Analysis and Application of the Generalized Mean-Shift Process

Yizong Cheng
Department of ECECS, University of Cincinnati
email: yizong.cheng@uc.edu

Abstract

The mean shift process repeatedly moves each data point to the mean of data points in its neighborhood. This process is generalized and analyzed. Its relation with maximum-entropy and K-means clustering methods is studied. Its nature of gradient mapping is revealed. Its applications in clustering, Hough transform, and overfitting relaxation are examined.

1 Introduction

The traditional paradigm of partitioning clustering is optimization. The problem is often formulated as partitioning the data set into k groups such that some global measure is optimized. An old and popular example is the k-means clustering, in which the data is divided into k (a system parameter) groups and the *within-group sum of squared errors* is minimized [11]. A more recent example is the *maximum-entropy* clustering algorithm [10], where the within-group sum of squared errors is given and the entropy is the one to maximize.

It has been known that problems like these are NP-hard [1]. Practical algorithms for partitioning clustering are thus often hill-climbing ones that finds only a local optimum that depends on some randomized or calculated initialization. Clustering algorithms become probabilistic, and the outcomes may be quite different from the suggested globally optimal solution to the formulated problem.

Similar situations arise in other areas of computational self-organization that include genetical algorithms, decision tree learning, and neural networks. Some of these algorithms are formulated as means attacking global optimization problems which are essentially intractible, and in reality settle down at certain local optima. Because it is unlikely for nature to solve intractible problems, some of these models may have idealized the netural mechanisms they were intended to mimic.

A recent example of nature vs. global optimization is the protein folding problem. The traditional formulation of the problem is to find the folded conformation having the minimum energy based on the protein's amino sequence. It has been shown that both this problem and the so-called inverse protein folding problem are NP-hard [6] [5]. The current scheme is to reformulate the problem into finding a pronounced local energy minimum based on a folding procedure [12], which is confirmed by many native protein configurations.

In this paper, we address an iterative self-organizing process called "mean-shift", and show its clustering behavior and some of its potential applications. Section 2 contains definitions of the mean shift process. In Section 3, it is proved that mean shift is gradient ascent. Section 4 considers the convergence and termination of the algorithm. Section 5 contains application examples in clustering.

2 The Mean-Shift Process

Let X be a Euclidean space and $S \subset X$ a finite subset called the *data*. Let λ be a positive number. A *flat kernel* is the following characteristic function for a λ-ball in X.

$$K(x) = \begin{cases} A & \text{if } |x| \le \lambda \\ 0 & \text{if } |x| > \lambda \end{cases} \tag{1}$$

A *sample mean* in the λ-neighborhood of $x \in X$ is

$$m(x) = \frac{\sum_{s \in S} K(s-x)s}{\sum_{s \in S} K(s-x)}. \tag{2}$$

The difference $m(x) - x$ is called *mean shift* by Fukunaga and Hostetler [4] and is used as an estimate of the density gradient at x.

Fukunaga and Hostetler also propose a *mean shift clustering algorithm* that is the repeated replacement of each data point $s \in S$ with the sample mean

of data in its neighborhood.

$$s \leftarrow m(s), \quad s \in S. \tag{3}$$

This process transforms the data set S generation by generation, and data points in S eventually merge into several clusters. This process is also discussed in Silverman[13]. This mean shift clustering algorithm also occurs in Cheng and Fu[2], where categorical data are embedded into a Euclidean space and quantization is used.

The flat kernel weights all data points in the λ-neighborhood evenly during the computation of the sample mean, $m(x)$. This boundary is rather arbitrary and abrupt. A generalization is to allow other kernels to be used in (2). The kernels considered in this paper are limited to those satisfying the following properties. First, a *kernel* must be a nonnegative real function K on X in the form of

$$K(x) = k(\|x\|^2) \tag{4}$$

where k is a piecewise continuous and non-increasing function defined over $[0, \infty)$: $k(a) \geq k(b)$ if $a < b$. We also require that $\int_0^\infty k(r)dr < \infty$. Let $\alpha > 0$ be a constant. When K is a kernel, so are K_α, where $K_\alpha(x) = K(x/\alpha)$, K^α, where $K^\alpha(x) = (K(x))^\alpha$, and αK, where $(\alpha K)(x) = \alpha K(x)$. When used in mean shift, (2), K and αK generate the same result.

A further step to generalize mean shift is to allow a weight independent to the kernel to be associated with each data point. This weight can be used to indicate the relative significance of a data point or any local context effect. In density estimation literature[13], this is considered as *adaptive kernel estimation*. With this weight, w, the kernel term $K(s-x)$ in (2) is replaced by $K(s-x)w(s)$ and the sample mean becomes

$$m(x) = \frac{\sum_{s \in S} K(s-x)w(s)s}{\sum_{s \in S} K(s-x)w(s)}. \tag{5}$$

Finally, mean shift can be performed on a subset $T \subset X$ and the data set S may remain unchanged.

$$t \leftarrow \frac{\sum_{s \in S} K(s-t)w(s)s}{\sum_{s \in S} K(s-t)w(s)} \quad t \in T, \tag{6}$$

The original mean shift algorithm is a special case of this when T is S. However, in general, T can be different from S. One choice is that T is initialized to S and then goes its own way. A more common choice is to have a randomly initialized T, with a size much smaller than S. This is what a large group of iterative clustering algorithms do.

This group of clustering algorithms also use a very special local weight:

$$w(s) = \frac{1}{\sum_{t \in T} K(s-t)}, \quad s \in S. \tag{7}$$

When the kernel is a Gaussian one,

$$K(x) = e^{-\beta \|x\|^2}, \quad \beta \geq 0, \tag{8}$$

the iteration (6) is the maximum entropy clustering algorithm of Rose, Gurewitz, and Fox[9]. When $\beta \to \infty$, the product $K(s-t)w(s)$ degenerates to

$$v(t,s) = \begin{cases} 1, & \text{if } t = \text{argmin}_T |s-t|^2 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

the iteration degenerates to the ordinary K-means clustering[10]. Indeed, when K is any strictly decreasing kernel, where $K(x) < K(y)$ if $|x| > |y|$, K^β when $\beta \to \infty$ as the kernel will make (6) and (7) approach K-means clustering.

3 Mode Seeking Properties

In this section, we investigate the mode-seeking properties of mean-shift steps, and in the next section, we study the convergence properties of iteration of these steps. The first result is that mean shift is hill climbing on a surface defined with a "shadow" kernel, which is related to the kernel used in mean shift by Theorem 1.

Theorem 1. From each kernel $K(x) = k(|x|^2)$, we can construct a kernel (called a *shadow* of K) $K_1(x) = k_1(|x|^2)$ such that

$$k_1(r) = \int_r^\infty k(t)dt. \tag{10}$$

The mean shift $m(x) - x$ using kernel K is in the direction of the gradient of the data density estimate using kernel K_1:

$$q(x) = \sum_{s \in S} K_1(s-x)w(s). \tag{11}$$

Proof. The mean shift using kernel K can be rewritten as

$$m(x) - x = \frac{1}{p(x)} \sum_{s \in S} k(\|s-x\|^2)w(s)(s-x) \tag{12}$$

with $p(x) = \sum_{s \in S} K(s-x)w(s)$, the density estimate using K. The gradient of (11) at x is

$$\nabla q(x) = -2 \sum_{s \in S} k_1'(\|s-x\|^2)(s-x)w(s). \tag{13}$$

To have (12) and (13) point to the same direction, we need $k_1'(r) = -ck(r)$ for all r and some $c > 0$. By the fundamental theorem of calculus and the

requirement that $\int_0^\infty k_1(r)dr < \infty$, (10) is the only solution. In this case, we have

$$m(x) - x = \frac{\nabla q(x)}{2cp(x)}, \tag{14}$$

or, the magnitude of mean shift is in proportion to the ratio of the gradient and the local density estimate using kernel K. □

Using formula (10), we find that mean shift with a flat kernel (1) is in the gradient direction of the density estimate using the *Epanechnikov kernel*,

$$K(x) = \begin{cases} 1 - |x|^2 & \text{if } |x| \leq \lambda \\ 0 & \text{if } |x| > \lambda \end{cases} \tag{15}$$

and mean shift with the Epanechnikov kernel is in the gradient direction of the density estimate using the *biweight kernel* [13],

$$K(x) = \begin{cases} (1 - |x|^2)^2 & \text{if } |x| \leq \lambda \\ 0 & \text{if } |x| > \lambda \end{cases}. \tag{16}$$

Theorem 1 characterizes mean shift as gradient ascent over a "shadow" surface. Mean shift is proportional to the gradient of the surface. It would be called a *gradient mapping* if mean shift were exactly the gradient. Theorem 2 below shows that Gaussian kernels are unique because they make mean shift a gradient mapping.

Theorem 2. Mean shift is a gradient mapping if and only if the kernel is Gaussian (8).

Proof. The "symmetry principle" says that a continuously differentiable mapping $F : R^n \to R^n$ is a gradient mapping if and only if the Jacobien matrix of F is symmetric [7]. Let F be identified with mean shift (12). Then for $i \neq j$,

$$\begin{aligned} \frac{\partial F_i(x)}{\partial x_j} = \frac{1}{p^2(x)} \Bigg(& 2 \sum_{s \in S} k'(\|s - x\|^2)(x_j - s_j)w(s) \\ & (s_i - x_i) \sum_{t \in S} k(\|t - x\|^2)w(t) \\ & -2 \sum_{s \in S} k'(\|s - x\|^2)(x_j - s_j)w(s) \\ & \sum_{t \in S} k(\|t - x\|^2)w(t)(t_i - x_i) \Bigg) \end{aligned} \tag{17}$$

The first term in the sum is symmetric with respect to the subscripts i and j. The second term is symmetric for *all* x values if and only if $k'(r) = -\beta k(r)$ with some constant β. Using the method of separation of variables, we have $\int dk/k = -\int \beta dr$ or $\log k(r) - \log k(0) = -\beta r$, which leads to $k(r) = k(0)e^{-\beta r}$, or the Gaussian kernel. In this case, mean shift $m(x) - x$ is equal to the gradient of the function

$$q(x) = \frac{\log \sum_{s \in S} e^{-\beta \|s - x\|^2} w(s)}{2\beta}. \tag{18}$$

□

Suppose there is a "perfect" mode in the density surface, for example,

$$q(x) = e^{-\gamma |x|^2}. \tag{19}$$

Then, $m(x) - x = -(\gamma/\beta)x$, which shows that data around a mode will shift toward the mode at a rate of $1 - \gamma/\beta$.

If a few discontinuous points on k are ignored, then a truncated Gaussian kernel also makes mean shift a gradient mapping, with the kernel being its own shadow.

The mode seeking mechanism of mean shift can be viewed as adaptive gradient ascent. The difference between this mechanism and gradient ascent with a fixed or *ad hoc* step size is important. When $\nabla q(x)$ is small, gradient ascent with a fixed step size will stall. $\log q(x)$ magnifies the difference in flat low density areas and thus avoids the plateau phenomenon.

4 Convergence

Mean shift can be repeated in various ways. In general, it can be applied to a set of *cluster centers*, T, such that $T \leftarrow m(T)$ is the iteration. The algorithm terminates when $T = m(T)$, or for all $t \in T$, $t = m(t)$. When S, the data, and w, the weight, do not change, the data density estimate $q(x)$ (11) remains the same, and each T point moves independently and settles at a local maximum of $q(x)$. Overall, the cluster centers, T, evolve to a fixed point that is a local maximum of the function

$$U(T) = \sum_{t \in T} q(t) = \sum_{t \in T} \sum_{s \in S} K_1(t - s)w(s), \tag{20}$$

where K_1 is a shadow of K.

One particular case is when T is initialized as S. Mean shift iterates will settle on a local maximum of $U(T)$ that can be reached via hill climbing from the initial U value

$$U(S) = \sum_{t \in S} \sum_{s \in S} K_1(t - s)w(s). \tag{21}$$

When T is S, and when $w(s)$ does not change, the process $S \leftarrow m(S)$ has stable fixed points that

are the local maxima of

$$V(S) = \sum_{s,t \in S} K_1(s-t)w(s)w(t). \qquad (22)$$

We will call mean shift with $T = S$ the *blurring process.*

In these two cases, that is, when T is S or is initialized to S, the initial S and the chosen kernel completely determine the final fixed point and also the particular local maximum of U or V the process will reach.

When S is fixed, each T point converges to a mode (local maximun) of $q(x)$. This mode seeking process is completely local and the limited precision in real execution of the process will make the fixed point reachable in finite time. The rest of the computation burden is the performance of each mean shift step. If the kernel has an infinite support, then the time required to complete a mean shift step is $O(n^2)$ where n is the data size. The number of iterations before termination is not a function of data size and can be considered as a constant. When the kernel has a limited support, that is, when it vanishes outside a neighborhood, data may be organized so that finding one's neighbors requires only $O(\log n)$ time and thus the overall time will be between $O(n \log n)$ and $O(n^2)$.

In the case of the blurring process, the use of a kernel with infinite support will result in a trivial outcome: all data points converge to a single point. This is also the case when the kernel support is finite but still large enough to cover the data set. When the kernel support is smaller than a critical volumn, data points converge to multiple "cluster centers". When the kernel support is even smaller, the initial data set will be a fixed point of the process and no clustering will take place.

The following theorem shows that, the blurring process terminates in finitely many steps, provided that data points are not infinitely close to each other.

Theorem 3. Suppose there is a minimum distance $\delta > 0$ between data points during the blurring process and there is a $\kappa > 0$ such that when $K(x) > 0$, $K(x) > \kappa k(0)$. Then the blurring process reaches its fixed point within finite number of iterations.

Proof. In a blurring process, the convex hull of data is shrinking. When it stops shrinking, the border data points can be eliminated from consideration because they are no longer influenced by other points and thus no longer influence others. The same argument then can be applied to the rest of the data set.

Some conditions are necessary to make the border points reach their final positions in finite number of steps. Without these conditions, the final configuration may only be approached but never reached. The purpose of these conditions is to avoid infinitesimal shifts that will never end. The first condition is that the distance between data points cannot be infinitely small. That is, when two data points are within each other's δ ball, for a small constant δ, they merge. The second condition is that the influence of a data point to another cannot be infinitely small. The way to guarantee this is to choose a kernel that is truncated, so that either two points are too far apart and thus do not influence each other, or their influence is larger than a small lower bound.

Define the *radius* of data, $r(S)$ as the distance from the origin (or any reference point) to the data point farthest from the origin. There exists a $h > 0$, which is related to both δ and the dimensionality of the space, such that in any direction a, there can be at most one data point $s \in S$ that satisfies $\pi_a(s) > r(S) - h$, where π_a is the projection mapping along the direction a. For any data point s to have $m(s) \neq s$, the kernel centering at s must cover at least two data points. For any direction a, one of these data points, u, must have the property that $\pi_a(u) \leq r(S) - h$, based on the definition of h. Hence, we must have

$$\begin{aligned} r(S) - \pi_a(m(s)) = & \\ & \frac{\sum_{t \in S} K(t-s)w(t)(r(S) - \pi_a(t))}{\sum_{t \in S} K(t-s)w(t)} \\ \geq & \frac{K(u-s)w(u)(r(S) - \pi_a(u))}{\sum_{t \in S} k(0)w(t)} \\ \geq & \kappa W h, \end{aligned} \qquad (23)$$

where

$$W = \frac{\min_{s \in S} w(s)}{\sum_{s \in S} w(s)} \qquad (24)$$

is a constant. This shows that the radius of data is either its final value, when the farthest point does not have a neighbor, or shrinking by at least a constant $\kappa W h$.

After a number of iterations that is less than $r(S)/(\kappa W h)$, the most remote data points reache their destination and can be removed from consideration and the rest of the data set is smaller.

If n is the number of data points in S, then the previous argument proves that the blurring process is at worst an $O(n^3)$ algorithm. □

5 Clustering

Clustering is the most obvious application of the mean shift process. Data points can be traced to

their final positions along the mean shift trajectories, if T is S or is initialized to S. A family of kernels can be chosen to perform mean shift, with a parameter that can be used as a measure of "resolution" for the "blurring" effect. For the blurring process, because kernels with infinite supports will merge all data points into one, an appropriate choice is a truncated Gaussian kernel,

$$K(x) = \begin{cases} e^{-\beta\|x\|^2} & \beta\|x\|^2 < 1 \\ 0 & \text{otherwise} \end{cases} . \qquad (25)$$

In the case of fixed data set, either a Gaussian or a truncated Gaussian family may be chosen. When the β value is small, the kernel encompasses the data set, and all merge to one point. When β is large enough, the initial data set becomes a fixed point. When β takes intermediate values, different merger patterns emerge. The general picture is that the larger β is, the more clusters the mean shift process will end up with. Those clustering outcomes that share large ranges of β should be considered more stable and more valid than those that are essentially transient.

When T is either S (the blurring process) or is initialized to S (a special case of clustering algorithms like the maximum-entropy clustering), the algorithm is deterministic, because there is no need to randomly initialize cluster centers. The number of clusters in the outcome is determined by β and the nature of the initial data set. By removing the probabilistic factor that makes the outcome uncertain, and removing the number of clusters as a system parameter, the clustering result becomes unique and characteristic of the data set and thus more meaningful.

Three mean-shift clustering examples will be demonstrated in this section.

Example 1. Roeder[8] fits the velocities of 82 galaxies with a bimodal normal mixture model and concludes that the observable universe contains two superclusters of galaxies surrounded by large voids. A study of the clustering result (Figure 1) from the blurring process with different β values concludes that the dominant outcome is three instead of two clusters. This shows a benefit of the blurring process, namely that the number of clusters is the result instead of a system parameter. □

Example 2. In a pattern matching method called the *generalized Hough transform*, subpatterns of a template are listed along with their relative locations to the center of the template. Then, the same subpatterns are searched in a larger image from which occurrences of the template pattern with slight distortion are to be located. When there is an exact match of a subpattern, a location relative to the subpattern where the center of an occurrence of the template might be is registered in a parameter space. High modes of the possible centers in the parameter space are then detected and considered as centers of possible occurrences of the template pattern in the larger image. To allow slightly distorted patterns to match the template, the parameter space is divided into bins and each subpattern casts vote to the bins that contains possible centers. The size and the boundary locations of the bins are system parameters to be determined by trial and error. In this example, a generalized Hough transform with the blurring mapping is used for the same purpose. The bin design is replaced by blurring in the parameter space.

A 270×270 image of 225 chinese characters (Fig. 2a) was used in this experiment. A template, shown in Fig.7b with a magnified view was to be found at any location in the image. Subpatterns were 3×3 windows with the center cell black. There are $2^8 = 256$ possible subpatterns, but only about 20 are present in the template. The probability of the occurrence of each subpattern in this context was estimated by frequency counting. The negative logarithm of this probability was used as an additive weight assigned to each subpattern. The image in Fig. 2a was scanned with a 3×3 window. When a subpattern matched one in the template, the weight of that subpattern was added to all the possible centers (pixel locations) that may generate the subpattern. Fig. 2c is the distribution of pixels with nonzero weights in the parameter space (also a 270×270 square) after the scanning. The pixels in this space with the eight highest accumulated weights were detected and the corresponding 19×19 cells in the original image centering at these locations are displayed in Fig. 2e. A blurring step was applied to the pixels with nonzero weights with a 5×5 neighborhood (a flat kernel). The resulting distribution of pixels with nonzero weights is shown in Fig. 2d, and the pixels with the eight highest weights were detected with the corresponding subimages displayed in Fig. 2f. Some of these pixels were close to each other, and thus generated overlapping subimages. The result in Fig.7f is better than that in Fig. 2e, generated before blurring. Three subimages that do not contain the template were no longer in Fig. 2f, while one subimage that contains the template with considerable distortion got in the picture. This is because distortion disperses the possible centers to several neighboring

locations, a blurring step merges them and allows the mode to win the selection. A bin design used in traditional generalized Hough transform may not solve the problem, because the boundary of a bin may cut the mode in halves, and neither half may win the selection. □

Example 3. When a training sample with noise is to be fitted by a curve, surface, or other functions as a possible model for an input-output system, one of the problems is overfitting. The curve, surface, or the function chosen by the fitting algorithm may fit the training sample *too well.* That is, the curve fits not only the model underlying the training sample, but the noise as well. The standard approach is to run a sequence of fittings with ascending complexity of the curve. Then a testing sample is used to weed out the effect of the noise. The curve in the sequence with the best fit for the testing sample is selected. For example, when a multi-layer feed-forward neural network [8] is used to fit a sample using the backpropagation algorithm, it is common to generate the fitting with the training sample using increasing number of hidden units. After the testing sample is tested on each of these trained networks, the one with the smallest error is chosen. Sometimes, too many hidden units generate an overly complex network that fits the noise in stead of the underlying model.

The blurring process can be used on an overfitting network to merge the hidden units, reduce the complexity, and thus smooth out the noise. With different kernel sizes, various reductions of the hidden units will be generated. Because the blurring process is very efficient, compared to the backpropagation training for a neural network, one can reach the same optimal fitting with only one backpropagation training phase and a sequence of application of the blurring process.

To demonstrate this application of the blurring process, a two-input and one-output function was to be learned with backpropagation. 20 points (x_i, y_i), $i = 1, \ldots, 20$ were randomly selected from the unit square, with corresponding output values z_i's generated using the following equation.

$$z = (1 + \nu)e^{-(x^2+y^2)} \tag{26}$$

where ν was a random number uniformly distributed in $[-0.1, 0.1]$, serving as the noise. The values of these sample points are listed in Fig. 3a. A multi-layer feed-forward neural network with one hidden layer was used as the fitting function:

$$o = \sum_j v_j g\left(w_{jx}x + w_{jy}y + w_{js}\right). \tag{27}$$

When there are n hidden units, there are $4n$ *weights* or unknown system variables, $v_j, w_{jx}, w_{jy}, w_{js}$, $j = 1, \ldots, n$, to be solved. The nonlinear function g must be an odd function. In particular, the tanh function was used.

$$g(a) = \frac{1 - e^{-a}}{1 + e^{-a}}. \tag{28}$$

The first 10 sample points were used in training. The number of hidden units was chosen to be 40, to ensure overfitting. The weights were randomly assigned values in the range of $[-1, 1]$. Backpropagation was used to modify them until the square error for the training set reaches 0.001 (after some 50,000 iterations with step size 0.1). The 10 sample points not used in training were used in testing, resulting in a square error of 0.019.

The blurring process with a truncated Gaussian kernel and squared radius from 0.01 to 0.60 was performed on the input weight vectors (w_{jx}, w_{jy}, w_{js}), $j = 1, \ldots, 40$, while the jth output weight, v_j, was used as the initial probability of the jth input weight vector (with necessary normalization). When v_j is negative, the signs of all weights for the jth unit were reversed. Because g is an odd function, this will not affect the trained model. On average, after eight iterations, the blurring process halted when no change larger than 0.01 in any direction took place. Shifted input weight vectors within the kernel radius to each other were merged (they should be very close to each other by the time the blurring process halted), and the sum of the probabilities (normalized output weights) of all merging individuals was used as the new probability for the combined unit. Fig. 3b shows the number of hidden units after blurring for different kernel radii.

Thereafter, both the training sample and the testing sample were used to test the reduced network. The squared roots of the square errors for these samples are displayed in Fig. 3c and Fig. 3d. The lowest square error for the testing sample occurs when the squared kernel radius was 0.22. Using the reduced network generated with this radius, the square error for the testing sample was 0.009 and that for the training sample was 0.006 (compared with 0.019 and 0.001 before blurring). The new network had only 19 hidden units (compared with 40 in the original one). The trajectories for the blurring process with radius 0.22 over two of the three dimensions for the input weight vectors are shown in Fig. 3e.

The optimally blurred network had a size that was half the original, but a fitting that was twice as good. Compared to the scheme of training a sequence of the networks, blurring is convincingly more efficient. □

References

[1] P. Brucker, "On the complexity of clustering problems", in Henn, Korte, and Oettli (eds.), *Optimization and Operations Research*, Berlin: Springer-Verlag, 1978.

[2] Y. Cheng and K.S. Fu, "Conceptual clustering in knowledge organization", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-7, 592-598, 1985.

[3] B.S. Everitt, *Cluster Analysis*, 3rd ed. London: Edward Arnold, 1993.

[4] K. Fukunaga and L.D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition", *IEEE Trans. Information Theory*, vol. IT-21, 32-40, 1975.

[5] R.H. Lathrop, "The protain threading problem with sequence amino acid interaction preferences is NP-complete", *Protein Engineering*, vol.7, 1059-1968, 1994.

[6] T. Ngo and J. Marks, "Computational complexity of a problem in molecular structure prediction", *Protein Engineering*, vol.5, 313-321, 1992.

[7] J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, San Diego: Academic Press, 1970.

[8] K. Roeder, "Density estimation with confidence sets exemplified by superclusters and voids in the galaxies", *J. Amer. Statist. Assoc.*, vol. 85, 617-624, 1990, also see [3].

[9] K. Rose, E. Gurewitz, and G.C. Fox, "Statistical mechanics and phase transitions in clustering", *Physical Review Letters*, vol. 65, pp.945-948, 1990.

[10] K. Rose, E. Gurewitz, and G.C. Fox, "Constrained clustering as an optimization method", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.15, 785-794, 1993.

[11] S.Z. Selim and M.A. Ismail, "K-means-type algorithms: a generalized convergence theorem and characterization of local optimality", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, 81-86, 1984.

[12] E.I. Shakhnovich and A.M. Gutin, "A new approach to the design of stable proteins", *Protein Engineering*, vol.6, 793-800, 1993.

[13] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall, 1986.

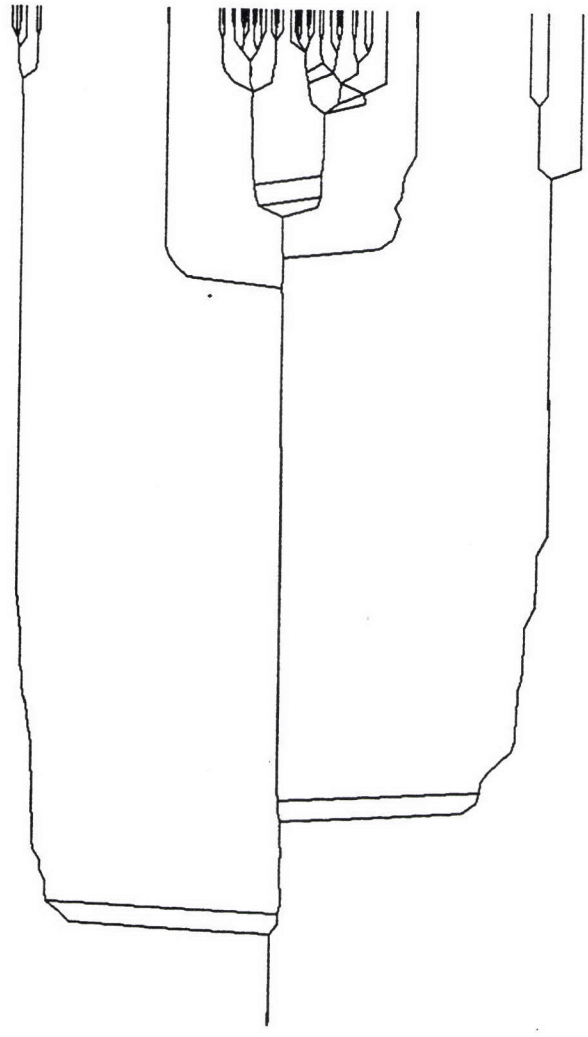


Figure 1: Clustering of 82 galaxies based on their velocities. Horizontal dimension is velocity, and vertical dimension is β value.

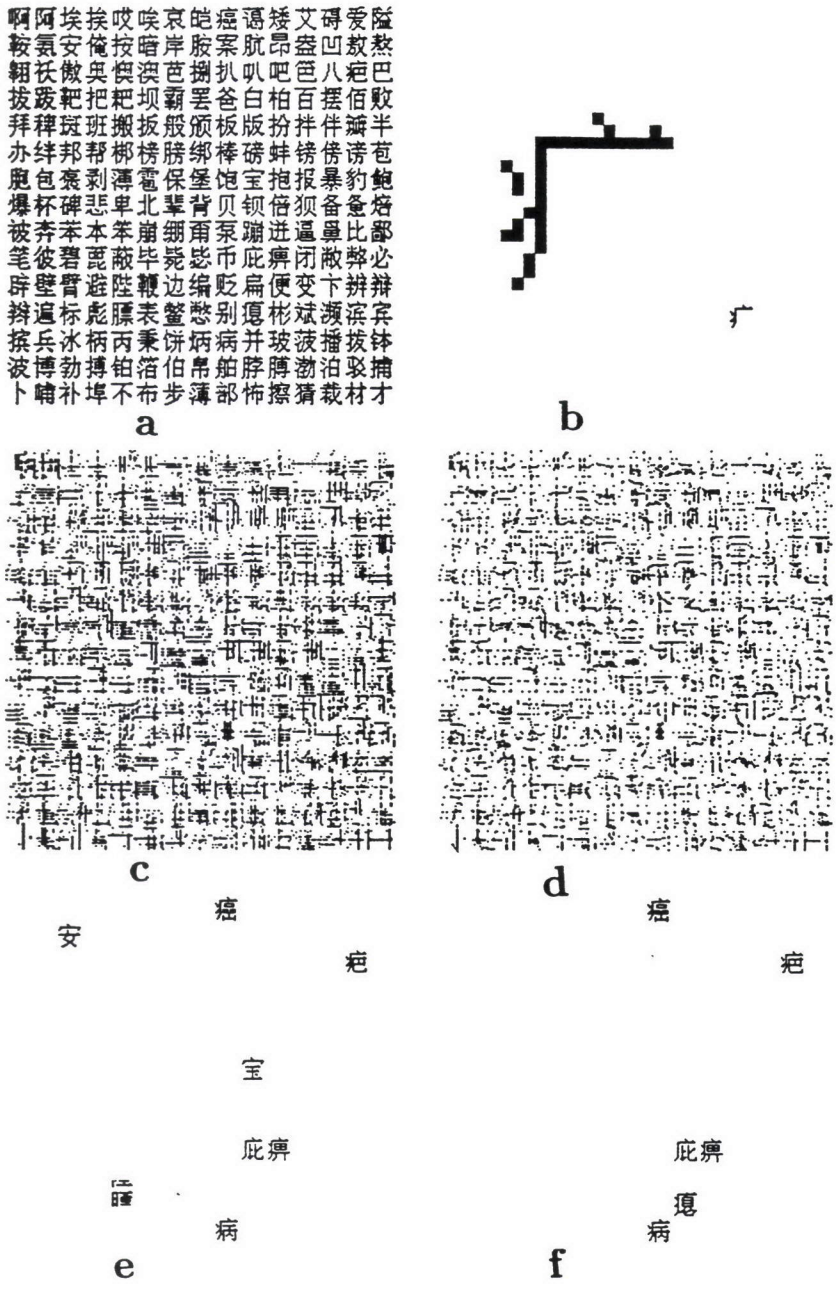
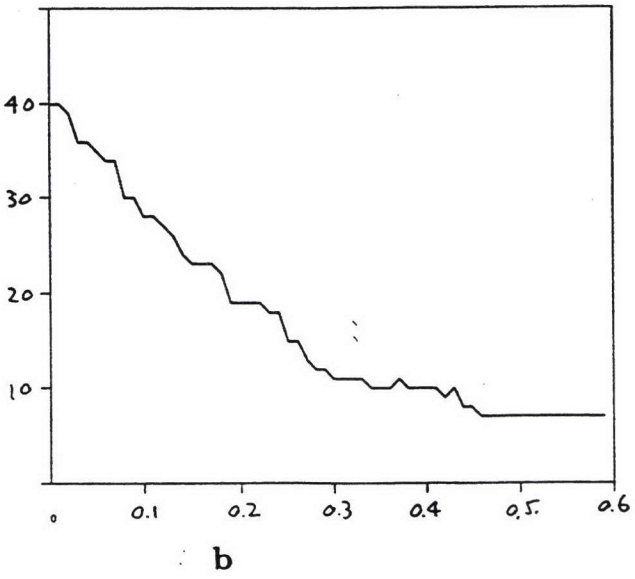


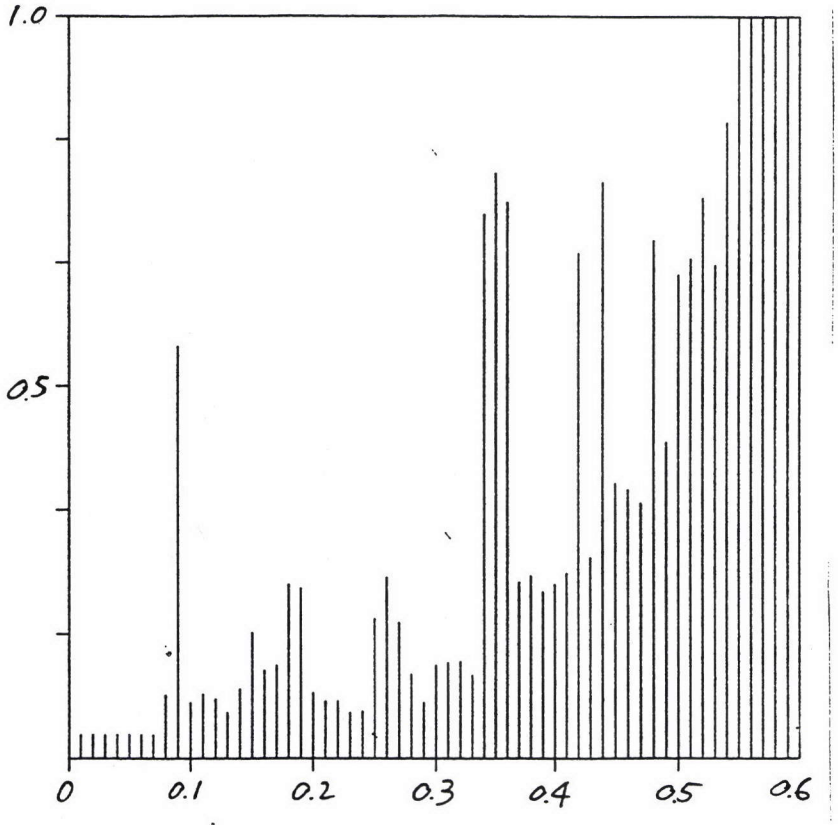
Figure 2: Blurring applied to the generalized Hough transform. **a.** A 270 × 270 image of 225 chinese characters. **b.** A template with its magnified version, to be used in pattern matching. **c.** The parameter space is the set of possible locations of the occurrences of the template, which is also 270 × 270. Black pixels are those locations with nonzero votes from 3 × 3 subpattern matching. **d.** After one blurring step with a flat square kernel of size 5 × 5, the new distribution of pixels with nonzero votes. **e.** 19 × 19 subimages from the original image at locations with the eight highest votes are displayed, some containing radicals similar to the template. **f.** Subimages from the original image corresponding to locations with the eight highest votes are displayed. Some locations are very close and thus only six characters are seen, but all contain parts very similar to the template.

x	y	z
0.4773	0.1004	0.7256
0.3066	0.9612	0.3405
0.0452	0.7563	0.6046
0.2483	0.0149	0.8598
0.8078	0.4742	0.3753
0.8295	0.6654	0.2957
0.4764	0.4885	0.5737
0.0155	0.5029	0.7769
0.0763	0.9360	0.3967
0.2446	0.4893	0.6973
0.1961	0.9666	0.3632
0.2984	0.2732	0.8090
0.5097	0.3184	0.6301
0.3769	0.5667	0.5709
0.4501	0.3745	0.7112
0.4623	0.2040	0.7244
0.5435	0.6804	0.4838
0.6132	0.6959	0.3949
0.1171	0.7722	0.5002
0.4087	0.0168	0.8616

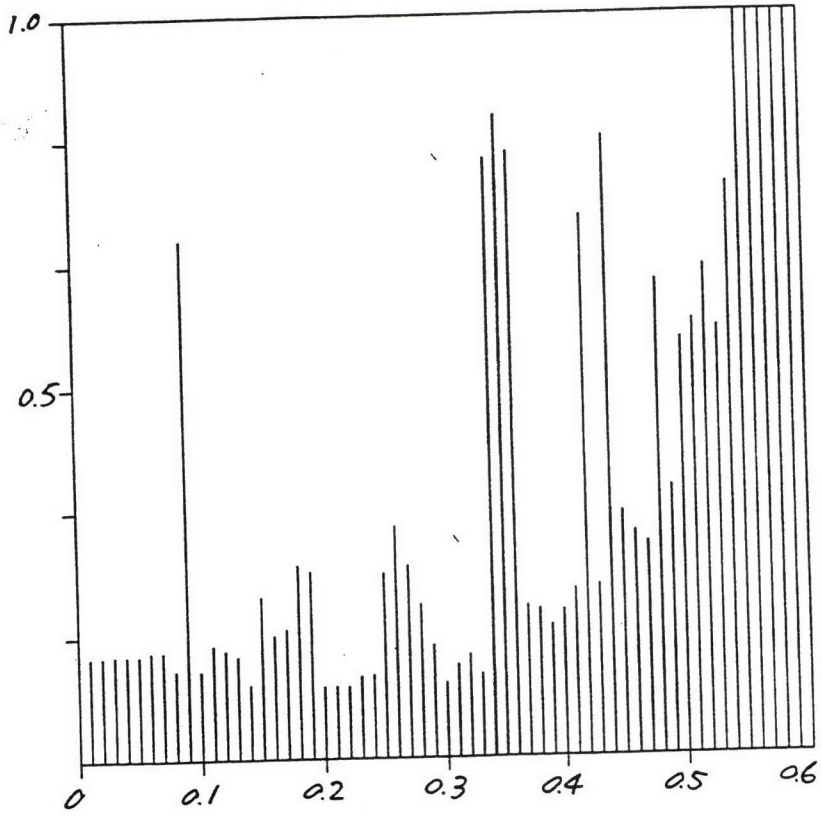
a



b



c



d

Figure 3:

e

Figure 3: Blurring applied to the backpropagation neural network. **a.** 20 sample points. The first 10 were used in training and the rest in testing. **b.** The 40 hidden units were blurred using different kernel size (truncated Gaussian). Some merged and the final number of hidden units is ploted against the kernel size. **c.** The square root of the sum of squares error from the reduced network on the original training sample is plotted against the kernel size. **d.** The square root of the sum of squares error from the reduced network on the testing sample is plotted against the kernel size. **e.** The trajectory of the blurring process on the hidden units when the square of kernel size is 0.22. Only two of the three dimensions are shown.