

# Stat 992: Lecture 04

## Numerical implementation of kernel smoothing.

Moo K. Chung mchung@stat.wisc.edu

January 28, 2004

1. *Nadaraya-Watson estimator.* In general, it is cumbersome to manipulate kernels with infinite support analytically and numerically. It is especially true when one tries to implement a computer algorithm although there is a way around it using the Fourier transform and its inverse transform (see Problem 8). So we usually truncate kernel  $K_H$  in  $\Omega^c$ . We define truncated and normalized kernel

$$\tilde{K}_H(t) = \frac{K_H(t)\mathbf{1}_\Omega(t)}{\int_\Omega K_H(t) dt}, \quad (1)$$

where  $\mathbf{1}_\Omega$  is an indicator function that gives zero everywhere except  $\Omega$  where it gives the value one. Then the kernel smoother is given by

$$\hat{\mu}(t) = \tilde{K}_H * Y(t).$$

Even though functional signal  $Y$  is continuous, it will be observed at discrete points  $t_i$  so the signal can be written as

$$Y(t) = \sum_{i=1}^n Y(t_i)\delta(t - t_i). \quad (2)$$

Then from the property of the Dirac-delta function,

$$\hat{\mu}(t) = \sum_{i=1}^n \tilde{K}_H(t - t_i)Y(t_i)$$

where  $\tilde{K}_H$  is now a discrete normalized version

$$\tilde{K}_H(t - t_i) = \frac{K_H(t - t_i)}{\sum_{t_j \in \Omega} K_H(t - t_j)}. \quad (3)$$

Although we used the same notation, kernel (3) is the discrete version of continuous version (1). The discrete normalized kernel estimator in nonparametric regression setting was first developed by Nadaraya (1964) and Watson (1964) so it is usually called Nadaraya-Watson (NW) estimator.

2. *Implementation.* We will show 2D isotropic version when  $H = \sigma I_n$ .  $\sigma$  is the bandwidth parameter that controls the amount of smoothing. The general anisotropic version should be trivial. Suppose that noisy observations are obtained at each point  $(\frac{i}{100}, \frac{j}{100}) \in [0, 1]^2$ . The *unknown* signal is assumed to be

$$\mu(t_1, t_2) = \cos(10t_1) + \sin(10t_2).$$

For simplicity, we assume unrealistic white noise error  $N(0, 0.4^2)$  for now and simulate a synthetic image.

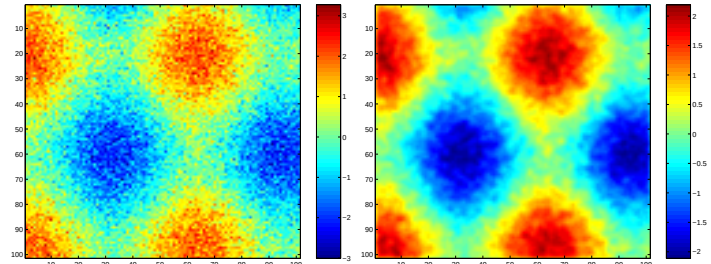


Figure 1: Left: observations  $Y$ . Right: estimation  $\hat{\mu}$ . The estimated signal doesn't seem smooth enough indicating more smoothing is needed.

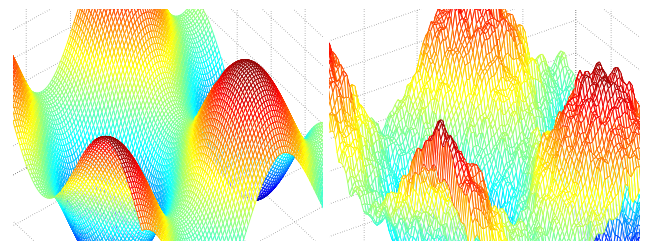


Figure 2: Left: unknown 2D signal  $\mu$ . Right: estimation  $\hat{\mu}$ . The estimated function is not smooth enough.

```
[px,py] = meshgrid([0:0.01:1]);
>>px(1:3,1:3)
ans =
```

```
0      0.0100    0.0200
0      0.0100    0.0200
0      0.0100    0.0200
```

```
>> py(1:3,1:3)
ans =
```

```
0      0      0
0.0100    0.0100    0.0100
0.0200    0.0200    0.0200
```

```
mu=cos(10*px)+sin(8*py);
mesh(mu);
figure; imagesc(mu); colorbar;
```

```
e=normrnd(0,0.4,101,101); Y=mu+e;
```

```
K=inline('exp(-(x.^2+y.^2)/2/sig^2)');
>>K
```

Inline function:

```
K(sig,x,y) = exp(-(x.^2+y.^2)/2/sig^2)
```

In defining kernel in MATLAB, The normalizing constant  $2\pi\sigma^2$  has been neglected since it will cancel out later.

```
[dx,dy]=meshgrid([-2:2]);
weight=K(0.5,dx,dy)/
        sum(sum(K(0.5,dx,dy)));
>>weight
0.0000 0.0000 0.0002 0.0000 0.0000
0.0000 0.0113 0.0837 0.0113 0.0000
0.0002 0.0837 0.6187 0.0837 0.0002
0.0000 0.0113 0.0837 0.0113 0.0000
0.0000 0.0000 0.0002 0.0000 0.0000

weight=K(1,dx,dy)/sum(sum(K(1,dx,dy)));
>>weight =
0.0030 0.0133 0.0219 0.0133 0.0030
0.0133 0.0596 0.0983 0.0596 0.0133
0.0219 0.0983 0.1621 0.0983 0.0219
0.0133 0.0596 0.0983 0.0596 0.0133
0.0030 0.0133 0.0219 0.0133 0.0030
smooth_Y=conv2(Y,weight,'same');
```

**Problem 7.** Write a MATLAB function that does 2D anisotropic kernel smoothing. Apply your function to the following image <http://www.stat.wisc.edu/~mchung/teaching/data/2Dbrain/sagittal.data> with different bandwidth matrix  $H$ . This is the sagittal cross section of the brain. Use command `load sagittal.data` to load data into MATLAB.

```
imagesc(brainslice);
colormap('bone');
colorbar;
```

**Problem 8.** Using the Fourier transform and its inverse transform, implement 2D Gaussian kernel smoothing in computer. Whenever you submit computer codes, please make it as a function clearly documenting what are inputs and outputs of the function. Hint: In MATLAB, you can perform *finite fourier transform*. Read any ODE book. Possible project topic if no student solve Problem 8 as a homework. For project do it on 3D and investigate how to implement anisotropic smoothing. Compare the performance of different Gaussian smoothing methods.

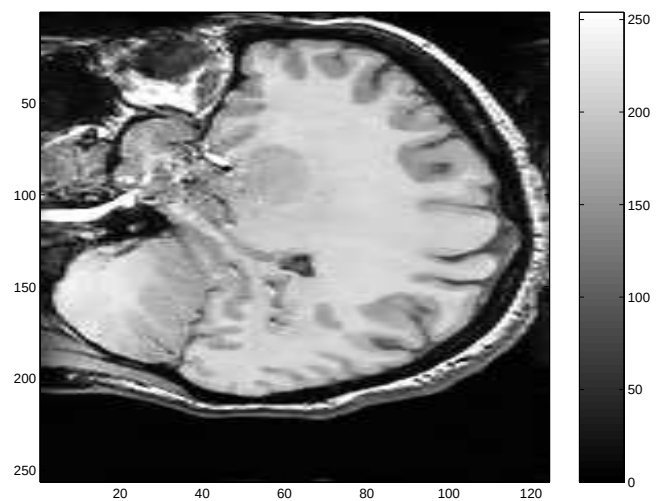


Figure 3: The sagittal cross section of the brain. Observations are tissue intensity values. White regions are the white matter and gray regions are the gray matter. Segmentation of gray and white matter can be formulated as the maximum likelihood estimation in Gaussian mixture using the Expectation-Maximization (EM) algorithm. Possible project topic if you can combine this idea within Bayesian framework.

3. *Iterated kernel smoothing* From Figure 1 and 2, we can see that our estimation  $\hat{\mu}$  is not smooth enough. So we need to smooth image more. But when we increase bandwidth  $\sigma$ , we need to increase the current window size which is set to be  $5 \times 5$ . But when we increase bandwidth parameter  $\sigma$ , we are chopping off tail regions and kernel will not be close to Gaussian anymore. So we need to increase the window size where the kernel weights are computed. Increasing window size may not be elegant in numerical implementation. By introducing the concept of iterated kernel smoothing, we avoid messy computer coding.