

Stat 992: Lecture 34

Anisotropic kernel smoothing I.

Moo K. Chung mchung@stat.wisc.edu

April 21, 2004

1. *Motivation.* An image may consists of two different classes and each class may have different variability so that when we want to smooth image we may want to apply different bandwidth to different classes. This requires smoothing along the class boundaries and the smoothed data should not cross over the boundary. Data may be defined along vector fields so that we need to smooth data along the tensor fields (Figure 1).

2. *Diffusion tensor data.* Anisotropic kernel smoothing is best illustrated by doing a specific example. Let $D(x) = (d_{ij}(x))$, $x \in \mathbb{R}^n$ be raw diffusion tensor images given in Figure 1. For fixed x , $d_{ij}(x)$ is a non negative definite tensor which can be checked by computing the determinant and trace. For simplicity we normalize D by $D / \max_x D(x)$. In this way two positive eigenvalues of D are constrained so that $1 \geq \lambda_1(x) \geq \lambda_2(x) \geq 0$. After downloading data from <http://www.stat.wisc.edu/~mchung/teaching/datatensor/DTI.mat> run following codes

```
load DTI
vecxx=vec(xx); vecxy=vec(xy);
vecyy=vec(yy);
gmax=max([vecxx;vecxy ;vecyy]);
xx=xx/gmax; xy=xy/gmax; yy=yy/gmax;
imagesc([xx xy yy]);colorbar;
```

3. Consider principal eigenvector $\mathbf{V} = (V_1, \dots, V_n)'$ of D . For simplicity, we will only consider smoothing along $\mathbf{V}$ for now. Running the following code, we generate $\mathbf{V}$ and λ_1 (Figure 2).

```
xdim=170; ydim=203;
for i=1:xdim*ydim
    A=[vecxx(i) vecxy(i)
      vecxy(i) vecyy(i)];
    [v,d]=eig(A);
    V(i,:)=v(:,2)';
    lambda1(i)=d(2,2);
end;
```

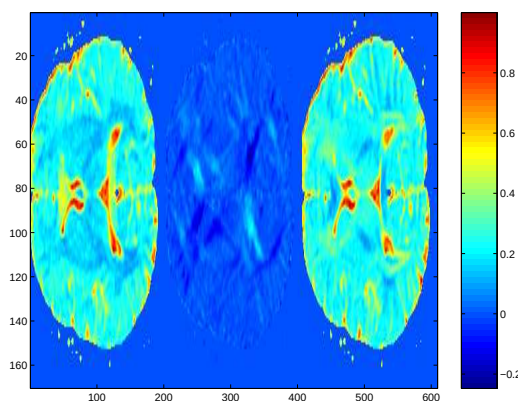


Figure 1: Diffusion tensor images D_{xx} , D_{xy} , D_{yy} .

The eigenvectors are normalized so that $\|\mathbf{V}\| = 1$. For simplicity, we may let $\mathbf{V} \leftarrow \lambda_1 \mathbf{V}$. The stream line $\mathbf{x} = \psi(t)$ corresponding to the vector field is given by

$$\frac{d\psi}{dt} = \mathbf{V} \circ \psi(t) + \epsilon(t)$$

where ϵ is random noise vector. This ordinary differential equation (ODE) gives a family of integral curves whose tangent vector is $\mathbf{V}$ (Betounes, 1998). The line element is

$$d\psi^2 = V_1^2 dx_1^2 + \dots + V_n^2 dx_n^2.$$

which gives Riemannian metric tensor $g_{ij} = V_i^2 \delta_{ij}$. If we smooth some data with anisotropic Gaussian kernel K_H such that we smooth more along the larger metric distance, we may let $HH' = 2tG$, $G = (g_{ij})$, i.e.

$$HH' = 2t \begin{pmatrix} V_1^2 & 0 & \dots & 0 \\ 0 & V_2^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & V_n^2 \end{pmatrix}.$$

By introducing the scaling parameter t , we left a room for adjusting the amount of smoothing possibly by an optimal bandwidth. For the above choice

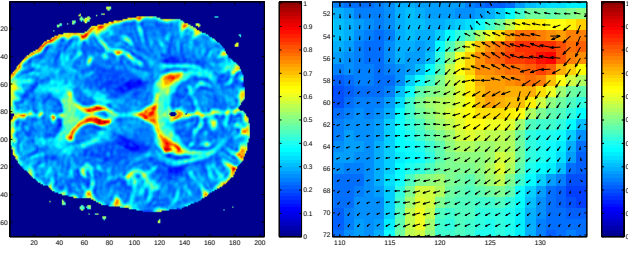


Figure 2: Left: Principal eigenvalues λ_1 , Right: Principal eigenvectors $\mathbf{V}$ displayed using `quiver` command in MATLAB.

of the covariance matrix, anisotropic kernel is given by

$$K_t(\mathbf{x}) = \frac{1}{(4\pi t)^{n/2}} \prod_{j=1}^n \frac{1}{|V_j|} \exp\left(-\frac{x_j^2}{4tV_j^2}\right).$$

In the case $V_j = 0$ while $|\mathbf{V}| \neq 0$ in (1), the kernel is undefined. To avoid the divergence in numerical implementation, we can introduce very small ϵ and let $HH' = 2tG + \epsilon I$ instead of $\mathbf{V}$ in computing kernel.

```
%kernel computation
[dx,dy]=meshgrid(-1:1)
K=inline('exp(-(x.^2+y.^2)/(4*t))')
/sum(sum(exp(-(x.^2+y.^2)/(4*t))))')
e=0.0001;
ker=zeros(ksize,ksize);
for i=2:(xdim-1)
    for j=2:(ydim-1)
        wx= Vx(i-1:i+1,j-1:j+1)+e;
        wy= Vy(i-1:i+1,j-1:j+1)+e;
        weight = K(0.5,dx./wx,dy./wy);
        kernel(:, :, i, j)=weight;
    end;
end;

%iterated smoothing
f=lambda1; g=zeros(xdim,ydim);
for k=1:20
    k
    for i=2:(xdim-1)
        for j=2:(ydim-1)
            temp=f((i-1):(i+1),(j-1):(j+1));
            g(i,j)=sum(sum(temp.*kernel
                (:, :, i, j)));
        end;
    end;
    f=g;
end;
```

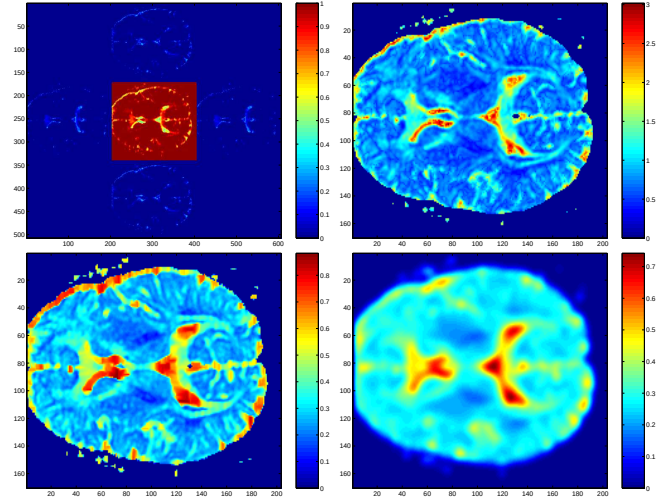


Figure 3: Top left: 9-pixel neighboring kernel, Top right: Spatially varying filter size. Filter size is large where the directional information is strongest. Smoothing of the principal eigenvalues. Bottom left: anisotropic smoothing with $m = 20, t = 0.5$ which gives maximum filter size of 3.1. Bottom right: Isotropic kernel smoothing with the uniform filter size of 3.1.

4. *Discussion.* This is a spatially adaptive filter which has the property of reducing noise while preserving edge information. Because the bandwidth changes spatially, in order to be able to compare the result with isotropic kernel, we need to come up with the amount of smoothing in average sense. Note that at each position $\mathbf{x}$, the amount of smoothing is $2tV_j^2$ along the j -th coordinate. The shape of kernel is an ellipsoid $\frac{x_1^2}{2tV_1^2} + \dots + \frac{x_n^2}{2tV_n^2} = 1$. The mean distance from the origin to the surface of the ellipsoid is $\sigma(\mathbf{x}) = \sqrt{2t \sum_{j=1}^n |V_j|/n}$. Then we perform iterated kernel smoothing with m iterations, the total amount of smoothing would be $\sigma_{total}(\mathbf{x}) = \sqrt{m}\sigma(\mathbf{x})$. Figure 3 shows spatially varying filter size. The advantage of using only the principal eigenvectors would be the simplicity of the implementation while the drawback is that the computation of the principal eigenvectors takes a fair amount of time in MATLAB as well as the fact that the method does not completely utilize full diffusion tensor information. Further the above kernel has a problem in the case of constant field $\mathbf{V} = (1, \dots, 1)/\sqrt{n}$. In this particular case the kernel is isotropic. A way to avoid this is to introduce the cross-product terms into the matrix tensor. Metric like $g_{ij} = V_i V_j$ would give such structure. However such metric tensor is not positive definite.