Name:	
ID number:	LiU-ID:

Passed:	Date:

TSBB06 Computer Exercise C
Normalised Convolution

Developed by Klas Nordberg

Computer Vision Laboratory, Linköping University, Sweden

October 31, 2018

Introduction

In this exercise you will work with various aspects of normalised convolution. The first examples apply normalised convolution on a 1D signal, e.g., a signal of time, and later examples apply normalised convolution onto an image, a 2D signal. An important property of this technique is that the algebraic computations for the 1D or 2D (or of any other number of dimensions) are the same, it is just their practical implementations that may have to differ. You will study both the case of a signal where all samples are known, and the case where some of the samples are unknown. In both cases we expand the local signal around each point in some suitably chosen basis, e.g., a basis of polynomials.

Extra

Sections marked with an "Extra" box may be temporarily skipped and completed when the other exercises are finished and if there is time left.

Major issue: The code presented in this guide does not work on Matlab version 7.7. If you don't know which version you are using or how to avoid this problem, ask the teacher at the exercise.

1 Preparatory exercises

Before coming to the computer exercise it is necessary that you have a clear understanding of the material related to dual bases and normalized convolution (lectures 2A-2B). Below, you will find a number of preparatory exercises to be answered **before** the session in order to save time for you, the assisting teacher, and your fellow students. You need to read and understand the specific tasks of the rest of the guide in order to answer some of the preparatory exercises.

1. Write the Taylor expansion of a function s around the point x, as a function of the displacement h.

---

2. For a fixed point x, explain how you can interpret the variation of s in a neighborhood around x as a linear combination of the functions 1, h, and h^2. What are the coefficients in the linear combination? Hint, see previous exercise.

---

3. At the end of Section 3, you are applying a set of three filters on the signal and then compute linear combinations (given by $\mathbf{G}^{-1}$) of the filter outputs to get the coordinates. What would happen if you did it the other way around, i.e., first form a linear combination of the filters (given by $\mathbf{G}^{-1}$) and then apply the resulting filters onto the signal?

---

4. Given a symmetric 2×2 matrix $\mathbf{G} = [G_{11}\ G_{12}; G_{12}\ G_{22}]$, what is its inverse?

---

5. How does the uncertainty signal `cert` change the scalar product $\mathbf{G}_0$?

6. Each element of the metric $\mathbf{G}$ is a scalar product between a specific combination of basis functions. If you look at element $G_{ij} = \langle \mathbf{b}_j | \mathbf{b}_i \rangle$ of the metric, this is a function of the position along the signal. An efficient way of computing each G_{ij} is in terms of a convolution between the certainty function and some function that only depends on the basis functions and the applicability. How?

7. In normalised averaging, described in Section 5, the filter derived from the basis function "1" together with the applicability function is just the applicability function. Why?

8. How do you compute the single element of the metric $\mathbf{G}$ in this case (normalised averaging)?

2 Starting and setting up things

type

Start Matlab and ~~change directory to the folder that is used in these exercises by typing~~ in Matlab's command window:

```
~~cd /site/edu/bb/MultidimensionalSignalAnalysis/ExerciseC/~~
addpath /courses/tsbb06/ExerciseC/
```

This exercise uses standard functions in Matlab, so there are no special functions involved here, but you will use the image `Scalespace0.png` that is found in this library. Alternatively, copy the file from here: https://en.wikipedia.org/wiki/File:Scalespace0.png

In order to simplify the exercises, you should write the Matlab code that you use into a script file (with extension `.m`) that can be executed in order to generate all results that are requested. This script file should be placed in your own home directory, ~~so you need to add it to your path (if it is not already there~~)

```
~~addpath /edu/dir~~            ~~Replace dir with your user name~~
```

In the exercise there are several questions that you should answer, before or during the session. Notice that some of the answers are best illustrated by an image. Make sure to keep this image in order to demonstrate the result to the examiner.

3 Polynomial Expansion

In this task you will compute a local polynomial expansion for a 1D discrete signal using polynomials up to order two. Start by creating a signal for which the local polynomial is known, e.g., a sinusoidal, and plot it:

```
k = 0:1:100;
s = sin(k/10);
figure(1);plot(s);
```

There are three linearly independent polynomials up to order two, the basis functions of the subspace where the local signal is being approximated, and you will use $1, x, x^2$. They are defined on a discrete coordinate $x = -3, \ldots, 3$:

```
x=(-3:3)';
b0=ones(7,1);
b1=x;
b2=x.^2;
figure(2);
subplot(4,1,1);plot(b0,'-o');
subplot(4,1,2);plot(b1,'-o');
subplot(4,1,3);plot(b2,'-o');
```

As applicability function a you will use a gaussian that fits reasonably well within a 7-point window:

```
a = exp(-x.^2/4);
figure(2);
subplot(4,1,4);plot(a,'-o');
```

The corresponding filters are constructed by multiplying (point-wise) each of the basis functions with the applicability function and then reversing the order of the points:

```
f0 = b0.*a; f0 = f0(end:-1:1);
f1 = b1.*a; f1 = f1(end:-1:1);
f2 = b2.*a; f2 = f2(end:-1:1);
figure(3);
subplot(3,1,1);plot(f0,'-o');
subplot(3,1,2);plot(f1,'-o');
subplot(3,1,3);plot(f2,'-o');
```

With the filters at hand, you can now convolve the signal with the filters:

```
h0 = conv(s,f0,'same');
h1 = conv(s,f1,'same');
h2 = conv(s,f2,'same');
```

In normalised convolution, the filter results at each point in time (or space in the 2D case) are interpreted as the scalar products between the local signal and the basis functions, where the scalar product is given by the applicability function. These scalar products are then the dual coordinates, relative to the basis function, of the projection of the local signal onto the subspace spanned by the basis function. In order to obtain proper coordinates you must multiply with the inverse of the metric. In this exercise the metric is constant along the signal:

```
G0 = diag(a);
B = [b0 b1 b2];
G = B'*G0*B
```

QUESTION: Is the basis orthonormal?

Compute the proper coordinates by transforming the dual coordinates by means of the inverse of the metric:

```
c = inv(G)*[h0;h1;h2];
figure(4);
subplot(3,1,1);plot(c(1,:))
subplot(3,1,2);plot(c(2,:))
subplot(3,1,3);plot(c(3,:))
```

QUESTION: The last three plots show the elements of the coordinate vector **c**. Are they as expected? Hint: polynomial expansion is related to a Taylor expansion of a function, as described in preparatory exercise 2. In what way?

Demonstrate that the idea of expanding the local signal approximately as a linear combination of the basis function and coordinates is correct. Pick some fixed point x, for example x=60, and plot the local signal around this point. Plot also the corresponding signal that is reconstructed from the basis functions at this point, and the difference between the two.

```
figure(5);
localsig=s(60-3:60+3);
reconsig=(B*c(:,60))';
diffsig=localsig-reconsig;
subplot(3,1,1);plot(localsig);
subplot(3,1,2);plot(reconsig);
subplot(3,1,3);plot(diffsig);
```

QUESTION: Is the reconstructed signal very different or very similar to the local signal? Explain why.

Extra

Try to do the same computations for the case of polynomials of higher order than two. What coordinates to you get?

4 Uncertain data

You will now look at what happens when the signal values are not known for all samples. This is done by generating a binary certainty function and then multiplying this point-wise onto the signal

```
cert = double(rand(1,101)>0.2);
scert = s.*cert;
figure(6);plot(scert);
```

This produces a signal vector $\mathbf{s}_c$ where 20% of the samples are set to zero. On the other hand the positions of these unknown samples are known.

Apply the same calculations on the new signal as in the previous task, without making use of the certainty information.

QUESTION: In what way do the results change? Explain why.

The way that normalised convolution deals with uncertain data is to include the certainty function in the scalar product $\mathbf{G}_0$. In the previous task, the scalar product is constant for all filter outputs. In this task, since the certainty function is no longer constant, the scalar product and therefore also the metric will be a function of position along the signal, i.e., if you want to transform the dual coordinates at a certain time point, you need to determine the metric, including its inverse, at that point. In order to simplify the computations, you will only use the first two basis functions, i.e., **G** is a 2×2 matrix that depends on the position of the signal.

You are now going to implement all these computations in order to compute a coordinate vector at each point that holds information about the local signal value and the local derivative, taking the certainty function into account. First the filter outputs:

```
h0 = conv(scert,f0,'same');
h1 = conv(scert,f1,'same');
```

Compute the elements of the metric **G** in terms of convolutions, using the expressions for the elements of **G** described in preparatory exercise 6:

```
G11 = conv(...);
G12 = conv(...);
G22 = conv(...);
```

Then transforming the dual coordinates (the filter responses) into proper coordinates (c_0, c_1) by multiplying by the inverse of **G**:

```
detG = G11.*G22-G12.^2;
c0 = (G22.*h0-G12.*h1)./detG;
c1 = (-G12.*h0+G11.*h1)./detG; figure(7);
subplot(2,1,1);plot(c0)
subplot(2,1,2);plot(c1)
```

QUESTION: How do you interpret the computations that are made for `c0` and `c1`?

QUESTION: Compare this result, (c_0, c_1), to the previous one where the certainty function was not taken into account. What it is the difference?

QUESTION: The signals `c0` and `c1` should not exhibit any border effects. In the previous task you did get border effects when you computed coordinates `c0` and `c1` (see figure 4). Explain this observation.

QUESTION: Modify the signal by removing a higher rate of the samples (e.g. 50%) and see how this changes the result. How large rate of samples can be removed without too much distortion in the measured coordinates (c_0, c_1)?

Extra

Try also other types of applicability functions, e.g., by using a smaller standard deviation of the Gaussian. How does this change the result?

Extra

Try to use only a single basis function. Try with "1" and "x", respectively, and analyse the result.

5 Normalised averaging of images

You are now going to extend the computations related to normalized convolution to 2D signals or images. The algebraic approach is the same and it is more the practical implementation of the computations that may differ from the 1D case. In particular you need sometimes to worry about reshaping 2D signals or filter into column vectors, or vice versa.

In this first task, you will analyse an image with uncertain data, and simply fill in the missing pixels. In order to do this, you need only the coordinate of the basis function "1", i.e., it is sufficient to use only a single basis function. Consequently, both the metric **G** and its inverse are 1×1 matrices, or a scalars. However, since the uncertainty of the signal varies over the image, they will both be functions of the image coordinates.

Start by loading the image

```
im = double(imread('Scalespace0.png'));
```

```
figure(8);colormap(gray);imagesc(im);
```

and produce a certainty signal that removes a rather high percentage of the pixels by setting them to zero. Try starting with something like 50% removed pixels. You can vary this percentage later on if you like. This distorted signal is what we have access to and can process:

```
cert = double(rand(size(im)) > 0.6); imcert = im.*cert;
figure(9);colormap(gray);imagesc(imcert);
```

The applicability function a will be realised on a 7×7 grid, with the origin at the center:

```
x = ones(7,1)*(-3:3)
y = x'
a = exp(-(x.^2+y.^2)/4);
figure(10);mesh(a);
```

The applicability function is a low-pass filter. A possibility of computing values for the missing pixel values in `imcert` is to average or low-pass filter the image with the applicability function:

```
imlp = conv2(imcert, a, 'same');
figure(11);colormap(gray);imagesc(imlp);
```

QUESTION: Did this solve the problem of missing pixels? Explain why.

Instead of just a low-pass filtering of the image, you are now going to compute the coordinate of basis function "1" in the proper way using normalised convolution. In this particular case, when a single basis function "1" is used, the operation is sometimes called *normalised averaging.*

In accordance to preparatory exercise 7, the filter output is given by `imlp` that has already been computed.

Compute the metric in accordance to preparatory exercise 8, and apply its inverse onto the dual coordinate

```
G = ...
c = imlp./G;
figure(12);colormap(gray);imagesc(c);
```

QUESTION: Is this result better than before? Explain why?

QUESTION: Change the width of the applicability function to smaller or larger values. What happens?

QUESTION: Increase the number of missing pixels. How large percentage can be removed until you cannot see the picture in the reconstructed image?

QUESTION: Remove 90% of the pixels and reconstruct the image by means of normalised averaging. There will be a few pixels of random positions in the result that are black and if you investigate their value it is *Not-A-Number*. Explain why.

QUESTION: How would you do to further improve the result?

Try to change the applicability a to improve the result.

QUESTION: How did you do it? Do you succeed?