathbb{R}^D$: $p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^N K\big(\|\frac{\mathbf{x}-\mathbf{x}_n}{\sigma}\|^2\big)$, where we will focus on the Gaussian kernel $G(t) = e^{-t/2}$. GMS is a fixed-point algorithm that finds the modes of $p(\mathbf{x})$ by iterating

$$p(n|\mathbf{x}^{(\tau)}) = \frac{\exp\left(-\frac{1}{2}\left\|(\mathbf{x}^{(\tau)}-\mathbf{x}_n)/\sigma\right\|^2\right)}{\sum_{n'=1}^N \exp\left(-\frac{1}{2}\left\|(\mathbf{x}^{(\tau)}-\mathbf{x}_{n'})/\sigma\right\|^2\right)} \tag{1a}$$

$$\mathbf{x}^{(\tau+1)} = \mathbf{f}(\mathbf{x}^{(\tau)}) = \textstyle\sum_{n=1}^N p(n|\mathbf{x}^{(\tau)})\mathbf{x}_n \tag{1b}$$

starting from each data point $\mathbf{x}_n$. All data points converging to the same mode form one cluster. GMS is an EM algorithm [2] that converges from any starting point but it is very slow because its convergence order is linear (and occasionally sublinear). GBMS differs from GMS in a small but crucial detail: after one GMS iteration is carried out in parallel for every data point, the entire dataset is updated. Specifically, we do $\mathbf{x}_n = \mathbf{f}(\mathbf{x}_n)$ for every $n = 1,\dots,N$ and keeping $\mathbf{f}$ fixed. Then, the kernel density estimate (and thus $\mathbf{f}$) is recomputed based on the new dataset, and the process is repeated. This results in a sequence of datasets $\mathbf{X}^{(0)}, \mathbf{X}^{(1)}, \dots$ (where each $\mathbf{X}$ is an $N \times D$ matrix of row vectors) that eventually converges to a dataset $\mathbf{X}^{(\infty)}$ with all points coincident ($\mathbf{x}_1^{(\infty)} = \cdots = \mathbf{x}_N^{(\infty)}$). However, before this happens, a phase occurs where points are locally clustered. Carreira-Perpiñán [1] gave a reliable criterion to detect this situation and stop the algorithm at that time (yielding the clustering), and also gave an accelerated GBMS algorithm that yields the same clustering; fig. 2 in that paper shows an example of the evolution of $\mathbf{X}^{(\tau)}$. In practice, accelerated GBMS works very well, yielding clusters which are very similar to those of GMS but 5–60 times faster.

Writing the GBMS update in matrix form reveals the core operation of the algorithm and suggests ways to define new algorithms. Calling $\mathbf{X}$ and $\tilde{\mathbf{X}}$ (both matrices of $N \times D$) the current and new datasets (at iterations τ and $\tau + 1$), then $\mathbf{W}$ is the $N \times N$ matrix of Gaussian affinities $w_{nm} = \exp\left(-\|\mathbf{x}_n - \mathbf{x}_m\|^2/2\sigma^2\right)$, $\mathbf{D} = \operatorname{diag}\left(\sum_{n=1}^N w_{nm}\right)$ is the degree matrix; and $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ is the $N \times N$ random-walk matrix, well-known in spectral clustering[1]. Then, the GBMS update becomes

[1] The Gaussian random-walk matrix $\mathbf{P}$ is positive definite, stochastic, with $p_{nm} \in (0,1)$ and $\sum_{m=1}^N p_{nm} = 1$. It has a single eigenvalue 1 associated with an eigenvector of ones ($\mathbf{P1} = \mathbf{1}$) and $N-1$ eigenvalues in $(0,1)$.

```
repeat
    $\mathbf{W} = \left(\exp\left(-\frac{1}{2}\left\|(\mathbf{x}_m - \mathbf{x}_n)/\sigma\right\|^2\right)\right)_{nm}$
    $\mathbf{D} = \operatorname{diag}\left(\sum_{n=1}^N w_{nm}\right)$
    $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$
    $\mathbf{X} = \phi(\mathbf{P})\,\mathbf{X}$
until stop
connected-components$\left(\{\mathbf{x}_n\}_{n=1}^N, \texttt{min\_diff}\right)$
```

Figure 1. Generalised Gaussian blurring mean-shift (GGBMS).

$\tilde{\mathbf{X}} = \mathbf{PX}$, which can be seen as a shrinking or smoothing of $\mathbf{X}$ using a filter defined by $\mathbf{P}$. This suggests that we can define new algorithms by replacing $\mathbf{P}$ with a different filter $\phi(\mathbf{P})$, thus defining the update as $\tilde{\mathbf{X}} = \phi(\mathbf{P})\,\mathbf{X}$. The objective of this paper is to study in theory and experiment the properties of such *generalised GBMS (GGBMS)* algorithms, which may guide the choice of ϕ in practice: how different are their clustering results? Do they converge, and with what order? How fast are they overall? We begin (section 1) by introducing several useful types of updates (explicit, implicit, exponential) dependent on a step-size parameter. We then show (section 2) an important property for them: that, for a Gaussian cluster, the updated cluster remains Gaussian but with a smaller variance. This allows us to characterise the conditions and order of convergence for each method. We give a similar analysis in the context of spectral clustering (section 3), showing that each method is equivalent to using a $\mathbf{P}$ matrix with the same eigenvectors but modified eigenvalues. We give the computational cost of each algorithm (section 4) and compare the algorithms with toy and image segmentation problems over a range of step sizes and bandwidths (section 5), and review related work (section 6).

1. Generalised GBMS (GGBMS)

We focus on the GGBMS algorithm of fig. 1. Its accelerated version will be described in section 4. Both are exactly like the algorithm of [1] except that the update there was the GBMS update $\tilde{\mathbf{X}} = \mathbf{PX}$ and here we consider a more general update $\tilde{\mathbf{X}} = \phi(\mathbf{P})\,\mathbf{X}$ where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^T$ is an $N \times D$ matrix. At each iteration, GGBMS builds the $N \times N$ affinity matrix $\mathbf{W} = \left(\exp\left(-\left\|\mathbf{x}_n - \mathbf{x}_m\right\|^2/2\sigma^2\right)\right)_{nm}$ and the degree matrix $\mathbf{D} = \operatorname{diag}\left(\sum_{n=1}^N w_{nm}\right)$, which define the random-walk matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. As in [1] GGBMS is stopped when the entropy of the distribution of distances $\{\|\mathbf{x}_n^{(\tau+1)} - \mathbf{x}_n^{(\tau)}\|\}_{n=1}^N$ does not change between two consecutive iterations. This detects the typical situation where points have clustered locally into K clusters (yielding a histogram with K bins) and then these clusters keep approaching each other over iterations (so the bins change location but not mass). After the iterations stop, the dataset consists of clusters where points are almost but not exactly coincident; a connected-components step (using a small positive threshold `min_diff`) merges all points into their clusters.

We consider the following updates $\tilde{\mathbf{X}} = \phi(\mathbf{P})\,\mathbf{X}$ (we refer to both parameters η, n as the step size):

GBMS: $\phi(\mathbf{P}) = \mathbf{P}$. Also explicit–$\eta, n$ for $\eta, n = 1$.

Explicit–η: $\phi(\mathbf{P}) = (1-\eta)\mathbf{I} + \eta\mathbf{P}$ for $\eta \in (0, 2]$ (overrelaxed: $\eta > 1$, underrelaxed: $\eta < 1$).

Explicit–n: $\phi(\mathbf{P}) = \mathbf{P}^n$ for $n = 1, 2, 3\dots$

Implicit–η: $\phi(\mathbf{P}) = ((1+\eta)\mathbf{I} - \eta\mathbf{P})^{-1}$ for $\eta > 0$.

Rational GBMS: $\phi(\mathbf{P}) = A(\mathbf{P})^{-1}B(\mathbf{P})$ where $A(\mathbf{P}) = \sum_{i=0}^n a_i\mathbf{P}^i$ (with $a_0 \neq 0$) and $B(\mathbf{P}) = \sum_{i=0}^n b_i\mathbf{P}^i$ are polynomials of order n with real coefficients. This generalises the explicit and implicit cases.

Exponential–η: $\phi(\mathbf{P}) = e^{-\eta(\mathbf{I}-\mathbf{P})}$ for $\eta > 0$, where the matrix exponential is defined as $e^{\mathbf{A}} = \sum_{i=0}^\infty \mathbf{A}^i/i!$ if the series converges.

The function $\phi(\mathbf{P})$ can be represented as a matrix polynomial in all cases. In particular, the rational function $\phi(\mathbf{P}) = A(\mathbf{P})^{-1}B(\mathbf{P})$ results from solving the linear system $A(\mathbf{P})\tilde{\mathbf{X}} = B(\mathbf{P})\mathbf{X}$.

2. Analysis of convergence and its order

Consider the GGBMS update $\tilde{\mathbf{X}} = \phi(\mathbf{P}(\mathbf{X}))\,\mathbf{X}$. For Gaussian datasets, the update can be characterised analytically (see proofs in appendix). For non-Gaussian datasets, this model seems to be a good approximation to the later stages of the algorithm, when each cluster of the data set does (in isolation) look Gaussian. Our main results are as follows:

- If $\mathbf{X}$ is Gaussian then $\tilde{\mathbf{X}}$ is Gaussian with the same mean, the update $\tilde{\mathbf{X}} = \mathbf{f}(\mathbf{X}) = \phi(\mathbf{P})\,\mathbf{X}$ is a linear mapping, and the standard deviation maps linearly and separately for each principal component. Specifically, if along one principal component the dataset is $x \sim \mathcal{N}(\mu, s^2)$ for $\mathbf{x} \in \mathbb{R}$, then the new dataset is $\tilde{x} \sim \mathcal{N}(\mu, (\phi(r)s)^2)$, where
$$r = r(s) = \frac{1}{1 + (\sigma/s)^2} \in (0, 1). \qquad (2)$$
This allows a complete characterisation of the conditions and order of convergence in terms of the real function $\phi(r)$, $r \in (0, 1)$, instead of a matrix function.
- The sequence of standard deviations satisfies $s^{(\tau+1)} = |\phi(r(s^{(\tau)}))|\, s^{(\tau)}$ for $s^{(0)} > 0$. We can then determine: (1) conditions of convergence, by requiring $|\phi(r)| < 1$ so that the cluster shrinks; and (2) the order

Method	$\phi(r)$	step size range	convergence order p	ϱ	cost c per iteration
explicit–η	$1-\eta+\eta r$	$\eta \in (0,2] \setminus \{1\}$	1	$1-\eta$	1
GBMS (explicit $\eta, n=1$)	r	$\eta = 1$	3	σ^{-2}	1
explicit ($\eta = 2$)	$-1+2r$	$\eta = 2$	sublinear	-1	1
explicit–n	r^n	$n = 1,2,3\dots$	$2n+1$	σ^{-2n}	n
implicit–η	$\frac{1}{1+\eta-\eta r}$	$\eta \in (0,\infty)$	1	$\frac{1}{1+\eta}$	$\frac{1}{3}\frac{N}{D}$
rational (polynomials A, B)	$B(r)/A(r)$	depends	depends	depends	depends
exponential–η	$e^{-\eta(1-r)}$	$\eta \in (0,\infty)$	1	$e^{-\eta}$	$2\frac{N}{D}$

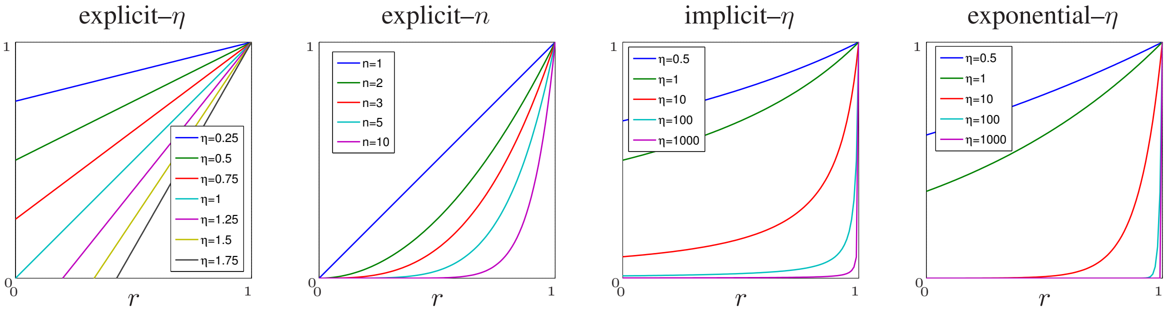

Table 1. Summary of the methods' properties and graph of their defining function $\phi(r)$, $r \in (0,1)$.

p of convergence, which is the largest $p > 0$ for which $\varrho = \lim_{\tau\to\infty} \left|s^{(\tau+1)} - s^{(\infty)}\right| / \left|s^{(\tau)} - s^{(\infty)}\right|^p < \infty$. The results (which we have checked empirically) are given in table 1 for each method and its step size.

Explicit–n offers the higher convergence order ($2n+1$), in particular GBMS converges cubically—extremely fast. Explicit–η (for $\eta \neq 1$) converges linearly; it is fastest for $\eta \approx 1$ (when $\varrho \to 0$) and slowest for η close to 0 or 2 (when $|\varrho| \to 1$), becoming sublinear for $\eta = 2$ (extremely slow). For $\eta \in (1,2]$ (overrelaxation) the method yields an alternating series.

Implicit–η and exponential–η behave quite similarly (as seen from their ϕ-curves in table 1 and in the experiments). They converge linearly but, for large step sizes, ϱ is very small so convergence will be fast. However, we cannot take $\eta \to \infty$ since then they would converge in one iteration to one cluster. In practice, η from 10 to 1 000 is fast, with larger η bringing only a slight improvement. For $0 < \eta \ll 1$, all 3 methods are equivalent, since $\frac{1}{1+\eta-\eta r} \approx e^{-\eta(1-r)} \approx 1-\eta+\eta r$.

The linearly convergent methods preserve the ratio of standard deviations along different directions, but not explicit–n. For the latter, the ratio is powered to $2n+1$ at each iteration, so the principal component quickly dominates all others, as happens for GBMS [1]. This results in strong denoising properties.

Ultimately, the convergence order is only part of the algorithm performance, as (1) each algorithm has a different cost per iteration and (2) part of the computation takes place in the first, highly non-Gaussian iterations. Sections 4 and 5 quantify this.

3. Relation with spectral clustering

In a typical spectral clustering algorithm (e.g. the normalised cut [11]), one builds the matrix $\mathbf{N} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ (equal to $\mathbf{I} - \mathcal{L}$ where $\mathcal{L}$ is the normalised graph Laplacian) and computes the eigenvectors $\mathbf{u}_1, \dots, \mathbf{u}_K$ corresponding to the leading K eigenvalues of $\mathbf{N}$. Then, the data are projected onto these eigenvectors, where their cluster structure is enhanced (but generally not obvious), and a secondary clustering algorithm (e.g. k-means) is used to obtain the clusters. All the GGBMS algorithms have a strong relation with spectral clustering. Recall that, for Gaussian affinities, $\mathbf{N}$ is symmetric positive definite with real eigenvalues $\lambda_1 > \dots > \lambda_N \in (0,1]$ and eigenvectors $\mathbf{u}_n$. Thus, $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}\mathbf{D}^{\frac{1}{2}}$ has the same eigenvalues λ_n and eigenvectors $\mathbf{v}_n = \mathbf{D}^{-\frac{1}{2}}\mathbf{u}_n$. Writing $\mathbf{N} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ in terms of its eigenvectors $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_N)$ and eigenvalues $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_N)$ and using $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}\mathbf{D}^{\frac{1}{2}}$, we see that $\mathbf{P}^n = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}^n\mathbf{D}^{\frac{1}{2}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}\mathbf{\Lambda}^n\mathbf{U}^T\mathbf{D}^{\frac{1}{2}}$ for $n = 0, \pm 1, \pm 2, \dots$ and thus that a polynomial $A(\mathbf{P}) = \sum_{i=0}^n a_i \mathbf{P}^i = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}\left(\sum_{i=0}^n a_i \mathbf{\Lambda}^i\right)\mathbf{U}^T\mathbf{D}^{\frac{1}{2}}$, etc. Therefore the matrix function $\phi(\mathbf{P})$ yields a matrix with the same eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_N$ as $\mathbf{P}$ but eigenvalues $\phi(\lambda)$ for all methods. Naturally, one could compute the eigenvalues and eigenvectors of $\mathbf{P}$ at each iteration and then modify at will the eigenvalues, but this would be very costly for large $\mathbf{P}$.

Consider now the GGBMS update $\tilde{\mathbf{X}} = \phi(\mathbf{P}(\mathbf{X}))\,\mathbf{X}$. If $\mathbf{P}$ were not updated after $\mathbf{X}$ has been updated, then repeated iterations would soon be dominated by the leading eigenvectors of $\phi(\mathbf{P})$: $\phi(\mathbf{P})^n\mathbf{X} \approx \sum_{i=1}^K \phi(\lambda_i)^n \mathbf{v}_i(\mathbf{v}_i^T\mathbf{D}\mathbf{X})$, which are also the leading eigenvectors of $\mathbf{P}$ since ϕ is monotonically increasing (see table 1) and so preserves the order of the eigenvalues. This situation would be very similar to spectral clustering except that the latter discards the eigenvalues and just uses the eigenvectors. The convergence to a dominant space of eigenvectors would be quite slow (determined by the ratio of consecutive eigenvalues), and ultimately convergence would be to the constant eigenvector $\mathbf{v}_1 = \mathbf{1}$ associated with $\lambda_1 = 1$. However, *the key distinguishing aspect of GGBMS algorithms is the fact that the dataset and also the matrix* $\mathbf{P}$ *is updated after each iteration.* The effect of this is that the dataset quickly clusters locally and the matrix $\mathbf{P}$ becomes almost perfectly split in diagonal blocks of 1s, and 0s elsewhere (and after many more iterations these clusters merge). Also, GGBMS does not discard trailing eigenvectors; the product $\phi(\mathbf{P})\mathbf{X}$ uses the information of all eigenvectors and eigenvalues of $\phi(\mathbf{P})$.

Analysing the eigenvalues $\phi(\lambda)$ of $\phi(\mathbf{P})$ is another way to obtain conditions for convergence of GGBMS. For $\phi(\mathbf{P})$ to be a stochastic matrix (so every iteration pushes the dataset towards a one-cluster solution) we must have $\phi(1) = 1$ (leading eigenvalue) and $|\phi(\lambda)| < 1$ for $\lambda \in (0,1)$ (remaining eigenvalues). This yields the ranges in table 1. For example, for the rational $\phi(\mathbf{P}) = A(\mathbf{P})^{-1}B(\mathbf{P})$ this implies the following condition on a_i, b_i:

$$\frac{\sum_{i=0}^{n} b_i}{\sum_{i=0}^{n} a_i} = 1 \text{ and } \left|\frac{\sum_{i=0}^{n} b_i\lambda^i}{\sum_{i=0}^{n} a_i\lambda^i}\right| < 1 \ \forall\lambda \in (0,1). \tag{3}$$

4. Computational cost

We consider as fundamental unit (*normalised iteration*) one product $\mathbf{PX} = \mathbf{D}^{-1}(\mathbf{WX})$ costing N^2D multiplications (strictly $ND + N^2D \approx N^2D$ for $N \gg D$). This operation underlies all methods, as solving a linear system (e.g. by conjugate gradients) can be accomplished by iterating such products. Table 1 shows the cost c of one iteration of unaccelerated GGBMS measured in this unit. The explicit methods are $\mathcal{O}(N^2D)$ while the implicit and exponential methods are $\mathcal{O}(N^3)$. For implicit–η, the dominant cost is solving a linear system $\mathbf{A}\tilde{\mathbf{X}} = \mathbf{X}$ with $\mathbf{A}_{N\times N}$ and $\mathbf{X}_{N\times D}$. Gauss elimination (Matlab: `A\X`) takes $\frac{1}{3}N^3$ multiplications [8, p. 98]. Today's preferred method to compute the matrix exponential is using the Padé approximation with scaling and squaring (Matlab: `expm(P)`), which takes (conservatively) $2N^3$ multiplications [8, p. 574]. We implement explicit–n and $A(\mathbf{P})$ and $B(\mathbf{P})$ in rational GGBMS with Horner's rule for polynomials.

The accelerated technique for GBMS proposed in [1] carries over to GGBMS. The idea is that, as points approach each other over the course of iterations, they form compact local groups (by definition, smaller than a small distance `min_diff` > 0); thus, we can replace one local group containing M points with a single point of mass M. This effectively reduces the total number of points $N^{(\tau)}$ at each iteration (which becomes faster) and yields the same clustering as the unaccelerated algorithm if `min_diff` is small enough. Accelerated GGBMS can be implemented by interleaving the GGBMS update with connected-components steps (of negligible cost).

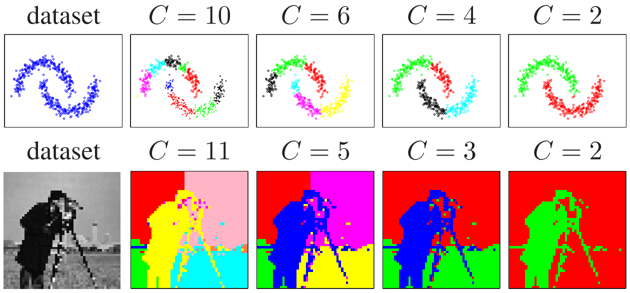

Figure 2. Example datasets and typical results (for all methods, each using a different σ value) for different numbers of clusters C.

Assuming k actual iterations, the speedup unaccelerated/accelerated is $kN^2/\sum_{\tau=0}^{k-1}(N^{(\tau)})^2$ for explicit and $kN^3/\sum_{\tau=0}^{k-1}(N^{(\tau)})^3$ for implicit/exponential methods. If using a sparse graph Laplacian instead, the cost of both versions would reduce proportionally to z/N (for some $z \lll N$) but the speedup would remain the same. In the experiments, for accelerated GGBMS we only report its actual iterations divided by the speedup.

5. Experimental results

We have run all the algorithms in several datasets of different dimensionality, size and type (e.g. toy, images), and over a large range of σ (not just one best value), with essentially equivalent conclusions. We show detailed results for the toy dataset of fig. 2 (which shows both a clustered and low-dimensional structure). In all experiments, we considered 4 types of methods (see table 1): explicit–η with step size $\eta = 0.25,\ 0.5,\ 1$ (GBMS), $1.25,\ 1.5,\ 1.75$; explicit–n with power $n = 1$ (GBMS), $2,\ 3,\ 5,\ 10$; and implicit–η and exponential–η with step size $\eta = 0.5,\ 1,\ 10,\ 1000$. These values explore the ranges of the respective step sizes and complement the theory of section 2. For each case we ran the unaccelerated and accelerated versions of GGBMS and the experiments were performed for 41 values of the bandwidth σ (over a range chosen to yield between 1 and many clusters). Fig. 3 reports (as a function of σ) the number of clusters, number of actual iterations (i.e., not normalised) and actual CPU time. Fig. 4 reports the number of normalised iterations for accelerated GGBMS.

In terms of clustering quality, all algorithms behave similarly, producing the same given clustering for appropriate σ and step size. The number of clusters C decreases with either increasing σ or increasing η (or n), and empirically we find these 3 variables are dependent. Specifically, for

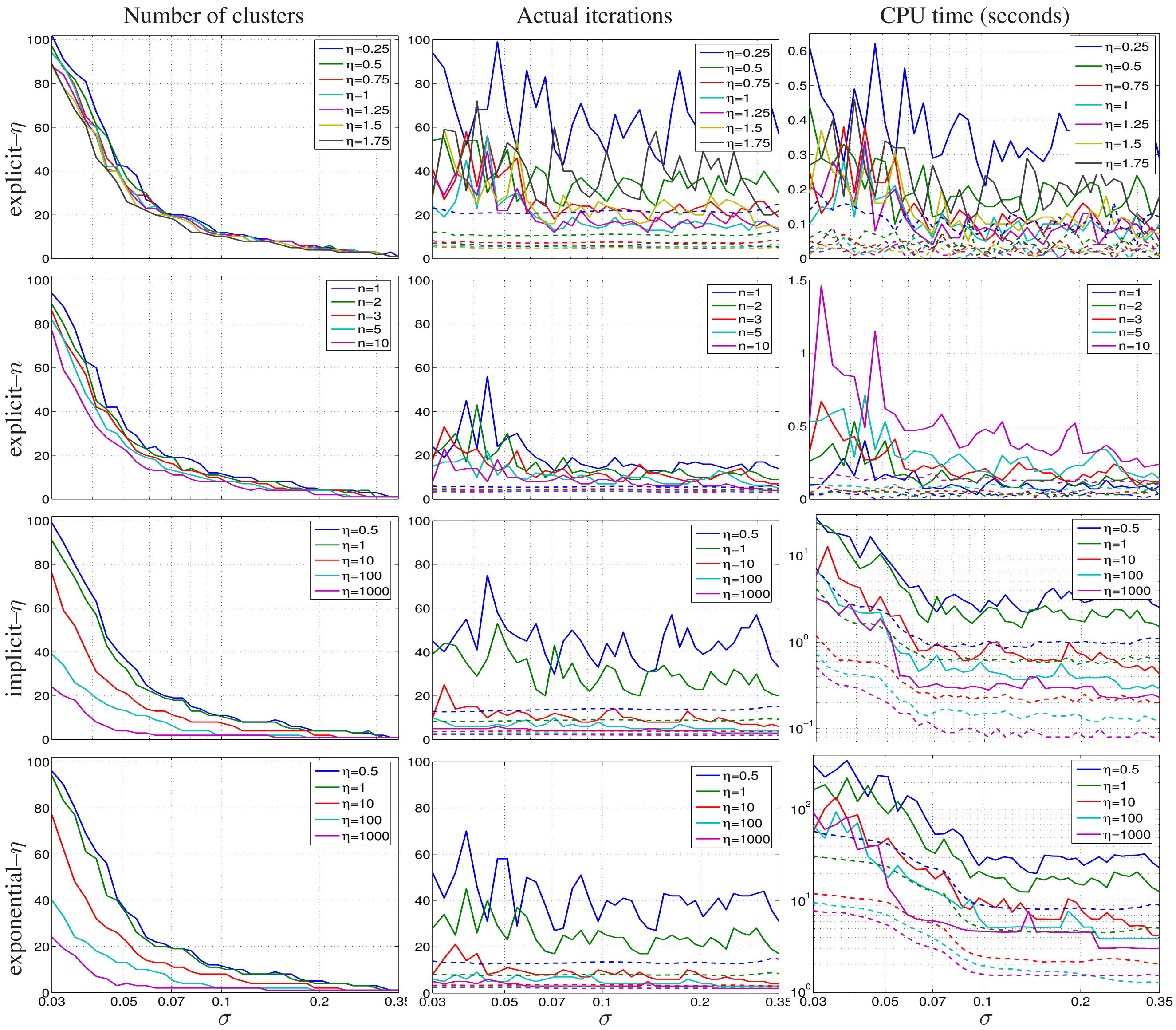

Figure 3. Performance of the algorithms on the toy dataset of fig. 2 ($N = 500$ points in 2D). The X axis (σ) and (for implicit/exponential) the CPU time are on a log scale. The dashed lines are for accelerated GGBMS.

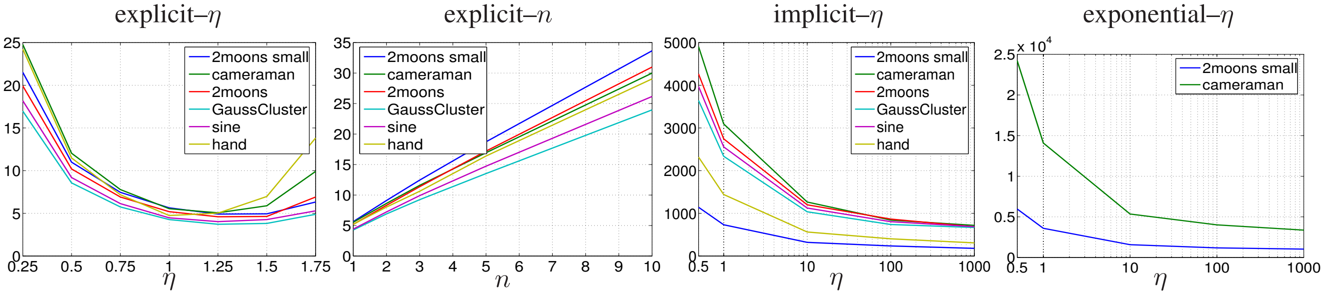

Figure 4. Average (over σ) number of normalised iterations for each accelerated GGBMS algorithm, for several representative datasets. The datasets were chosen to contain clustering and/or low-dimensional manifold structure, as follows (size,dimension): 2moons small: 500, 2D (fig. 2, 2 curved clusters); 2moons: 2000, 2D; sine: 2000, 2D (sinecurve, no clusters); GaussCluster: 2000, 2D (3 Gaussian clusters); cameraman: 2500, 3D (fig. 2, greyscale image); hand: 2000, 5D (colour image).

all datasets we tried, $\log\sigma - \alpha\log\eta$ is a linear function of C with $\alpha \approx \frac{1}{6}$ for the implicit/exponential, $\alpha \approx \frac{1}{12}$ for explicit–n and $\alpha \approx 0$ for explicit–η (i.e., independent of η). This gives a rough guideline to select a good range of σ given a step size; and suggests fixing the step size to the one that runs fastest, and using σ to control the number of clusters (a useful rule in practice).

As predicted by the theory, the lowest numbers of actual iterations occur for the largest η (implicit/exponential) or n (explicit–n), but these methods have a high cost per iteration. As shown by the CPU curves[2], the explicit methods are the fastest: although they take more iterations, these are very cheap (one $\mathbf{PX}$ product).

In practice, the bottomline performance is that of the accelerated GGBMS, whose number of actual iterations (dashed lines in fig. 3) is, unlike that of the unaccelerated GGBMS, remarkably independent of σ. Fig. 4 shows the normalised iterations (averaged over all σ), and thus indicates the optimal step size for each method. Notably, the fastest method of all is explicit–η with $\eta \approx 1.25$ (overrelaxation, taking just 4–5 iterations), even though it has only linear convergence. The curve for explicit–η deviates a bit from the theoretical rate $\varrho = 1 - \eta$ (which is symmetric around its optimum at $\eta = 1$); perhaps datasets lacking symmetry benefit from overshooting the step size (alternating sequence). GBMS ($\eta = 1$) is only slightly slower, because the curve is quite flat for $\eta \in [1, 1.5]$; it is also optimal for explicit–n. For implicit/exponential, the optimum occurs for $\eta \to \infty$, but a step size of 10–1 000 is practically optimal, requiring just 1–2 actual iterations. However, this is still far more costly in runtime than explicit–η.

6. Related work

The development of the (G)BMS algorithm in the machine learning literature has been motivated by a probabilistic modelling point of view: by turning the empirical dataset into a kernel density estimate (by smoothing with a Gaussian kernel), we can derive from its gradient a convenient algorithm to find its modes (mean-shift), and by moving each data point according to it we obtain the GBMS update $\tilde{\mathbf{X}} = \mathbf{PX}$. The GBMS update was introduced by Fukunaga and Hostetler [7], who also noted its potential for clustering and denoising data if the algorithm was stopped when the desired clusters or denoising arose. Cheng [3] showed its convergence to a single cluster. The algorithm in its present form, with an automatic stopping rule and the connected-components acceleration, was introduced by Carreira-Perpiñán [1], who also proved its cubic convergence order for Gaussian clusters and demonstrated its performance in image segmentation. The algorithm has also been applied as a preprocessing step to improve density estimation [4] or classification [9]. Our present paper is the natural continuation of this line of work, by extending the theory and experiments to a more general family of GBMS algorithms for clustering.

A similar type of work (but with significant differences) has developed in the computer graphics literature. While in the machine learning literature the problem driving work on GBMS has been clustering, denoising or preprocessing a possibly high-dimensional dataset in an abstract setting, in computer graphics the problem has been much more concrete: denoising a large dataset of known dimensionality (and possibly known topology: holes, juntions, corners, etc.), specifically denoising (smoothing) a 2D surface living in 3D space. Such a surface is obtained by laser-range techniques, which yield millions of 3D points approximately lying on the surface of an object. The objective is to remove the (small) noise while preserving features such as corners. Common aspects of these techniques are: they assume a sparse mesh (neighbourhood graph) that implicitly represents the surface, often by triangulation; they must be fast (if possible, linear in the number of points) to allow interactive operation with large models; and they use guidance from the user, e.g. to determine the mesh or topology, stop the algorithm and tune parameters. A basic smoothing technique is *Laplacian smoothing*, used to refine irregularly triangulated meshes in finite-element methods [10]. This moves each interior (non-boundary) point in the mesh to the mean of its neighbours: $\tilde{\mathbf{x}}_n = \frac{1}{k}\sum_{m\sim n}\mathbf{x}_m$. It is thus a form of mean-shift with a sparse random-walk matrix $\mathbf{P}$ with constant entries $p_{nm} = \frac{1}{k}$ if $n \sim m$, where k is the number of neighbours of $\mathbf{x}_n$. Taubin [12, 13] extended this idea by approaching the problem from a signal processing point of view, where different filters $\phi(\mathbf{P})$ could be used to obtain different denoising results; in particular, he proposed the use of the $\lambda|\mu$ algorithm (an explicit method), as providing a good bandpass filter. He also suggested more general filters: explicit (IIR filter), and implicit and rational (FIR filter). Desbrun et al. [6] noted that the $\lambda|\mu$ algorithm takes many iterations to achieve good denoising and instead proposed an implicit filter implemented with preconditioned conjugate gradients, and claimed it takes 60% the CPU time. It is important to note the following differences with our work. (1) The nature of the problems (clustering in our case, surface denoising in computer graphics) is different and so is the behaviour of the algorithms in terms of quality of results and CPU time. (2) Crucially, the computer graphics literature seems to have focused on algorithms that update $\mathbf{X}$ but not the matrix $\mathbf{P}$ (or graph Laplacian) after each itera-

[2] The CPU times correlate very well with our normalised iterations (table 1, cost c) except in two aspects for the implicit/exponential methods. (1) Exponential–η is considerable slower than implicit–η than our prediction suggests (other than that, both methods behave almost identically). (2) For both implicit/exponential the CPU time varies with σ for a fixed problem size (N, D): linear systems for larger σ seem to be solved by Matlab 10× as fast as for small σ (note the elbow in the CPU curves, without a correlate in the iterations curves).

tion. This is partly to reduce computation, but also because in the popular Laplacian smoothing $\mathbf{P}$ is constant.[3] This means that, effectively, they are running power iterations of $\mathbf{P}$ with eigenvalues modified by ϕ (see section 3), which defeats the advantage of GGBMS algorithms: that the updates of $\mathbf{X}$ and $\mathbf{P}$ feedback each other resulting in faster convergence. Also, the computer graphics work seems not to have used affinities based on a scale parameter (such as Gaussian with bandwidth σ), which is necessary to control the number of clusters. (3) The algorithms are stopped by hand or trial and error when the user observes sufficient smoothing (but not so much that the surface shrinks).

Finally, the GGBMS algorithms can also be seen from the perspective of the diffusion equation [6, 9]. Consider a continuous low-dimensional manifold whose mass diffuses across the high-dimensional space that contains it: $\frac{\partial \mathbf{X}}{\partial t} = \nabla^2 \mathbf{X}$ where ∇^2 is the Laplace-Beltrami operator. The problem of interest in denoising is the inverse case, where the diffused data return to the manifold. Considering then $\frac{\partial \mathbf{X}}{\partial t} = -\nabla^2 \mathbf{X}$ and discretising the equation we obtain $\frac{\partial \mathbf{X}}{\partial t} \approx \frac{\mathbf{X}_{t+1} - \mathbf{X}_t}{\eta}$ for the temporal derivative (forward difference with step size η) and $\nabla^2 \mathbf{X} \approx -(\mathbf{I} - \mathbf{P})\mathbf{X}_t$ or $\nabla^2 \mathbf{X} \approx -(\mathbf{I} - \mathbf{P})\mathbf{X}_{t+1}$ for the spatial derivative (graph Laplacian defined as $\mathbf{I} - \mathbf{P}$), which yield respectively the explicit and implicit methods. This suggests another way to define generalised algorithms, by using different finite differences and graph Laplacians.

7. Conclusion

We have introduced generalised versions of the GBMS algorithm for nonparametric clustering by defining explicit, implicit, rational and exponential updates controlled by a step-size parameter. Our theoretical analysis shows that the algorithms turn Gaussian clusters into Gaussian clusters with a smaller variance, eventually compressing them to a point, and we have given conditions for convergence on the step size for each method, as well as determined the order of convergence (which ranges from sublinear to linear, cubic and even higher order depending on the method and step size). From the spectral clustering point of view, these algorithms alternate between updating the dataset with one power iteration by the random-walk matrix $\mathbf{P}$ (with modified eigenvalues), and updating $\mathbf{P}$ itself. Experimental results show that all methods are practically equivalent in terms of quality of clusters, i.e., over a range of bandwidths they can obtain the same clustering (at possibly different σ values). However, they markedly differ in computational cost, with the explicit update of step size $\eta \approx 1.25$ being fastest (even though its convergence is linear), closely followed by the original GBMS ($\eta = 1$, cubic convergence). We have also described the relation with algorithms used in computer graphics for denoising (smoothing) surfaces. We hope that future work in machine learning and computer graphics will continue to exploit these connections.

While we have explored an important subset of generalised GBMS algorithms, future work should be directed to active design of updates $\phi(\mathbf{P})$ that are optimal (in some sense) for clustering. The present paper has also focused on Gaussian affinities using a full graph. We are exploring the use of sparse graphs and efficient linear solvers, which may offer great computational savings for image segmentation.

Acknowledgements: this work was supported by NSF CAREER award IIS–0754089.

A. Convergence results for the Gaussian case

We focus on the case where the dataset is Gaussian. This can be solved in closed form for all methods, which allows a complete characterisation of the conditions and order of convergence in terms of the real function ϕ. The analysis of the non-Gaussian case is difficult; in our practical experience, the Gaussian model is a good approximation to the later stages of the algorithm.

Theorem A.1. *Let $\mathbf{X}$ be a Gaussian random variable in D dimensions (the dataset) and consider the GGBMS update $\tilde{\mathbf{X}} = \mathbf{f}(\mathbf{X}) = \phi(\mathbf{P})\,\mathbf{X}$, where ϕ is a real function in $[0,1]$ with $\phi(1) = 1$, and $\mathbf{P} = p(\mathbf{y}|\mathbf{x})$ is the random-walk (continuous) matrix. Then the random variable $\tilde{\mathbf{X}}$ is Gaussian with the same mean as $\mathbf{X}$, the update is a linear mapping, and the standard deviation maps linearly and separately for each dimension of $\mathbf{X}$.*

Proof. We build on the proof approach of [1] for GBMS, which carries over directly to explicit–η but needs modification for implicit–η. To avoid dependence on a given finite dataset, we consider a continuous dataset with a Gaussian distribution, so the update replaces sums over data points with integrals. For example, the following two equations become eqs. (8) and (11) below, respectively:

$$\text{explicit–}\eta\text{: } \tilde{x}_m = (1-\eta)x_m + \eta \textstyle\sum_{n=1}^N p(n|x_m)x_n \qquad (4)$$

$$\text{implicit–}\eta\text{: } (1+\eta)\tilde{x}_m - \eta \textstyle\sum_{n=1}^N p(n|x_m)\tilde{x}_n = x_m. \qquad (5)$$

To simplify the analysis, let us first see that (1) the update is covariant to translation and rotation, and (2) the dimensions separate. To see that the GGBMS update covaries under rotation and translation (this is true for any dataset, finite or infinite, and not just Gaussian), let $\mathbf{R}_{D\times D}$ be an orthogonal matrix and $\mathbf{t} \in \mathbb{R}^D$ a vector, and call $\mathbf{X}' = \mathbf{X}\mathbf{R}^T + \mathbf{1}\mathbf{t}^T$. Then

$$\tilde{\mathbf{X}}' = \phi(\mathbf{P}(\mathbf{X}'))\,\mathbf{X}' \overset{\text{(a)}}{=} \phi(\mathbf{P}(\mathbf{X}))\,(\mathbf{X}\mathbf{R}^T + \mathbf{1}\mathbf{t}^T) \qquad (6)$$

$$= \phi(\mathbf{P}(\mathbf{X}))\,\mathbf{X}\mathbf{R}^T + \phi(\mathbf{P}(\mathbf{X}))\,\mathbf{1}\mathbf{t}^T \overset{\text{(b)}}{=} \tilde{\mathbf{X}}\mathbf{R}^T + \mathbf{1}\mathbf{t}^T \qquad (7)$$

since (a) $\mathbf{P}(\mathbf{X}') = \mathbf{P}(\mathbf{X})$ (because $\mathbf{P}$ depends only on the Euclidean distances $\|\mathbf{x}_n - \mathbf{x}_m\|$, which are invariant to rotation and translation), and (b) $\phi(\mathbf{P})$ is constructed so it yields a stochastic matrix ($\phi(1) = 1$ as a scalar function). This means we can always choose $\mathbf{X}$ to be zero mean and with diagonal covariance (by translation and rotation). To see that the dimensions separate, note

[3] Distance-based $\mathbf{P}$ have been used, e.g. proportional to $\|\mathbf{x}_n - \mathbf{x}_m\|^2$, but even in this case either $\mathbf{P}$ is not updated, or if it is then the neighbourhood structure is not updated, so $\mathbf{P}$ has the same zeros.

(see below) that in the infinite-sample case each column (dimension) of $\tilde{\mathbf{X}} = \mathbf{PX}$ or in general $\mathbf{P}\,\mathbf{f}(\mathbf{y})$ (where $\mathbf{y}$ is a column of $\mathbf{X}$) equals the expectation of $\mathbf{f}(\mathbf{y})$ under $p(\mathbf{y}|\mathbf{x}) \propto G(\mathbf{x}-\mathbf{y})q(\mathbf{y})$, which is factorised since G (the isotropic, or even diagonal, kernel of the kernel density estimate) and q (the dataset distribution) are factorised. The same holds for polynomials ϕ, as these can be written as repeated expectations (e.g. $\mathbf{P}^2 = \mathrm{E}\left\{\mathrm{E}\left\{\cdot\right\}\right\}$).

Hence, w.l.o.g. we consider the zero-mean 1D case. Let the data distribution be $q(x) = \mathcal{N}(x; 0, s^2)$ (Gaussian with zero mean and variance s^2) and the Gaussian kernel $G(x) = \mathcal{N}(x; 0, \sigma^2)$, so the kernel density estimate $p(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^N K\big(\|\frac{\mathbf{x}-\mathbf{x}_n}{\sigma}\|^2\big)$ is, as $N \to \infty$, the convolution $p(x) = (q*G)(x) = \mathcal{N}(x; 0, s^2+\sigma^2)$.

Explicit case. For explicit–η, the new dataset is

$$\tilde{x} = (1-\eta)x + \eta\int_{-\infty}^{\infty} p(y|x)y\,dy \qquad \forall x \in \mathbb{R} \tag{8}$$

where, by Bayes' theorem, $p(y|x) = G(x-y)q(y)/p(x) = \mathcal{N}(y; rx, r\sigma^2)$ with

$$r = \frac{1}{1+(\sigma/s)^2} \in (0,1). \tag{9}$$

Thus, the integral equals the posterior mean $\mathrm{E}\left\{y|x\right\} = rx$ and so the new dataset is a linear map of the original dataset:

$$\tilde{x} = (1-\eta+\eta r)x = \phi(r)x = f(x), \tag{10}$$

with distribution $p(\tilde{x}) = \mathcal{N}(\tilde{x}; 0, (\phi(r)s)^2)$.

Implicit case. The implicit–η case is more complicated. The new dataset satisfies

$$(1+\eta)f(x) - \eta\int_{-\infty}^{\infty} p(y|x)f(y)\,dy = x \qquad \forall x \in \mathbb{R} \tag{11}$$

where $\tilde{x} = f(x)$ is the unknown mapping that we want to determine and $p(y|x) = \mathcal{N}(y; rx, r\sigma^2)$ as before. The integral

$$I = \int_{-\infty}^{\infty} p(y|x)f(y)\,dy, \tag{12}$$

which is the posterior expectation of $f(y)$, can be solved as follows:

1. Change variables $z = y - rx$, so

$$I = \int_{-\infty}^{\infty} p(z|x)f(z+rx)\,dz \tag{13}$$

   with $p(z|x) = \mathcal{N}(z; 0, r\sigma^2)$.

2. Taylor-expand $f(z+rx) = f(rx) + f'(rx)z + \frac{1}{2}f''(rx)z^2 + \dots$ We get:

$$I = f(rx) + \frac{1}{2}f''(rx)r\sigma^2 + \sum_{n=2}^{\infty}\frac{f^{(2n)}(rx)}{(2n)!}\mu_{2n} \tag{14}$$

   where μ_{2n} is the $2n$-th moment of the Gaussian $\mathcal{N}(z; 0, r\sigma^2)$.

Plugging I into (11) we obtain an ordinary differential equation for f:

$$(1+\eta)f(x) - \eta\left(f(rx) + \frac{1}{2}f''(rx)r\sigma^2 + \begin{matrix}\text{terms in}\\ f^{\text{iv}}, f^{\text{vi}}, \dots\end{matrix}\right) = x.$$

Since its solution must be unique and a linear function satisfies it, the solution is

$$f(x) = \frac{1}{1+\eta-\eta r}x = \phi(r)x \tag{15}$$

so the new dataset is again (as in the explicit case) a linear map of the original: $\tilde{x} = \phi(r)x$ with distribution $p(\tilde{x}) = \mathcal{N}(\tilde{x}; 0, (\phi(r)s)^2)$.

It is easy to see that the approach holds for polynomials ϕ (again, these are just sums of repeated expectations) and so for all the algorithms defined in the paper. □

The previous theorem reduces the analysis of the matrix update to the analysis of the scalar function $\phi(r)$, $r \in (0,1)$. We can now determine conditions and order of convergence. The sequence of standard deviations satisfies $s^{(\tau+1)} = |\phi(r(s^{(\tau)}))|\,s^{(\tau)}$ for $s^{(0)} > 0$. A general condition on ϕ for the sequence to converge to 0 may be obtained from the contractive mapping theorem, namely $-\infty < \phi(0) < \infty$ (so $s = 0$ is a fixed point) and $|d(s\phi(rs))/dx| < 1\ \forall s > 0$, which by means of the chain rule leads to $|\phi(r) + 2r(1-r)\phi'(r)| < 1\ \forall r \in (0,1)$. Simpler conditions may be obtained for the functions ϕ considered in the paper by requiring $|\phi(r)| < 1$ so that the cluster shrinks. The order p of convergence is the largest $p > 0$ for which $\varrho = \lim_{\tau\to\infty}|s^{(\tau+1)} - s^{(\infty)}|/|s^{(\tau)} - s^{(\infty)}|^p < \infty$. The results are given in table 1 for each method and its step size.

References

[1] M. Á. Carreira-Perpiñán. Fast nonparametric clustering with Gaussian blurring mean-shift. In *ICML*, 2006.

[2] M. Á. Carreira-Perpiñán. Gaussian mean shift is an EM algorithm. *IEEE Trans. PAMI*, 29(5):767–776, May 2007.

[3] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. PAMI*, 17(8):790–799, Aug. 1995.

[4] E. Choi and P. Hall. Data sharpening as a prelude to density estimation. *Biometrika*, 86(4):941–947, Dec. 1999.

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 2002.

[6] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 1999*, pp. 317–324, 1999.

[7] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with application in pattern recognition. *IEEE Trans. Info. Theory*, 21(1):32–40, 1975.

[8] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, third edition, 1996.

[9] M. Hein and M. Maier. Manifold denoising. In *NIPS* 19, pages 561–568, 2007.

[10] K. Ho-Le. Finite element mesh generation methods: A review and classification. *Computer-Aided Design*, 20(1), 1988.

[11] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, Aug. 2000.

[12] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 1995*, pages 351–358, 1995.

[13] G. Taubin. Geometric signal processing on polygonal meshes. In *Eurographics'2000: State of the Art Reports*, 2000.