Fast optimal bandwidth selection for kernel density estimation

Vikas C. Raykar and Ramani Duraiswami
University of Maryland, CollegePark
{vikas,ramani}@cs.umd.edu

2006 SIAM Conference on Data Mining
Bethesda, April 20, 2006

Kernel methods

- In many kernel methods the computational bottleneck is to compute a weighted sum of the form

$$f(x) = \sum_{i=1}^{N} \alpha_i K_h(x, x_i)$$

- Computing $f(x)$ at M points is of complexity $\mathcal{O}(MN)$.

- Fast Gauss Transform [Greengard and Strain 1991] reduced complexity to $\mathcal{O}(p^d(M+N))$ and is effective in low dimensions ($d \leq 3$).

- Improved Fast Gauss Transform [Yang et. al. 2003] reduced complexity to $\mathcal{O}(d^p(M+N))$, which scales better with d.

Hyperparameter selection for kernel methods

- Most kernel methods require choosing some parameter (e.g. bandwidth h of the kernel).

- Optimal procedures to choose these parameters are iterative with each iteration costing $\mathcal{O}(N^2)$.

- Here we show how to accelerate the state-of-the-art method [Sheather and Jones, 1991] for bandwidth selection for KDE.

Kernel density estimation

- The most popular method for density estimation is the kernel density estimator (KDE).

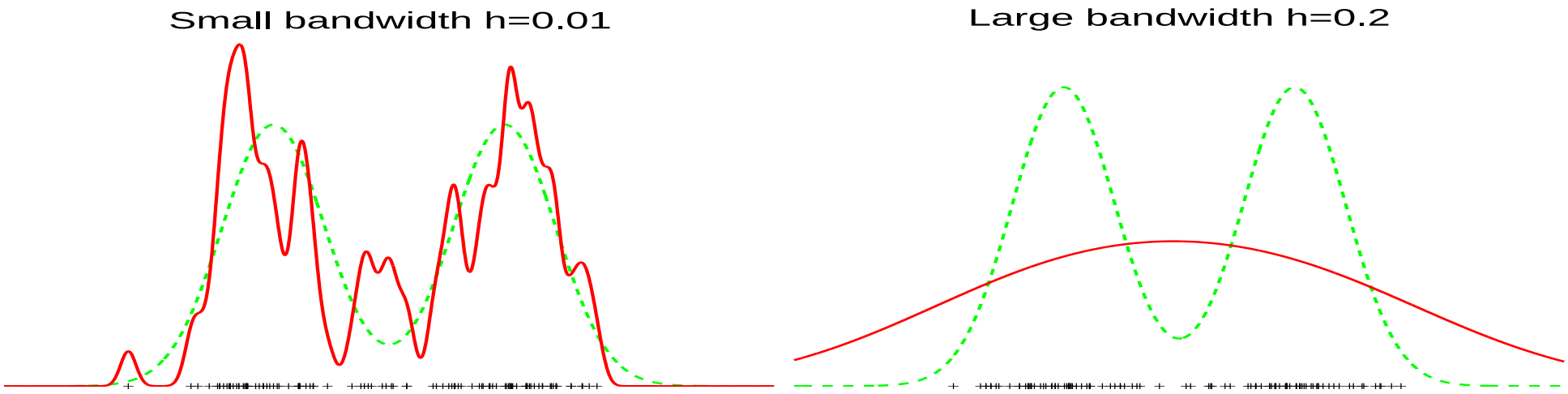
$$\widehat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h} K\left(\frac{x - x_i}{h}\right)$$

- **Efficient use of KDE requires choosing h optimally.**

The bandwidth h is a very crucial parameter

- As **h decreases towards 0**, the number of modes increases to the number of data points and the KDE is very noisy.

- As **h increases towards ∞**, the number of modes drops to 1, so that any interesting structure has been smeared away and the KDE just displays a unimodal pattern.



Gist of the paper

- Optimal bandwidth selection for kernel density estimation scales as $\mathcal{O}(N^2)$.

- We present a fast computational technique that scales as $\mathcal{O}(N)$.

Fast kernel density derivative estimation

- The core part is a fast $\epsilon - exact$ algorithm for **kernel density derivative estimation** which reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

- For example for $N = 409,600$ points.
  - Direct evaluation $\rightarrow$ 12.76 **hours**.
  - Fast evaluation $\rightarrow$ 65 **seconds** with an error of around 10^{-12}.

Speedup for bandwidth estimation

	h_{direct}	h_{fast}	T_{direct} (sec)	T_{fast} (sec)	Speedup	Rel. Err.
1	0.122213	0.122215	4182.29	64.28	65.06	1.37e-005
2	0.082591	0.082592	5061.42	77.30	65.48	1.38e-005
3	0.020543	0.020543	8523.26	101.62	83.87	1.53e-006
4	0.020621	0.020621	7825.72	105.88	73.91	1.81e-006
5	0.012881	0.012881	6543.52	91.11	71.82	5.34e-006
6	0.098301	0.098303	5023.06	76.18	65.93	1.62e-005
7	0.092240	0.092240	5918.19	88.61	66.79	6.34e-006
8	0.074698	0.074699	5912.97	90.74	65.16	1.40e-005
9	0.081301	0.081302	6440.66	89.91	71.63	1.17e-005
10	0.024326	0.024326	7186.07	106.17	67.69	1.84e-006
11	0.086831	0.086832	5912.23	90.45	65.36	1.71e-005
12	0.032492	0.032493	8310.90	119.02	69.83	3.83e-006
13	0.045797	0.045797	6824.59	104.79	65.13	4.41e-006
14	0.027573	0.027573	10485.48	111.54	94.01	1.18e-006
15	0.023096	0.023096	11797.34	112.57	104.80	7.05e-007

Speedup for projection pursuit

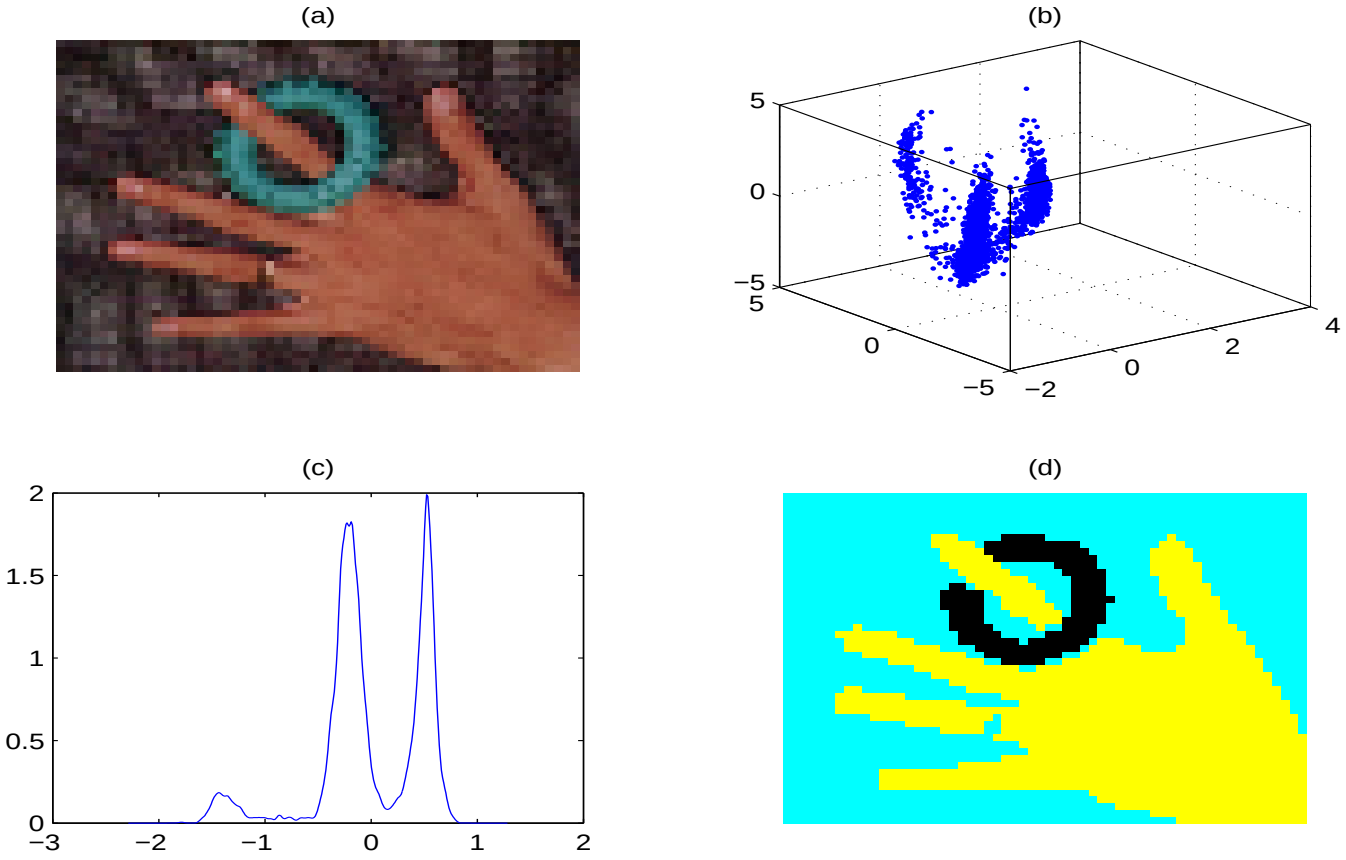


Image segmentation via PP with optimal KDE took 15 minutes while that using the direct method takes around 7.5 hours.

Software

- The code is available for academic use.

- `http://www.cs.umd.edu/~vikas/code/optimal_bw/optimal_bw_code.htm`