Entropic Affinities: Properties and Efficient Numerical Computation

Max Vladymyrov and **Miguel Carreira-Perpiñán**

Electrical Engineering and Computer Science
University of California, Merced
http://eecs.ucmerced.edu

June 18, 2013

ICML | Atlanta
International Conference on Machine Learning

Summary

- The entropic affinities define affinities so that each point has an effective number of neighbors equal to K.

- First introduced in:

G. E. Hinton & S. Roweis: "Stochastic Neighbor Embedding", NIPS 2002.

- Not in a widespread use, even though they work well in a range of problems.

- We study some properties of entropic affinities and give fast algorithms to compute them.

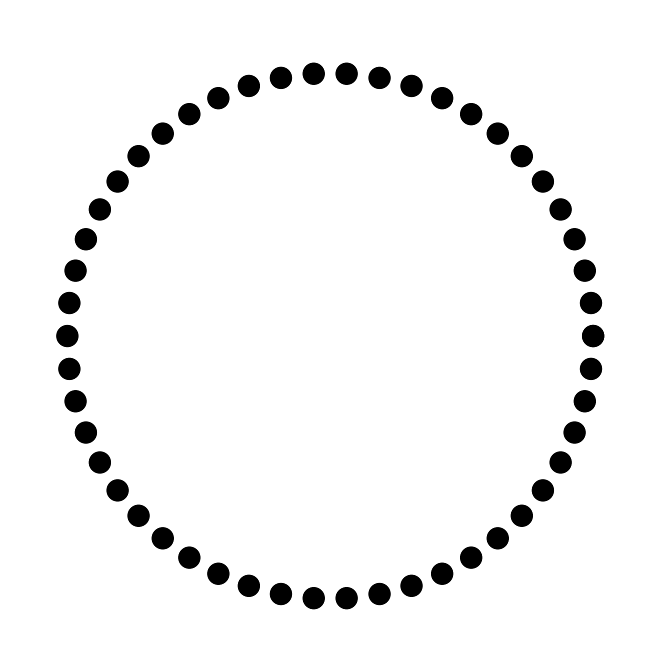
Affinity matrix

Defines a measure of similarity between points in the dataset.

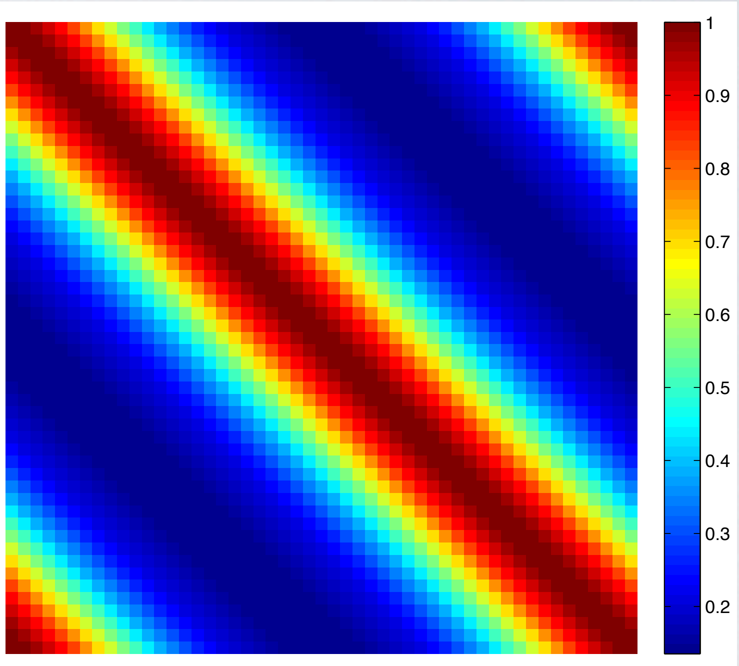
Used in:

- Dimensionality reduction:
  - Stochastic Neighbor Embedding, *t*-SNE, Elastic Embedding, Laplacian Eigenmaps.
- Clustering:
  - Mean-Shift, Spectral clustering.
- Semi-supervised learning.
- and others

Data set



Affinity matrix



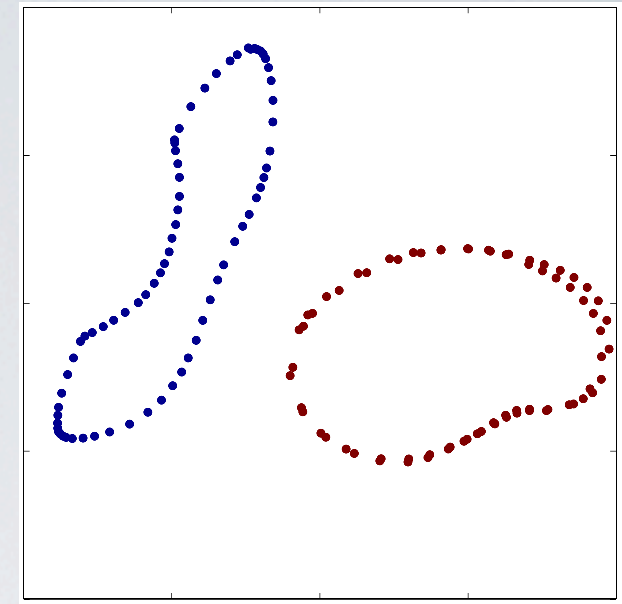
The performance of the algorithms depends crucially of the affinity construction, govern by the bandwidth σ.

Common practice to set σ:

- constant,
- rule-of-thumb (e.g. distance to the 7th nearset neighbor, Zelnik & Perona, 05).

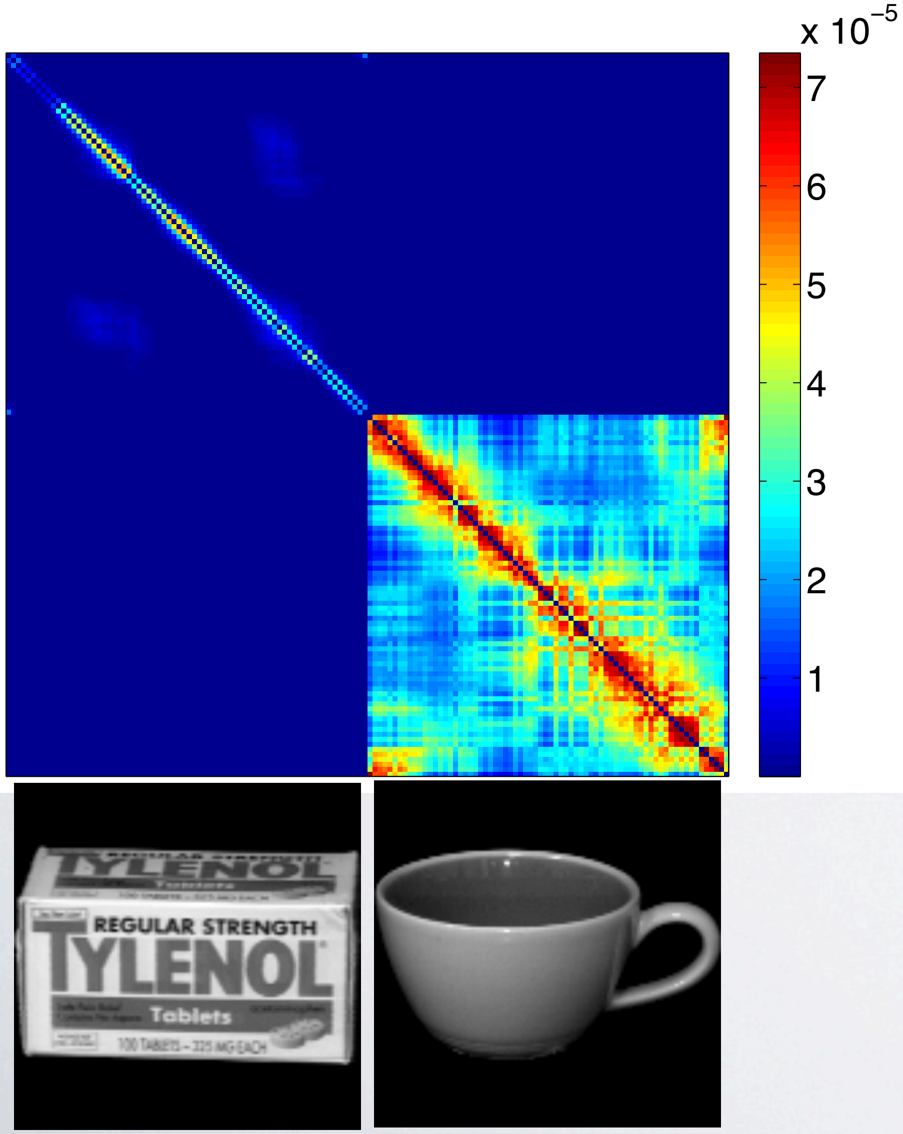
Motivation: choice of σ

COIL-20: Rotations of objects every 5°; input are greyscale images of 128×128.

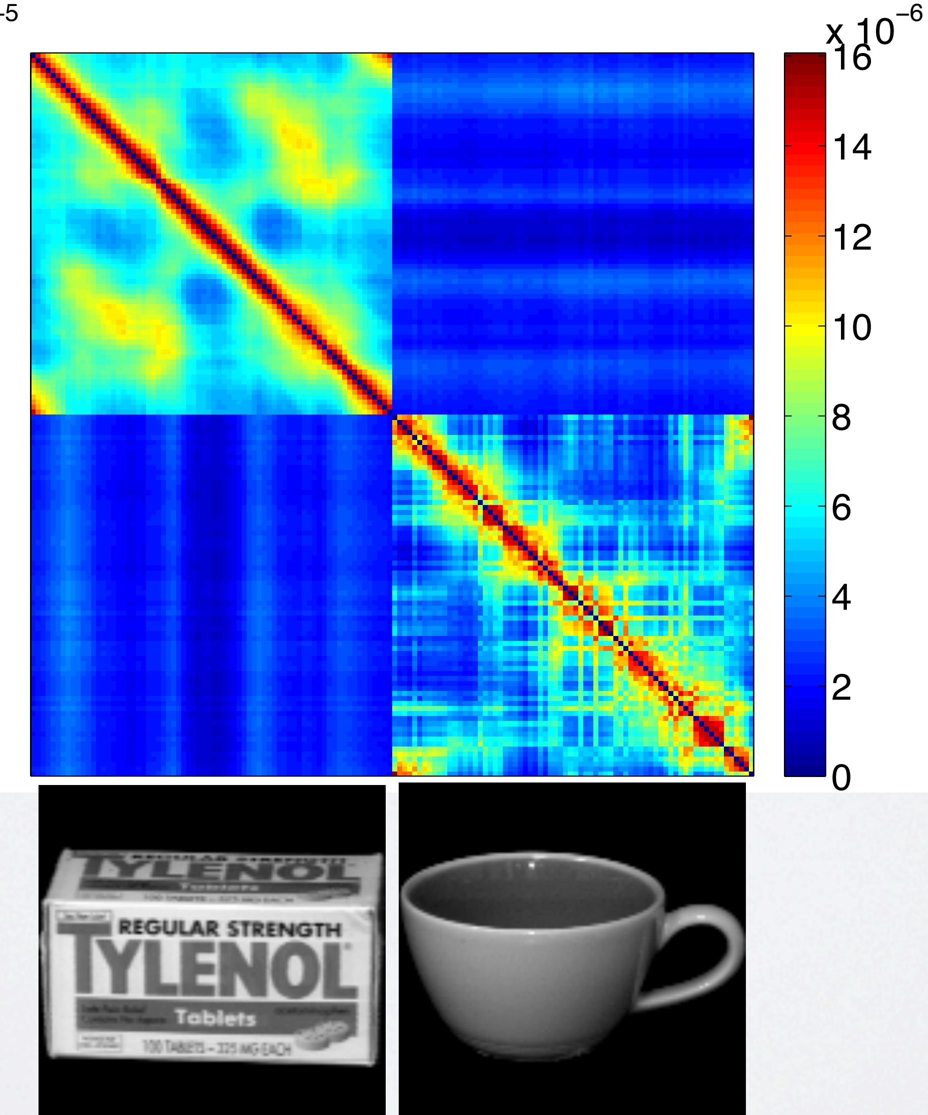


Affinity matrices:

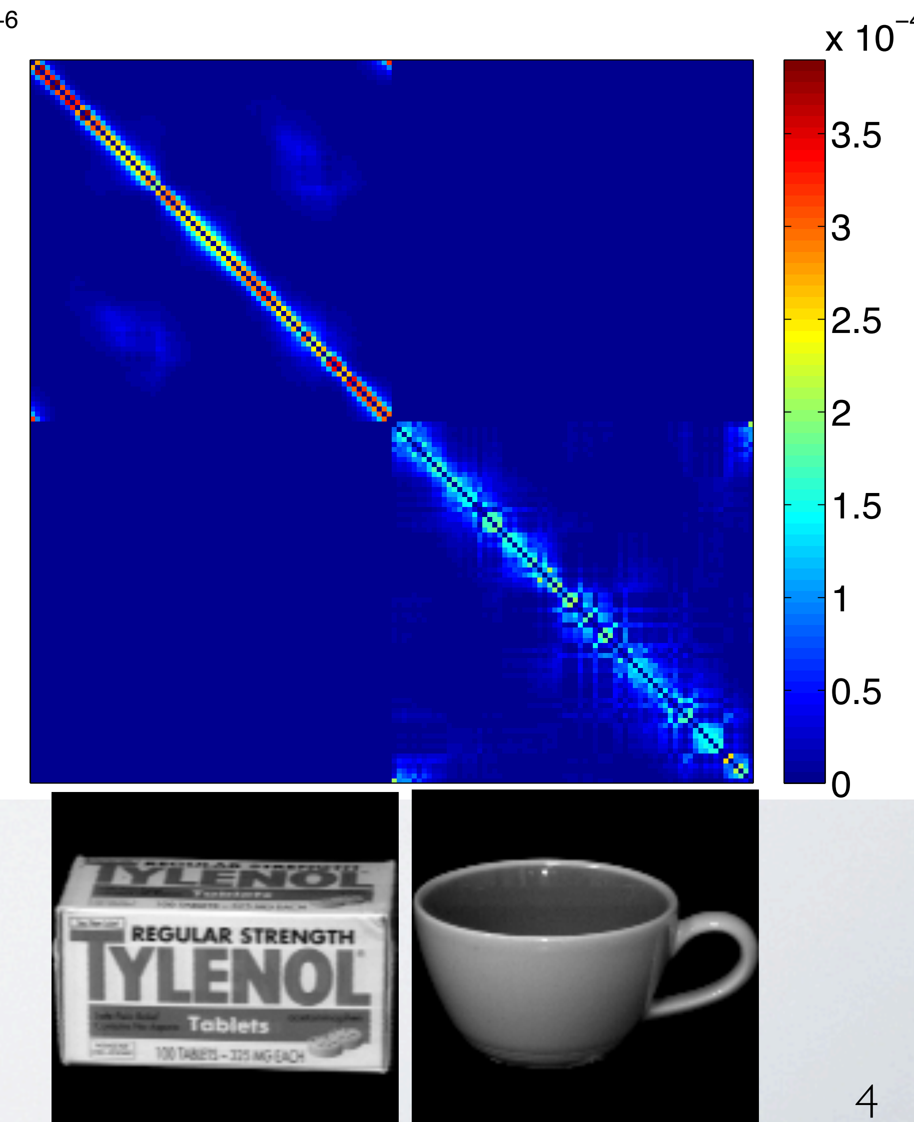
Constant sigma



Rule-of-thumb:
Dist. to the 7th nn (Zelnik & Perona, 05)



Entropic affinities



Motivation: choice of σ

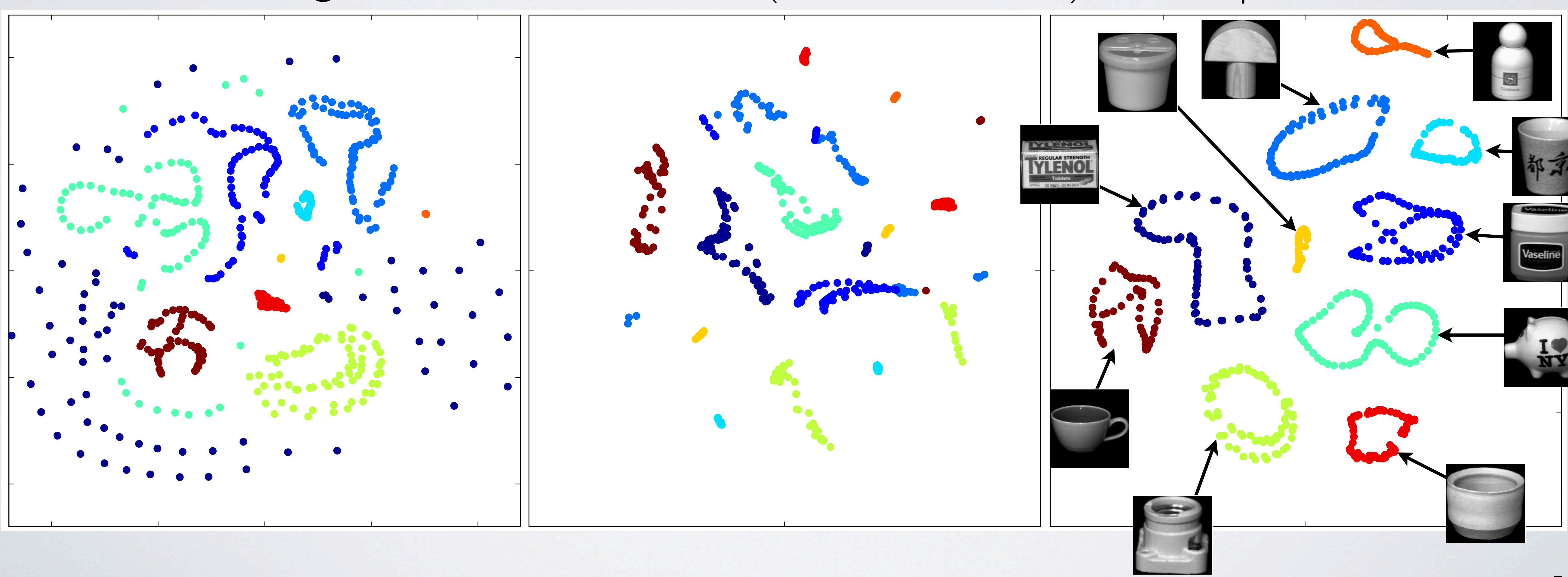
COIL-20: Rotations of objects every 5°; input are greyscale images of 128×128.

Dimensionality Reduction with Elastic Embedding algorithm:

Constant sigma

Rule-of-thumb:
Dist. to the 7th nn (Zelnik & Perona, 05)

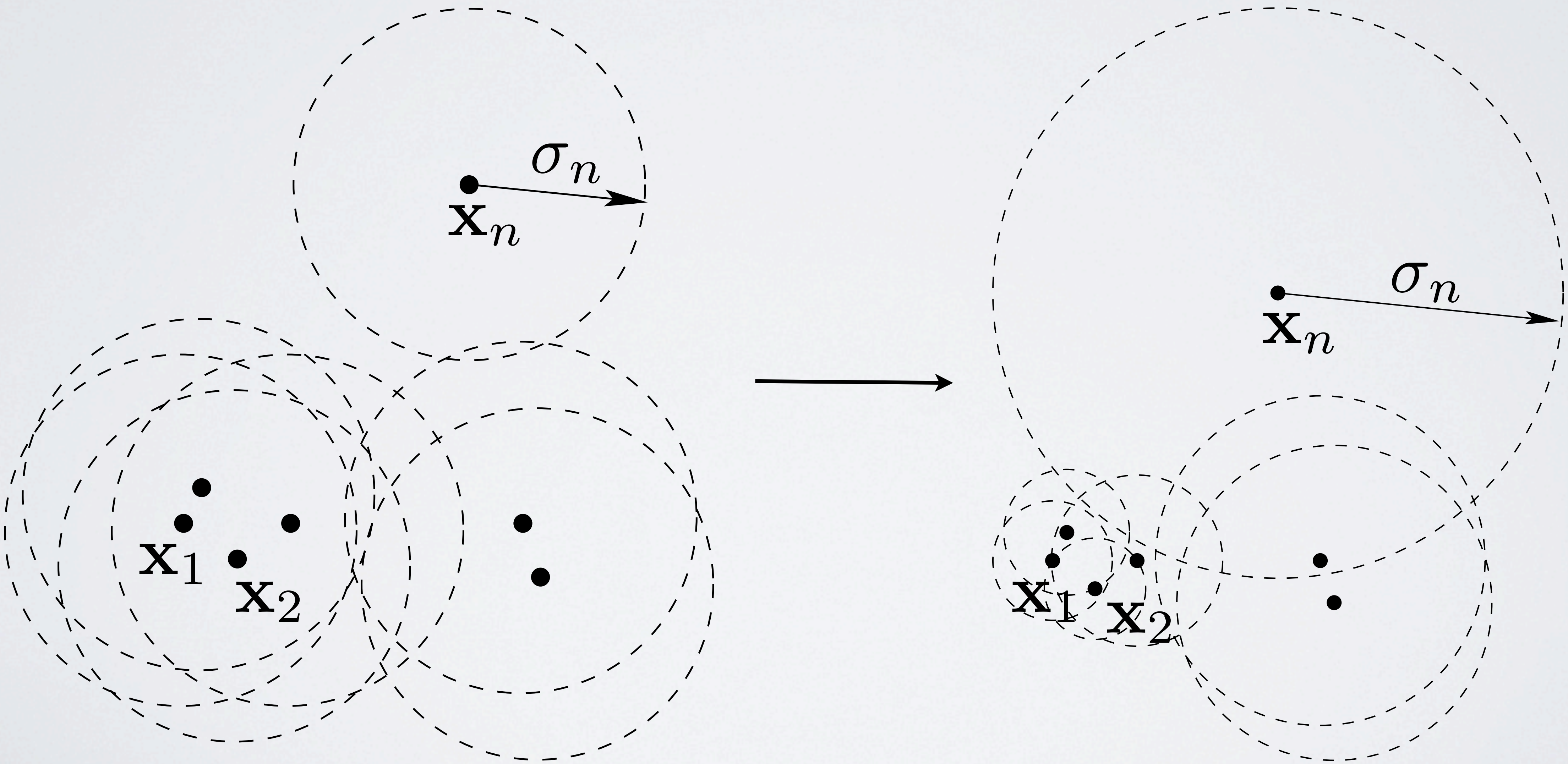
Entropic affinities



Search for good σ

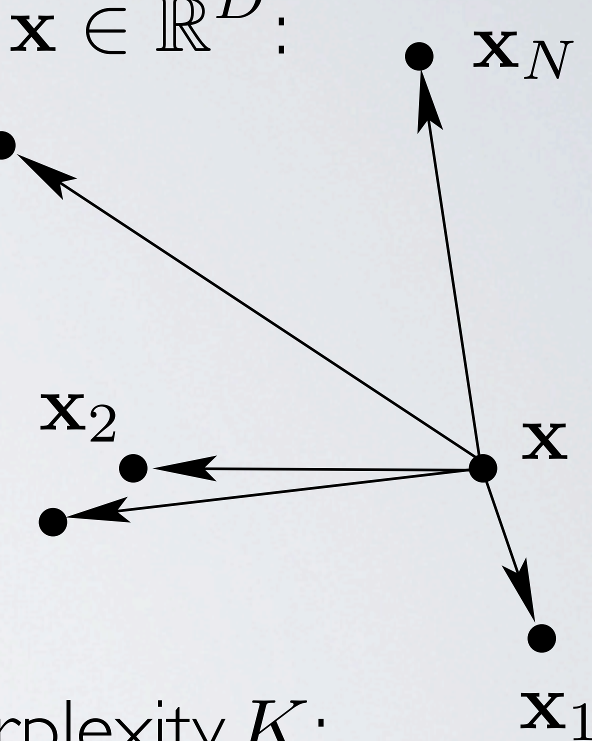
Good σ should be:

- Set separately for every data point.
- Take into account the whole distribution of distances.



Entropic affinities

In the entropic affinities, the σ is set individually for each point such that it has a distribution over neighbors with fixed perplexity K (Hinton & Rowies, 2003).



- Consider a distribution of the neighbors $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ for $\mathbf{x} \in \mathbb{R}^D$:

$$p_n(\mathbf{x}; \sigma) = \frac{K\left(||(\mathbf{x} - \mathbf{x}_n)/\sigma||^2\right)}{\sum_{k=1}^N K\left(||(\mathbf{x} - \mathbf{x}_k)/\sigma||^2\right)}$$

posterior distribution of Kernel Density Estimate.

- The entropy of the distribution is defined as

$$H(\mathbf{x}, \sigma) = -\sum_{n=1}^N p_n(\mathbf{x}, \sigma) \log(p_n(\mathbf{x}, \sigma))$$

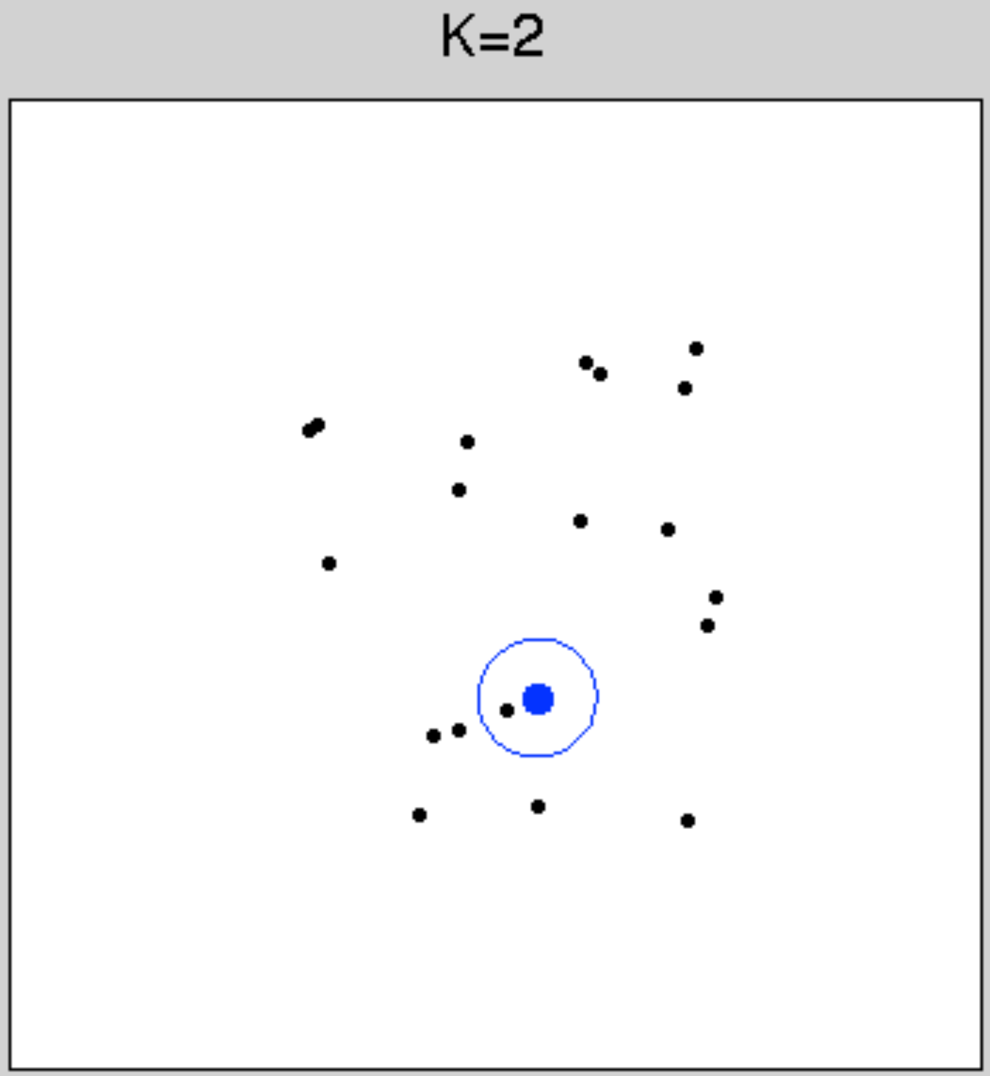
- Consider the bandwidth σ (or precision $\beta = \frac{1}{2\sigma^2}$) given the perplexity K:

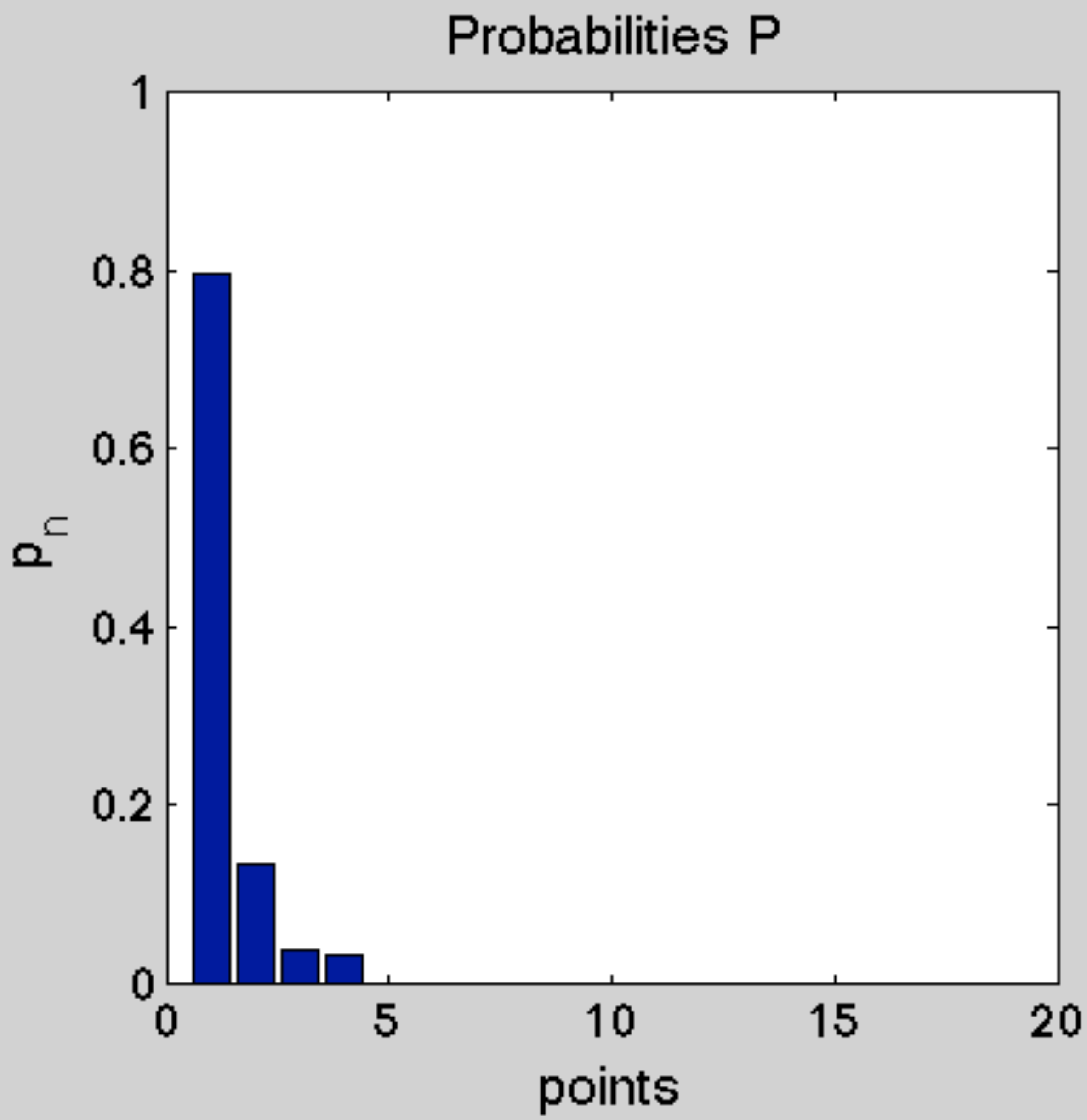
$$H(\mathbf{x}, \beta) = \log K$$

- Perplexity of K in a distribution p over N neighbors provides the same surprise as if we were to choose among K equiprobable neighbors.

- We define entropic affinities as probabilities $\mathbf{p} = (p_1, \ldots, p_N)$ for $\mathbf{x}$ with respect to β. Thos affinities define a random walk matrix.

Entropic affinities: example

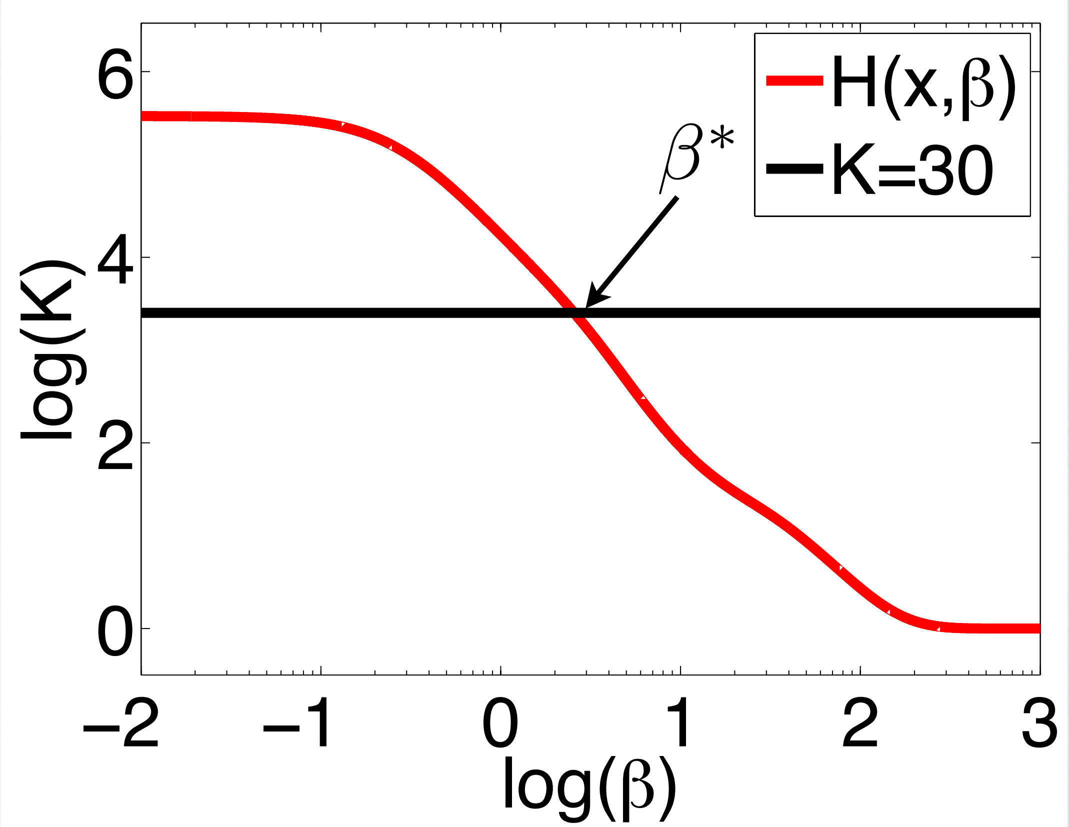




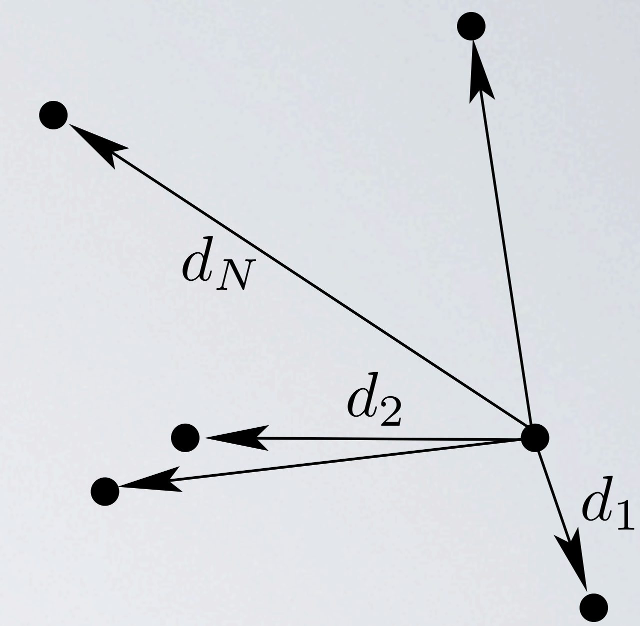
Entropic affinities: properties

$$H(\mathbf{x}_n, \beta_n) \equiv -\sum_{n=1}^{N} p_n(\mathbf{x}_n, \beta_n) \log(p_n(\mathbf{x}_n, \beta_n)) = \log K$$

- This is a $1D$ root-finding problem or an inversion problem $\beta_n = H_{x_n}^{-1}(\log K)$.
- Should be solved for $\mathbf{x}_n \in \mathbf{x}_1, \dots, \mathbf{x}_N$
- We can prove that:
  - The root-finding problem is well defined for a Gaussian kernel for any $\beta_n > 0$, and has a unique root for any $K \in (0, N)$.
  - The inverse is a uniquely defined continuously differentiable function for all $\mathbf{x}_n \in \mathbb{R}^N$ and $K \in (0, N)$.



Entropic affinities: bounds



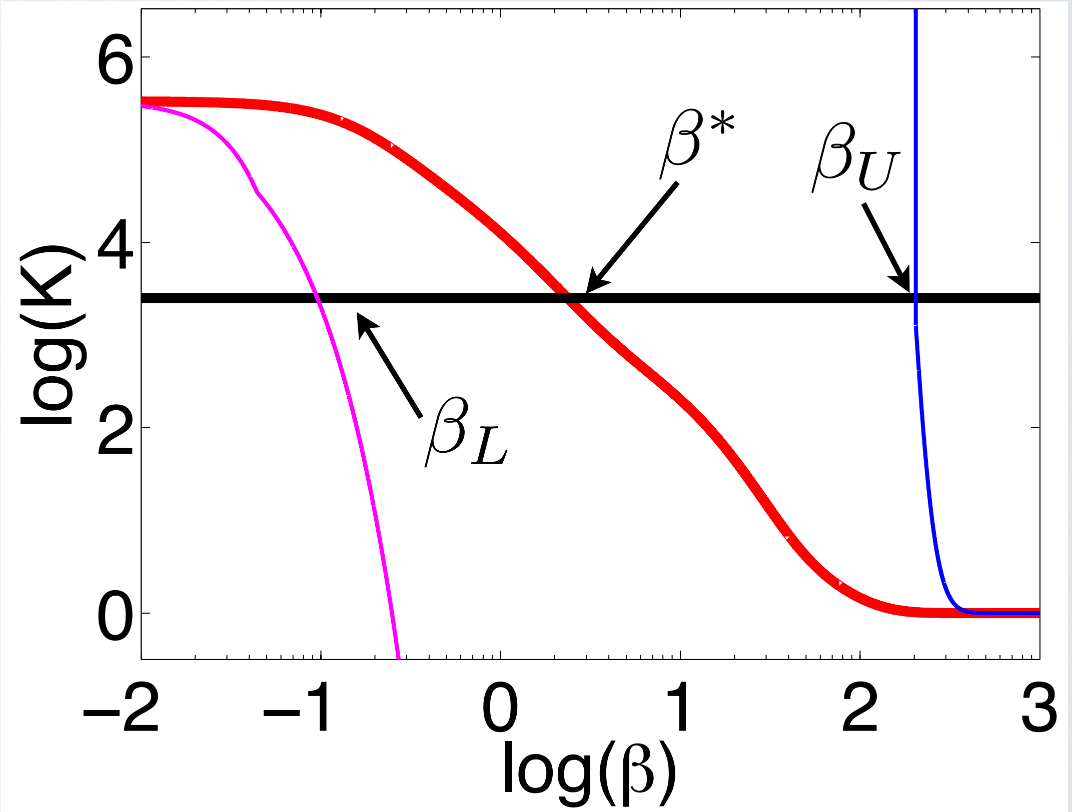
The bounds $[\beta_L, \beta_U]$ for every $K \in (0, N)$ and $\mathbf{x}_n \in \mathbb{R}^N$:

$$\beta_L = \max\left(\frac{N \log \frac{N}{K}}{(N-1)\Delta_N^2}, \sqrt{\frac{\log \frac{N}{K}}{d_N^4 - d_1^4}} \right),$$

$$\beta_U = \frac{1}{\Delta_2^2} \log\left(\frac{p_1}{1-p_1}(N-1) \right),$$

where $\Delta_2^2 = d_2^2 - d_1^2$, $\Delta_N^2 = d_N^2 - d_1^2$, and p_1 is a unique solution of the equation

$$2(1-p_1)\log \frac{N}{2(1-p_1)} = \log\left(\min(\sqrt{2N}, K)\right)$$



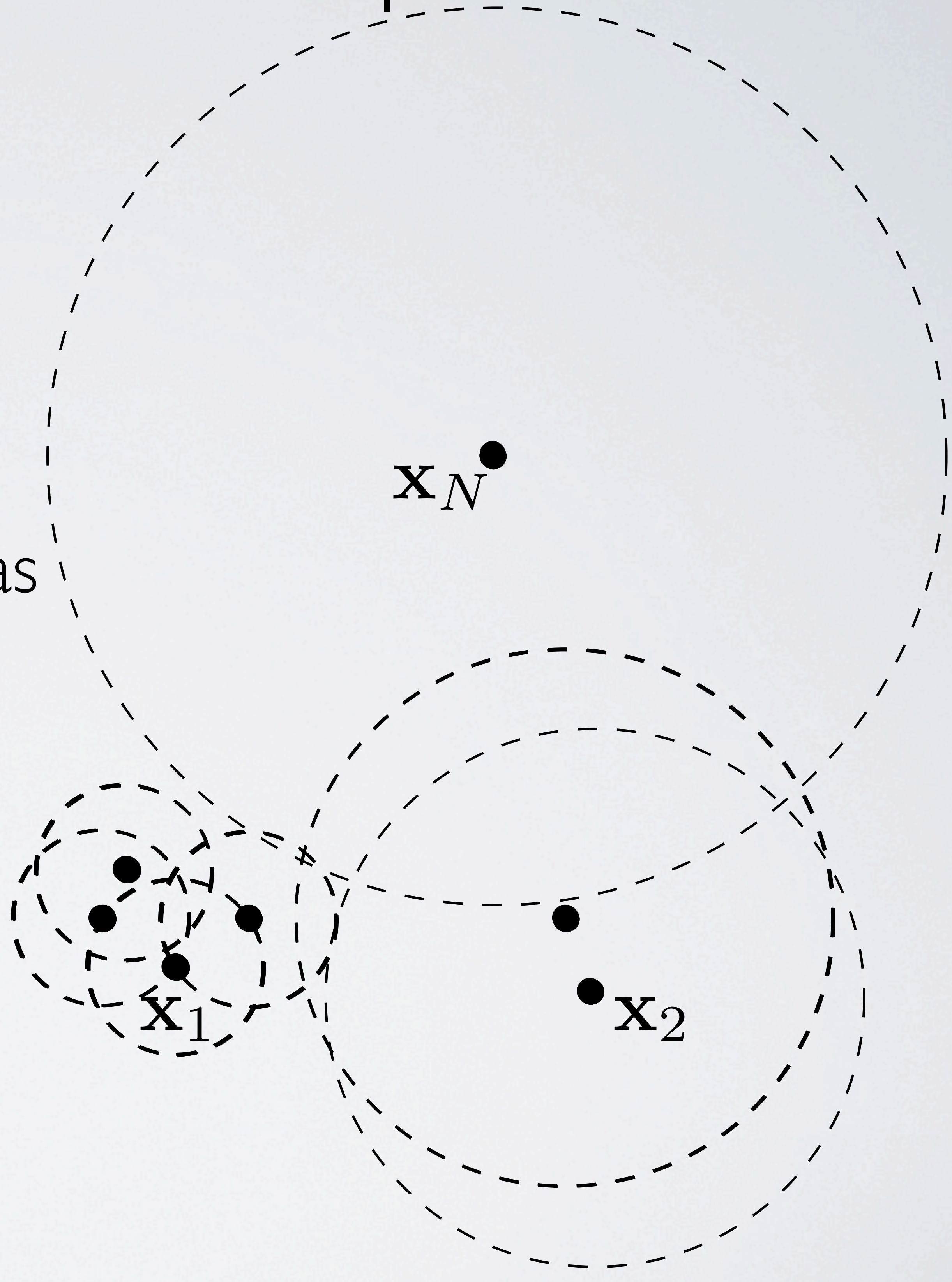
The bounds are computed in $\mathcal{O}(1)$ for each point.

Entropic affinities: computation

For every $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

$$H(\mathbf{x}_n, \beta_n) = \log K$$

1. Initialize β_n as close to the root as possible.
2. Compute the root β_n.



1. Computation of β_n; the root-finding

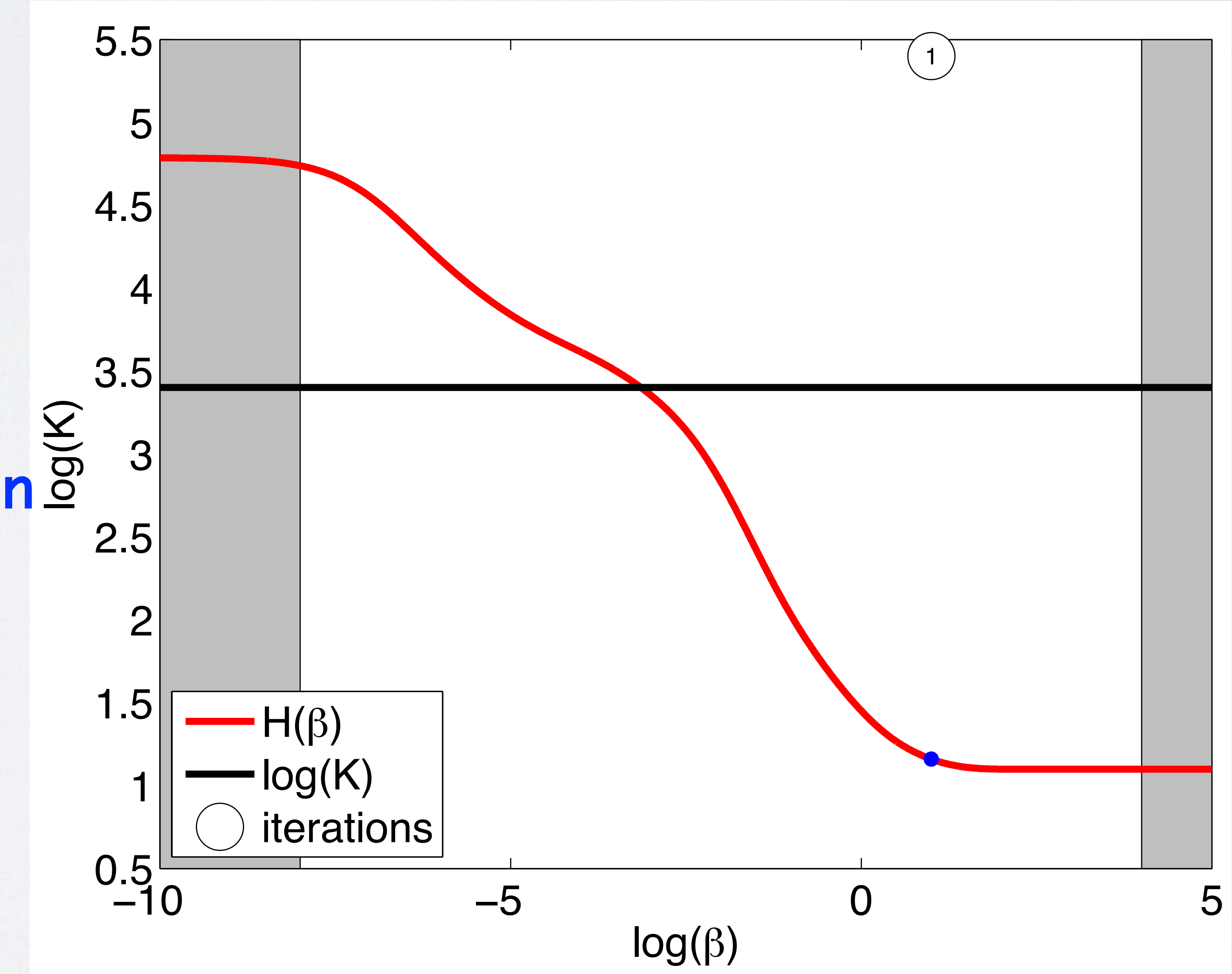
Methods		Convergence order	Derivatives order	Number of $\mathcal{O}(N)$ evaluations
Derivative-free	Bisection	linear	0	1
	Brent	linear	0	1
	Ridder	quadratic	0	2
Derivative-based	Newton	quadratic	1	2
	Halley	cubic	2	3
	Euler	cubic	2	3

- The cost of the objective function evaluation and each of derivative is $\mathcal{O}(N)$.
- Derivative-free methods above generally converge globally. They work by iteratively shrinking an interval bracketing the root.
- Derivative-based methods have higher convergence order, but may diverge.

Robustified root-finding algorithm

- We embed the derivative-based algorithm into bisection loop for global convergence.
- We run the following algorithm for each $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

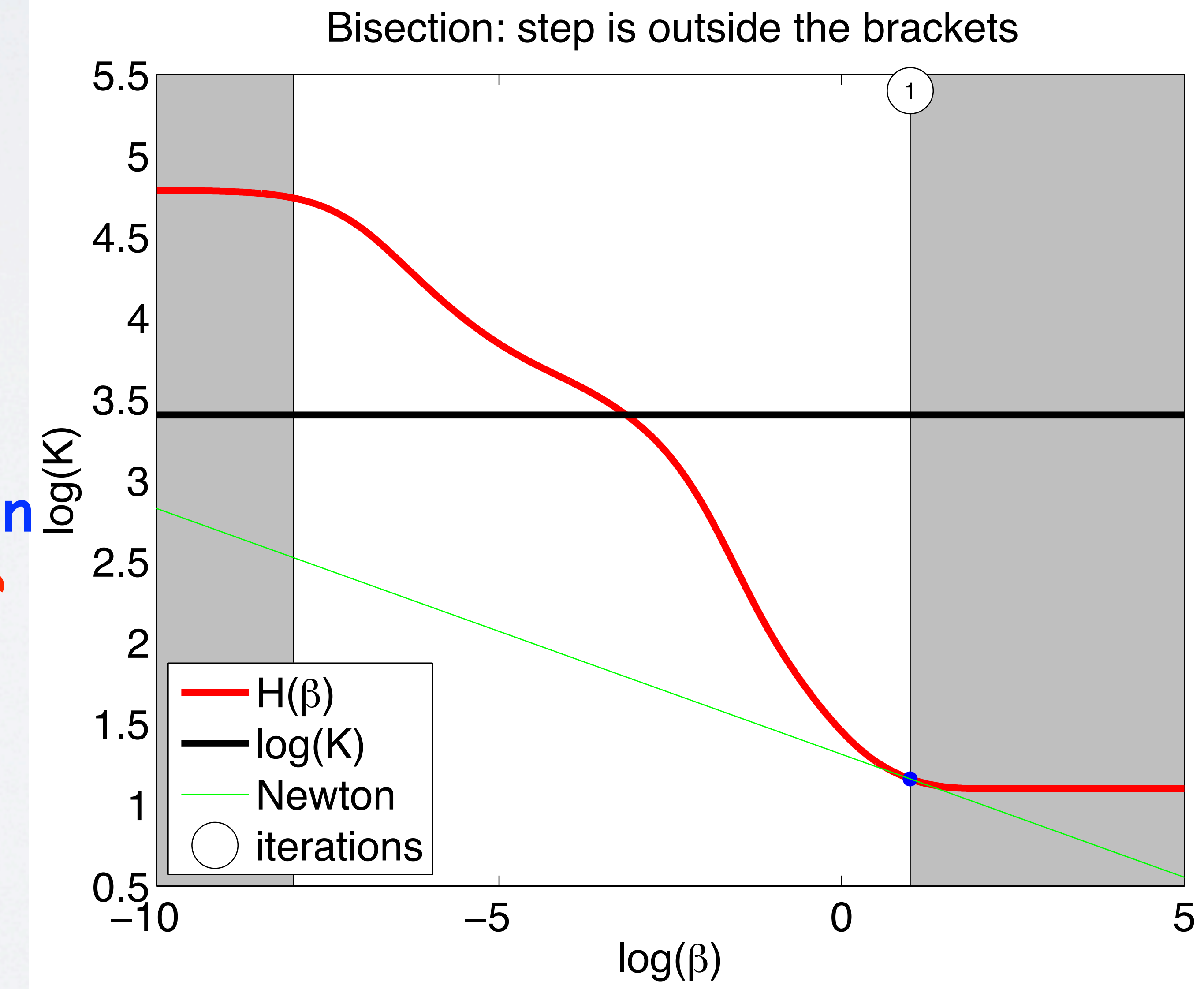
```
Input: initial β, perplexity K,
distances d_1^2, ..., d_N^2, bounds 𝓑.
while true do
  for k = 1 to maxit do
    compute β using a derivative-
    based method
      if tolerance achieved return
      if β ∉ 𝓑 exit for loop
      update 𝓑
  end for
  compute β using bisection
  update 𝓑
end while
```



Robustified root-finding algorithm

- We embed the derivative-based algorithm into bisection loop for global convergence
- We run the following algorithm for each $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

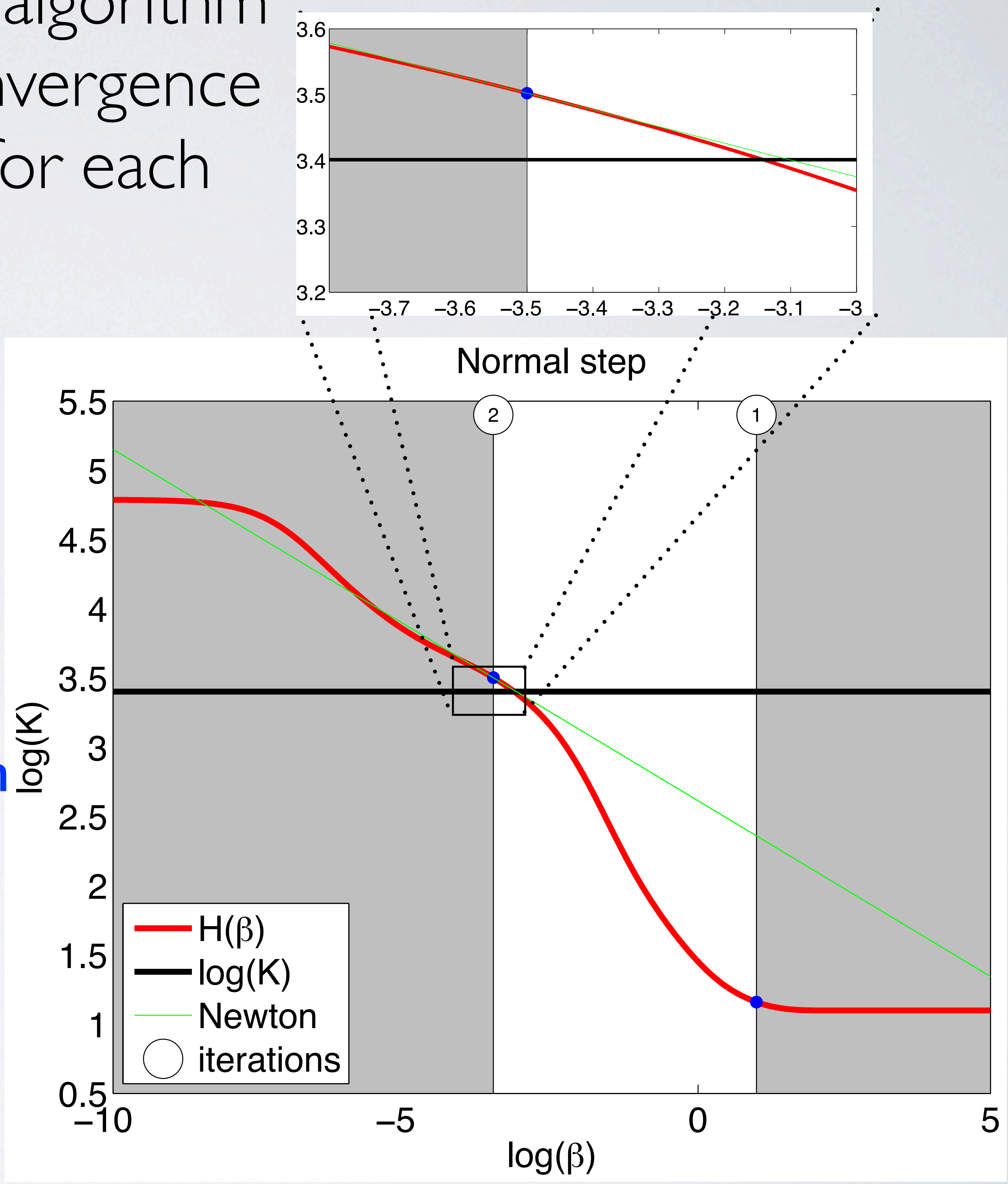
```
Input: initial β, perplexity K,
distances d_1^2, ..., d_N^2, bounds 𝓑.
while true do
  for k = 1 to maxit do
    compute β using a derivative-
    based method
      if tolerance achieved return
      if β ∉ 𝓑 exit for loop
      update 𝓑
  end for
  compute β using bisection
  update 𝓑
end while
```



Robustified root-finding algorithm

- We embed the derivative-based algorithm into bisection loop for global convergence
- We run the following algorithm for each $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

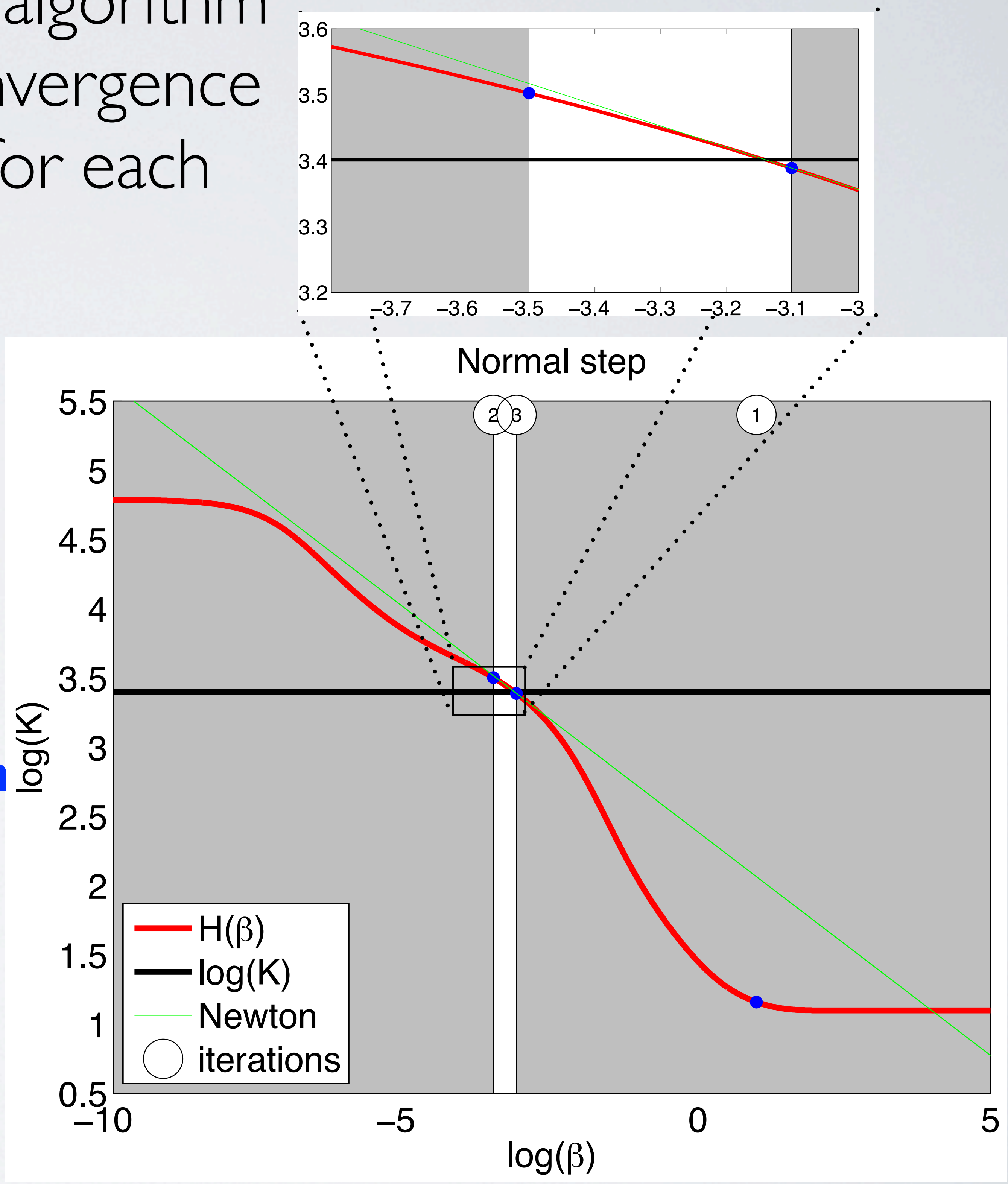
Input: initial β, perplexity K, distances $d_1^2, \ldots, d_N^2$, bounds $\mathcal{B}$.
while true **do**
  for $k = 1$ **to** maxit **do**
    compute β using a derivative-based method
      if tolerance achieved **return**
      if $\beta \notin \mathcal{B}$ **exit for loop**
      update $\mathcal{B}$
  end for
  compute β using bisection
  update $\mathcal{B}$
end while



Robustified root-finding algorithm

- We embed the derivative-based algorithm into bisection loop for global convergence
- We run the following algorithm for each $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

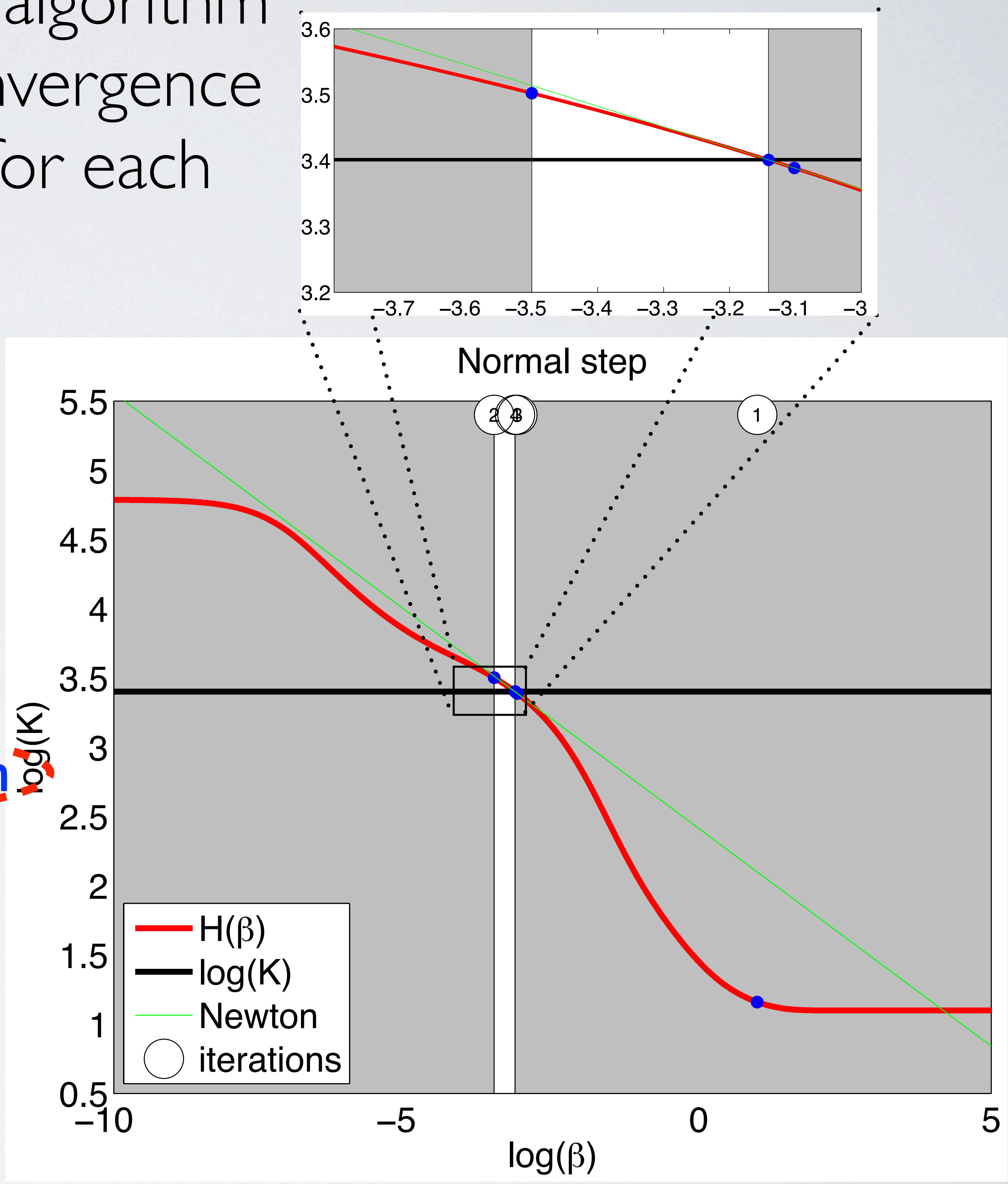
```
Input: initial β, perplexity K,
distances d_1^2, ..., d_N^2, bounds B.
while true do
  for k = 1 to maxit do
    compute β using a derivative-
    based method
      if tolerance achieved return
      if β ∉ B exit for loop
      update B
  end for
  compute β using bisection
  update B
end while
```



Robustified root-finding algorithm

- We embed the derivative-based algorithm into bisection loop for global convergence
- We run the following algorithm for each $\mathbf{x}_n \in \mathbf{x}_1, \ldots, \mathbf{x}_N$

```
Input: initial β, perplexity K,
distances d_1^2, ..., d_N^2, bounds B.
while true do
  for k = 1 to maxit do
    compute β using a derivative-
    based method
     if tolerance achieved return
     if β ∉ B exit for loop
     update B
  end for
  compute β using bisection
  update B
end while
```



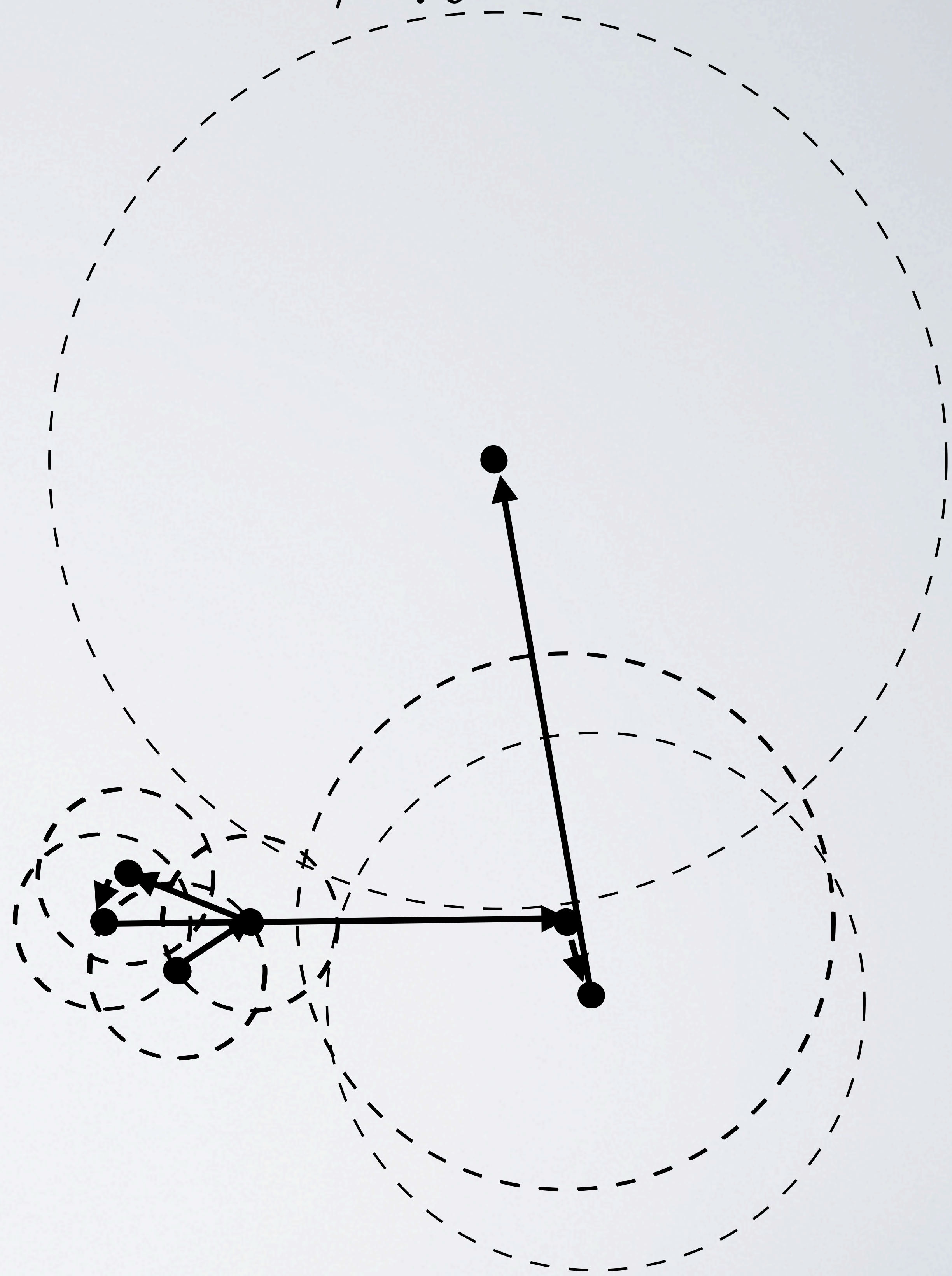
2. Initialization of β_n

1. Simple initialization:
   - midpoint of the bounds,
   - distance to kth nearest neighbor.

   Typically far from root and require more iterations.

2. Each new β_n is initialized from the solution to its predecessor:
   - sequential order;
   - tree order.

   We need to find orders that are correlated with the behavior of β.



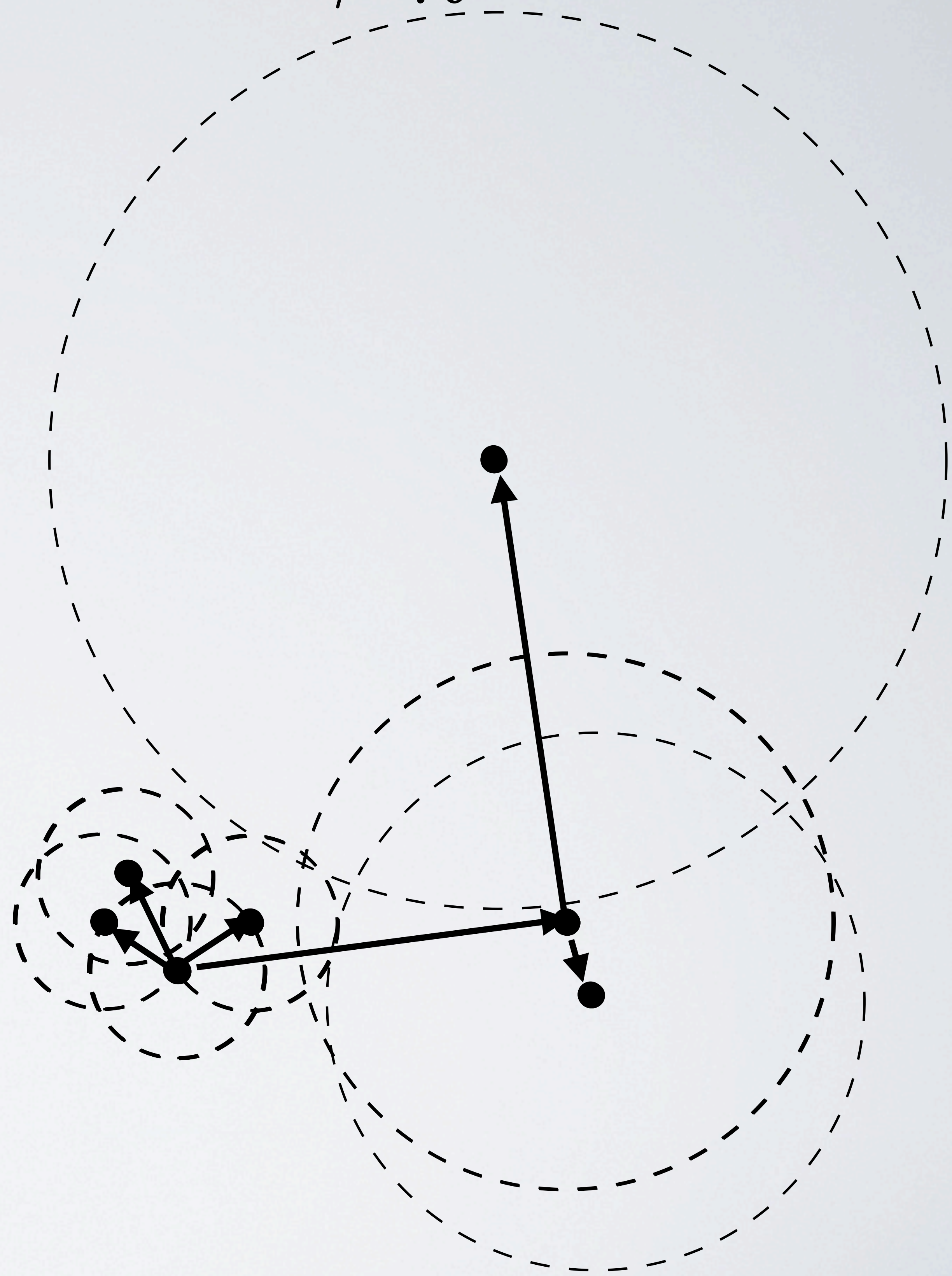
2. Initialization of β_n

1. Simple initialization:
   - middle of the bounds,
   - distance to kth nearest neighbor.

   Typically far from root and require more iterations.

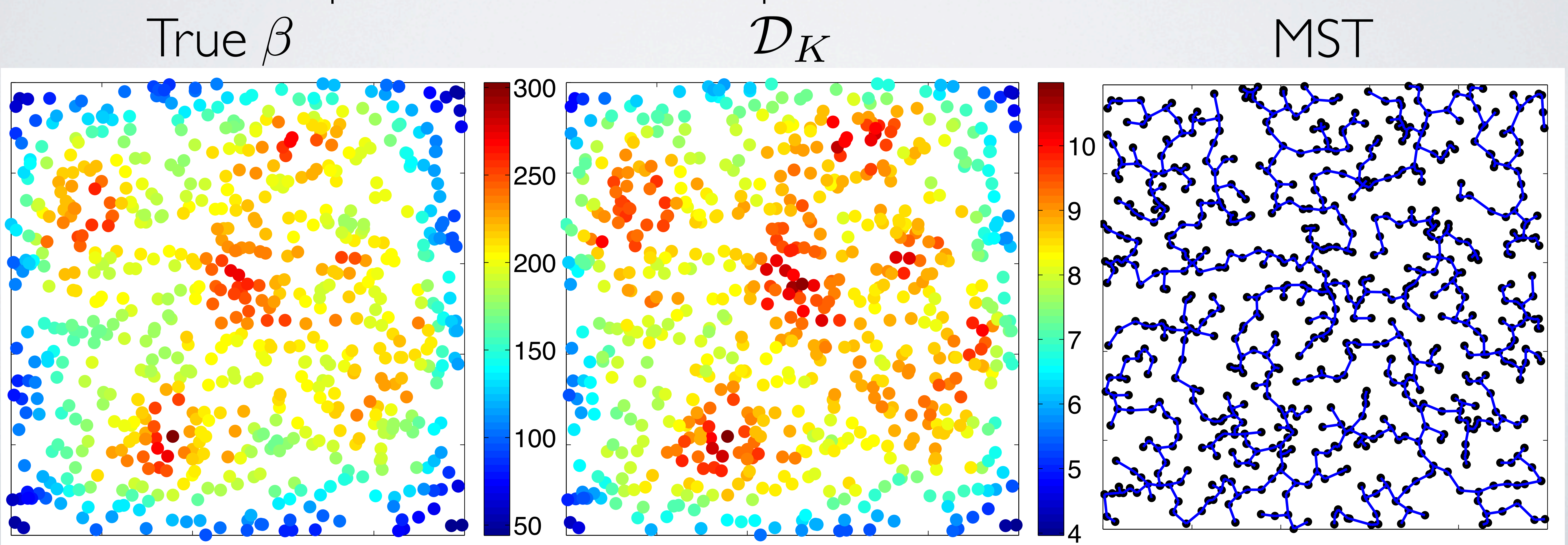
2. Each new β_n is initialized from the solution to its predecessor:
   - sequential order;
   - tree order.

   We need to find orders that are correlated with the behavior of β.



Sequential or tree order

- $\mathcal{D}_k$, *density* strategy: for the fixed entropy value, β is larger in dense regions and smaller in sparser ones.
    - Use nearest neighbor density estimate.
    - β_n is proportional to the distance to kth nearest neighbor of $\mathbf{x}_n$.
- MST, *local* strategy: nearby points have similar β values.
    - Build a MST around the data.
    - Process the points in level-order, so parents are solved for before children.



Experimental evaluation: setup

We set the perplexity to $K = 30$ and the tolerance to 10^{-10}.

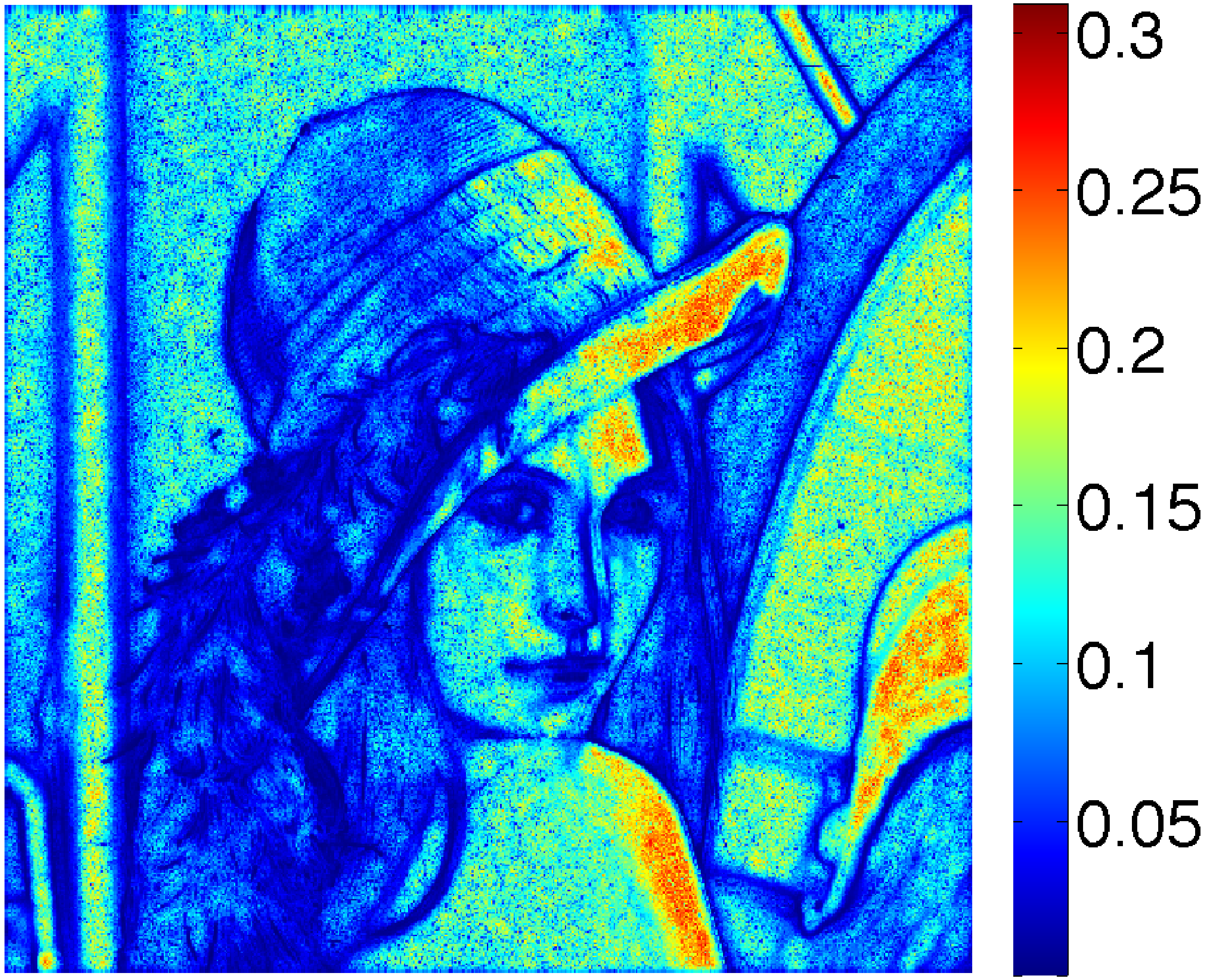
Initializations:

- "oracle": processes the points in the order of their true β values,
- MST: local-based order,
- $\mathcal{D}_K$: density-based order,
- bounds: initialize from the midpoint of the bounds,
- random: initialize from one of $\mathbf{x}_n$ chosen at random.

Root-finding methods:

- Derivative-free: Bisection, Brent, Ridder.
- Derivative-based: Newton, Euler, Halley.

Experimental evaluation: Lena



Bisection: > 10 min.

Our method: 1 min.

Computing just the affinities given βs: 20 s.

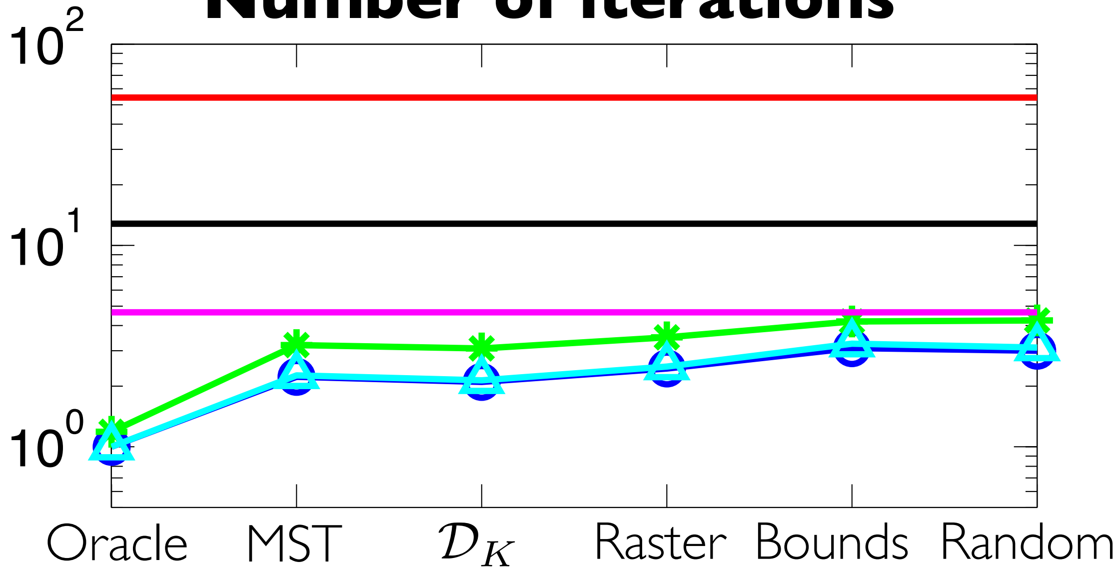
Experimental evaluation: image

512×512 Lena image. Each data point is a pixel represented by spatial and range features$(i, j, L, u, v) \in \mathbb{R}^5$:

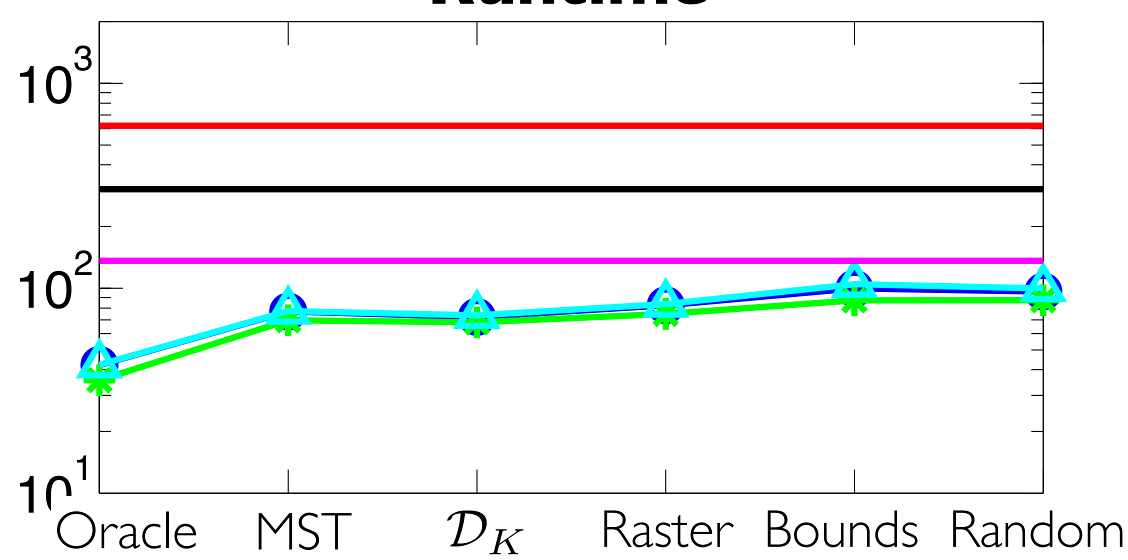
- (i, j) is the pixel location;
- (L, u, v) the pixel value.

$N = 262\,144$ points, $D = 5$ dimensions

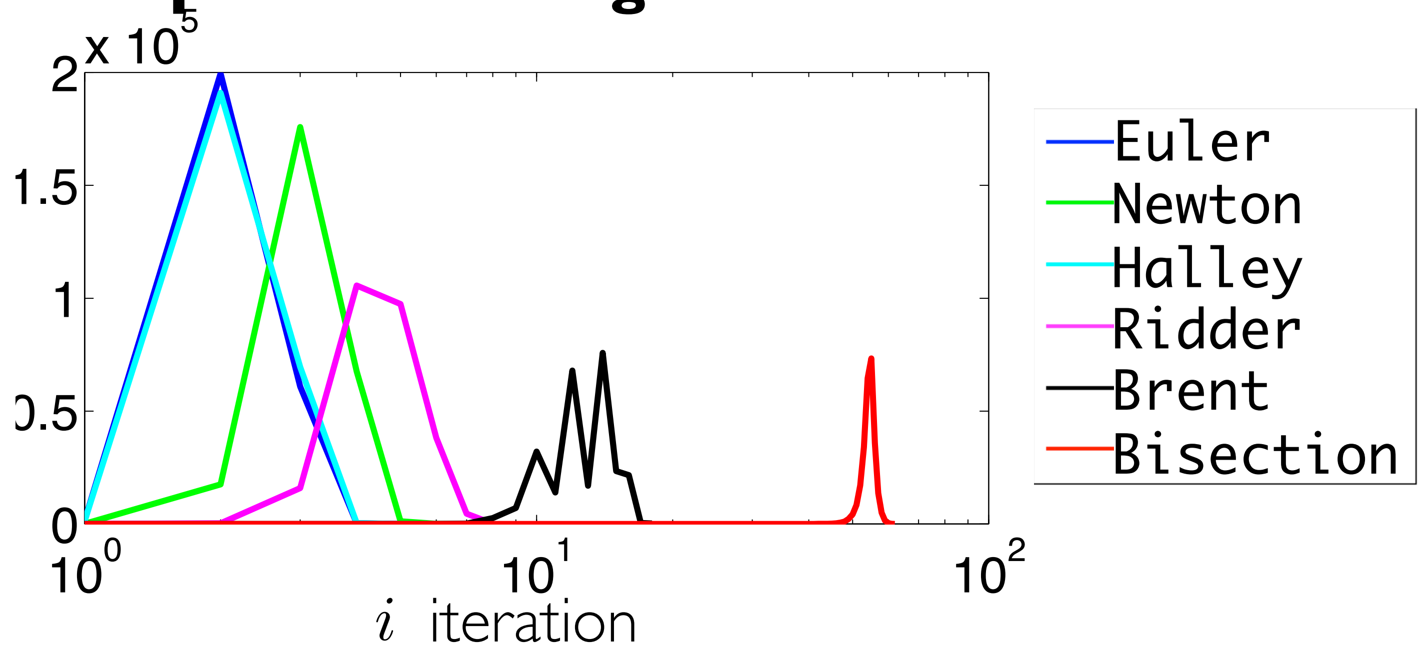
Number of iterations



Runtime



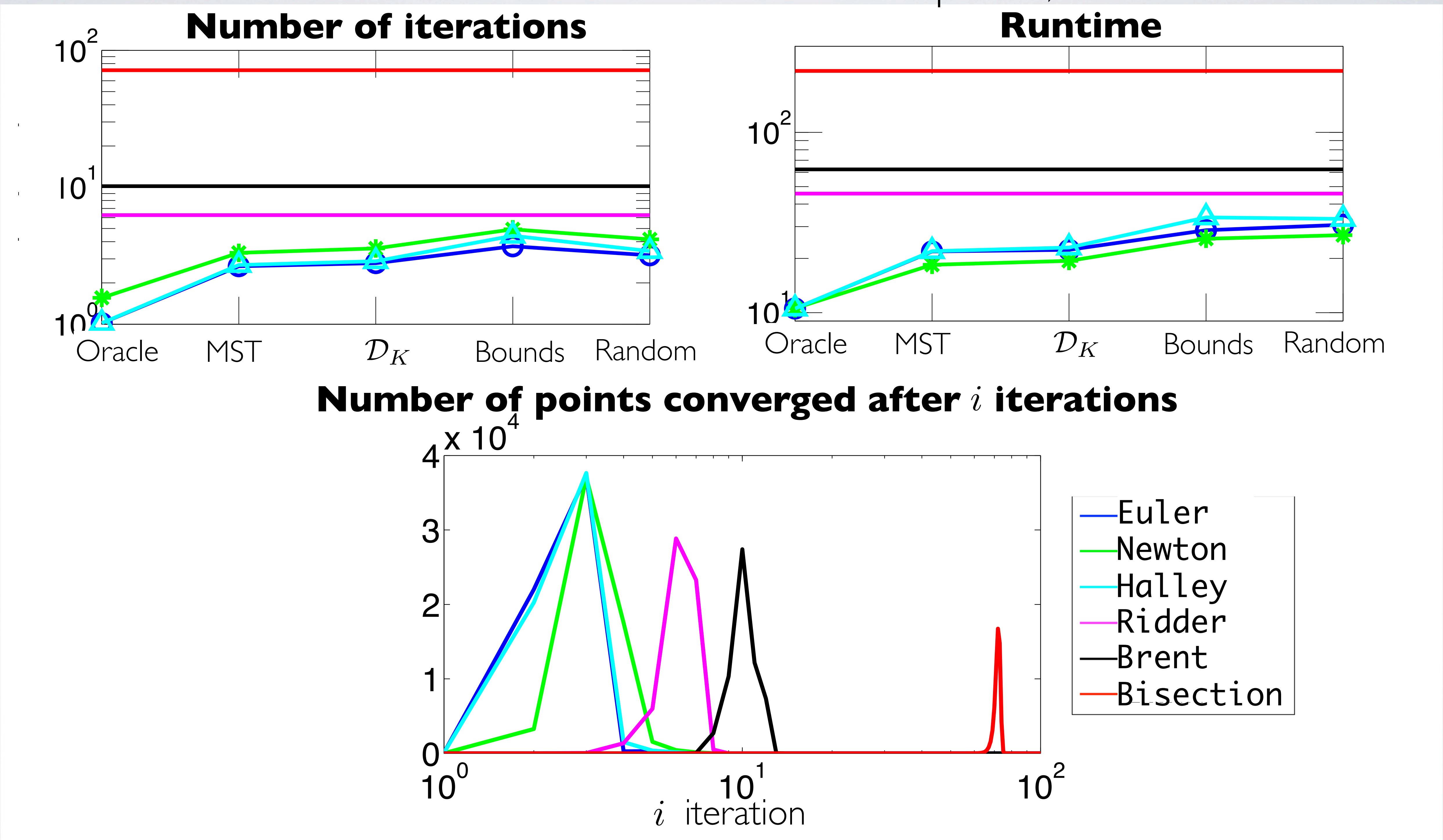
Number of points converged after i iterations



Experimental evaluation: digits

$60\,000$ handwritten digits from the MNIST dataset. Each datapoint is a 28×28 grayscale image.

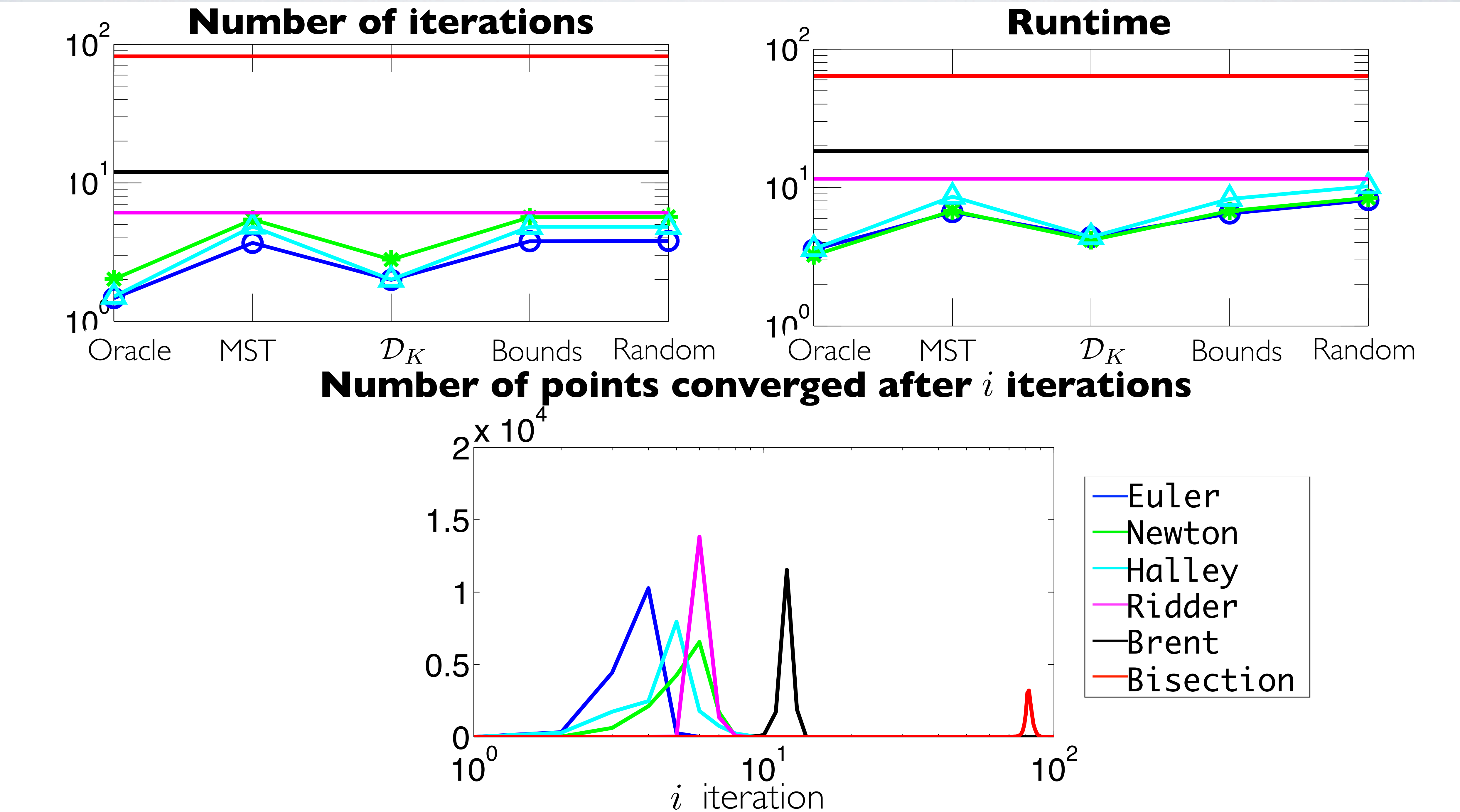
$N = 60\,000$ points, $D = 784$ dimensions



Experimental evaluation: text

Articles from Grolier's encyclopedia. Each point is a word count of the most popular $15\,275$ words from $30\,991$ articles.

$N = 30\,991$ points, $D = 15\,275$ dimensions



Conclusions

- We studied the behavior of entropic affinities and their properties.
- Search for the affinities involves finding the root of non-linear equation.
- We can find the root almost to machine precision in just over one iteration per point on average using:
  - bounds for the root,
  - root-finding methods with high-order convergence,
  - warm-start initialization based on local or density orders.
- In applications such as spectral clustering and embeddings, semi-supervised learning using entropic affinities should give better results than fixing the bandwidth to a single value or using a rule-of-thumb.
- The only user parameter is the global perplexity value K.
- MATLAB code online at http://eecs.ucmerced.edu. Run it simply like `[W,s] = ea(X,K)`.