On the use of a good distance in machine learning

Summary

The purpose of this practical session is to illustrate the influence of a major choice in machine learning : the definition of the distance between the objects observed, which is used in algorithm to predict a value (regression) or a class (classification). A suitable cooling-off period is necessary before defining a distance that is :

- *robust to the noise on the data*
- *describe among useless generalities on the data the essential phenomenon we are looking for*
- *can be computed efficiently*

is a crucial step to design efficient machine learning algorithms. Since we will use many signal processing tools in this session, a natural programming language will be Matlab, but of course it would be desirable to make some efforts to translate the codes below in Python.

1 A toy example

To illustrate the topic of this session, we first imagine that we have in our hand some “functionnal" observations, *i.e.* 1-dimensional curves defined on $[0,1]$ that are either observed at any point of $[0,1]$, or in a discrete grid. In what follows, we assume that we can observe virtually two sets of functions

$$Y_i(x) = f(x) + W_i(x), \quad \forall x \in [0,1], \tag{1}$$

where W_i stands for a white noise. Even if this white noise has a sharp mathematical definition, we will only keep in mind an intuitive signification of this noise has a centered and continuous random variable.

1.1 Description of the data

1. Download and extract the zip file “TP6.zip".

2. Use the following Matlab code.

```
%Script of the database generation
clear all
close all
p=5;
s=1;
%Random selection of a baseline signal
theta1=signal(p,s);
%Random selection of a baseline signal
theta2=signal(p,s);

sigma=0.1;ntrain=100;
%Generation of samples
[tab1,curves1]=data_gen(theta1,sigma,p,ntrain);
[tab2,curves2]=data_gen(theta2,sigma,p,ntrain);
tabr1=real(tab1);tabi1=imag(tab1);
tabr2=real(tab2);tabi2=imag(tab2);

%Database
tableau=cat(1,tab1,tab2);
```

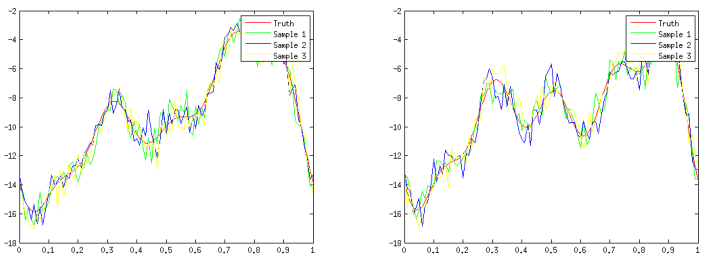

FIGURE 1 – Examples of functionnal samples produced by the sequence.

3. While reading the .m code of each used program, explain the job of “signal.m" and “data_gen.m." What kind of theoretical signal processing tool is used here ?

1.2 Naive distance

What is the natural choice for measuring a distance between samples (write a formula that involves the signal processing tool used to generate the data) ? What do you think about the effect of the smoothness of the signals according to the infinite dimensional space that described the data ?

Unsupervised classification (clustering)

Comment the following code (statistical framework, used distance, graphical outputs).

```
tic
[eps_L2,data]=classif_L2(tableau);
toc
```

Discuss on the quality of the results with respect to the noise on the data.

Supervised classification

In a similar way, discuss on the following code

```
%Training samples
[tab1,curves1]=data_gen(theta1,sigma,p,ntrain);
[tab2,curves2]=data_gen(theta2,sigma,p,ntrain);
Xtrain=[tab1;tab2];
Ytrain=[ones(1,ntrain),-ones(1,ntrain)];

%Testing samples
ntest=1000;
[test1,curvestest1]=data_gen(theta1,sigma,p,ntest);
[test2,curvestest2]=data_gen(theta2,sigma,p,ntest);
Xtest=[test1;test2];
Ytest=[ones(1,ntest),-ones(1,ntest)];

d=4;

kn=ceil(ntrain^(2/(2+d)));
decision=knn(kn,Xtrain(:,1:d),Ytrain,Xtest(:,1:d));
1-mean(2*decision-1==Ytest)
```

In particular,

1. Take a moment to justify a limitation in d and in the choice of k_n.
2. In a white noise model with Fourier coefficients of the observed signals corrupted by a gaussian noise $\mathcal{N}(0, \sigma^2)$, how should behave the optimal threshold d if the function f is s-smooth (Hard thresholding regularization) ?
3. Propose an algorithm to optimize the choice of k_n and write the corresponding code.

1.3 Another noise model

Let us now consider a slightly different tricky toy model of functionnal signals :

$$Y_i(x) = f \circ \phi(x), x \in [0,1]$$

where ϕ is a random bijective transformation of the state $[0,1]$, which is of course unobserved. A typical example is the following one

$$Y_i(x) = f(x - \tau_i), x \in [0,1], \tag{2}$$

where τ_i is an unobserved random shift parameter, whereas f is a 1-periodic signal.

Data generation

1. In this particular case, what is the bijection ϕ ?
2. Run the following code

```
clear all
close all

p=5;
s=1;
%%%%%%%%%%%%%%%%%%%%
theta1=signal(p,s); theta2=signal(p,s);

sigma=0.1;ntrain=10;
% Sample Generation
```

```
[tab1,curves1]=data_gen2(theta1,sigma,p,ntrain);
[tab2,curves2]=data_gen2(theta2,sigma,p,ntrain);
```

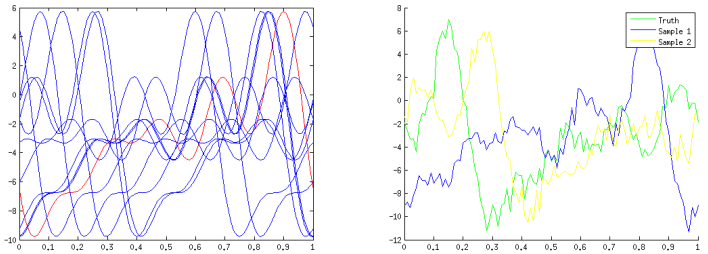

FIGURE 2 – Examples of functionnal samples produced by the sequence.

3. While reading the function data_gen2, briefly explain how it works.
4. How should be chosen σ above to be compatible with the model (2) ? How is distributed the random variable τ ?
5. Generate the data according to (2).

Supervised and unsupervised classification

1. Following carefully the previous matlab lines, perform an Hierarchical Ascendant Classification and compare the results to the one obtained in the white noise model (1).
2. In the meantime, draw a similar comparison of the obtained supervised classification with the nearest neighbours.
3. How should we improve the statistical treatment of such a toy model ?

Fréchet distance

It is worth trying to improve the distance used to measure the differences between signals : ideally, we would like to get signals in the same class close, and in a different class significantly distant each others. A first natural approach is to use the following distance

$$d_{Frechet}(Y,Z)^2 := \inf_{\tau\in[0,1]} \|Y(.-\tau) - Z\|_2^2, \qquad (3)$$

where $Y(.-\tau)$ stands for the translated curve of a value τ.

1. Explain the benefits of the use of $d_{Frechet}$ when one faces the model (3)
2. Briefly explain the following code.

```
tic
%Classification par distance de Fréchet
eps_F=classif_Frechet(tableautest);
toc
```

3. Test numerically the robustness of this distance while adding a white noise model with $\sigma > 0$ in the model (2).
4. Develop a matlab code to deal with the supervised classification problem while using the Nearest Neighbour algorithm associated to the Frechet distance.
5. Can you find a drawback to this approach ?

Tricky distance

Another way to deal with the randomly shifted curve models is maybe to improve again the distance to make-it easier to compute. Hence, a natural idea is to build some “invariants" that fully characterize the classes of the signal, while removing as much as possible the noise.

1. Use the following code.

```
tic
%Classification par distance L2 standard
[eps_inv,datatest_inv]=classif_inv(tableautest);
toc
```

2. Explain theoretically the mechanism of the function classif_inv.
3. Extend the use of this idea to build an efficient supervised classification algorithm with the baseline nearest neighbour.
4. Would it be possible to use another supervised classification algorithm with the improvements yielded by the invariants built above ?

1-dimensional real example

Let us now discuss on a first “simple" real example : a one-dimensional set of curves obtained with a biological dataset that describes the evolution of some transcripts and proteins w.r.t. the time on a kidney transplant study.

The purpose of the statistical study is to exhibit some families of curves that behave similarly.

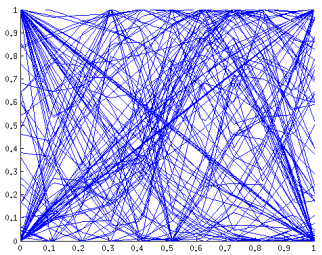

FIGURE 3 – Examples of the different functionnal observations in the kidney transplant database.

We can load the database as follows.

```
clear all
close all
load splineOut.gene.wt.txt
X=splineOut_gene_wt;
[n,l]=size(X);
xtrue=[0:1/(l-1):1];
Khz=3;
for i=1:n
    Y(i,:)=X(i,:);
    M=max(Y(i,:));m=min(Y(i,:));
    Y(i,:)=(Y(i,:)-m)/(M-m);
    figure
    plot(xtrue,Y(i,:));
    coeff(i,:)=fourier_dec(xtrue,Y(i,:),Khz);
end
data=[real(coeff),imag(coeff)];
```

It seems a little bit hard to find something meaningful in the set of curves shown in Figure 3...

1. Briefly explain the commands above.
2. Build a first clustering with the Euclidean distance and the ascendant hierarchical classification algoritghm.
3. Do the same analysis with the invariant distances introduced before.

2-dimensional real example

The last real dataset is the famous MNIST handwritten digits recognition problem. The database can be downloaded here : http://yann.lecun.com/exdb/mnist/.

```
images = loadMNISTImages('train-images-idx3-ubyte');
labels = loadMNISTLabels('train-labels-idx1-ubyte');
display_network(images(:,1:400));disp(labels(1:20));
```

Some very partial observations of the database can be found in Figure 4.

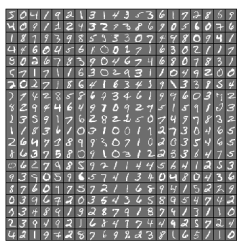

FIGURE 4 – Examples of the different functionnal observations in the kidney transplant database.

The next code aims to test several distances on this dataset to produce a supervised classification algorithm.

1. We start some elementaries command.

```
%Train
Xtrain = loadMNISTImages('train-images-idx3-ubyte');
Ytrain = loadMNISTLabels('train-labels-idx1-ubyte');

%Test
Xtest = loadMNISTImages('t10k-images-idx3-ubyte');
Ytest = loadMNISTLabels('t10k-labels-idx1-ubyte');

% Easy manipulation with handwritten digits

size(Xtrain(:,1))
Z=Xtrain(:,1);
ZZ=reshape(Z,28,28);

imagesc(ZZ,'EraseMode','none',[-1 1])
```

2. A bruteforce NN algorithm can be used quite easily with subsampling in the training set (to obtain a reasonnable time of execution).

```
Xtrain=Xtrain';
Xtest=Xtest';

% 1-1) With a subsampling procedure:

Xtrain=Xtrain(1:500,:);
Ytrain=Ytrain(1:500);
[n,p]=size(Xtrain);
kn=ceil(n^(1/3));
decision=knn_multi(kn,Xtrain,Ytrain,Xtest);

[m,p]=size(Xtest);
% Error rate:
(m-sum(Ytest==decision'))/m
```

Of course, the statistical result is quite poor.

3. It is nevertheless possible to improve this result while taking all the data of the training set and then build a NN.

```
Xtrain = loadMNISTImages('train-images-idx3-ubyte');
Ytrain = loadMNISTLabels('train-labels-idx1-ubyte');
Xtrain=Xtrain';

Xtest = loadMNISTImages('t10k-images-idx3-ubyte');
Ytest = loadMNISTLabels('t10k-labels-idx1-ubyte');
Xtest=Xtest';

Xtest=Xtest(1:1000,:);
Ytest=Ytest(1:1000,:);
[n,p]=size(Xtrain);
kn=ceil(n^(1/3));
decision=knn_multi(kn,Xtrain,Ytrain,Xtest);
[m,p]=size(Xtest);
e=(m-sum(Ytest==decision'))/m
```

NN achieves around 5% of misclassification, and it takes a very long time to compute this classifier ...

4. Again, the above performance is attained while using a standard euclidean distance. Nevertheless, we can use also the *Tangent Distance*, suitably developed by Yann Le Cun : http://yann.lecun.com/exdb/publis/pdf/simard-00.pdf. Explain briefly the main features of this new distance.

5. We now use this famous tangent distance to improve our classification on the MNIST database. First, get the tangent distance C code and compile it with Matlab.

```
mex td.c
```

Now learn how to use the matlab macro to compute both euclidean and tangent distances on the handwritten digits.

```
X1=Xtrain(:,1);X2=Xtrain(:,12);
% An example of L2 distance between two "5"
L2_distance(X1,X2)
% Tangent distance between the same "5"
tangent_d(X1,X2)
```

6. It is possible to study the influence of the tangent distance on the way the several classes are “separated" in the dataset.

```
Z0=[];Z1=[];Z2=[];Z3=[];Z4=[];Z5=[];Z6=[];Z7=[];
Z8=[];Z9=[];
for j=1:length(Ytrain)
    if Ytrain(j)==0          Z0=[Z0,j];      end
    if Ytrain(j)==1          Z1=[Z1,j];      end
    if Ytrain(j)==2          Z2=[Z2,j];      end
    if Ytrain(j)==3          Z3=[Z3,j];      end
    if Ytrain(j)==4          Z4=[Z4,j];      end
    if Ytrain(j)==5          Z5=[Z5,j];      end
    if Ytrain(j)==6          Z6=[Z6,j];      end
    if Ytrain(j)==7          Z7=[Z7,j];      end
    if Ytrain(j)==8          Z8=[Z8,j];      end
    if Ytrain(j)==9          Z9=[Z9,j];      end
end
m=100;U=zeros(10,m);
U(1,:)=Z0(1:m);U(2,:)=Z1(1:m);U(3,:)=Z2(1:m);
U(4,:)=Z3(1:m);U(5,:)=Z4(1:m);U(6,:)=Z5(1:m);
U(7,:)=Z6(1:m);U(8,:)=Z7(1:m);U(9,:)=Z8(1:m);
U(10,:)=Z9(1:m);
for i=0:9
    for j=i:9
        dij=[];
        ddij=[];
        for k=1:m
            for l=1:m
               dij=[dij,L2_distance(Xtrain(:,U(i+1,k)),Xtrain(:,U(j+1,l)))];
               ddij=[ddij,tangent_d(Xtrain(:,U(i+1,k)),Xtrain(:,U(j+1,l)))];
            end
        end
%         subplot(100,i,j);
%         boxplot(dij);
        md(i+1,j+1)=mean(dij);
        vd(i+1,j+1)=var(dij);
        mdd(i+1,j+1)=mean(ddij);
        vdd(i+1,j+1)=var(ddij);
    end
end
```

It would be nice to obtain a 10×10 tabular that represent some crossed boxplots or violinplots of the L2 and tangent distances in the database. (I do not have the stat toolbox ☹).

7. Run the NN algorithm with the tangent distance and conclude the study.