Lecture Notes

Supervised and Unsupervised Classification of Hyperspectral Images

Gabriel Dauphin

March 20, 2024

Contents

1 Classification of hyperspectral images **3**
1.1 Hyperspectral images . . . 3
1.2 Supervised and unsupervised classification of hyperspectral images . . . 9
1.3 Simple predictors . . . 10
1.4 Accuracy and loss functions . . . 12
1.5 Training, testing and validation sets . . . 13
1.6 Confusion matrix . . . 13

2 Image processing **14**
2.1 Segmentation . . . 14
2.2 Edges as a mean for segmentation . . . 15
2.3 Detection of connected components . . . 18
2.4 Use of iterated algorithms . . . 20
2.5 Clustering regarded as an optimization problem . . . 23

3 Learning regarded as an optimization problem **25**
3.1 Optimization problem . . . 25
3.2 Simulated annealing . . . 25
3.3 Method of least squares . . . 28

4 Predicting the learning performances and probabilistic framework **29**
4.1 Linear discriminant analysis . . . 29
4.2 Predicting the true probabilities . . . 30
4.3 Prior and Bayes formula . . . 30

5 More in depth with probabilities **33**
5.1 Probabilities . . . 33
5.2 Using Gaussians . . . 34
5.3 Probabilities as a loss function designer . . . 36

6 Curse of dimensionality, regularization and sparsity **41**
6.1 Data preparation . . . 41
6.2 Feature construction . . . 42
6.3 Kernel trick . . . 46
6.4 Curse of dimensionality and feature extraction . . . 47
6.5 Principal Component Analysis . . . 49
6.6 Supervised feature extraction . . . 54
6.7 Regularization . . . 54
6.8 Feature selection . . . 59

7 Spatial context **61**
7.1 Spatial context . . . 61
7.2 Texture descriptors . . . 61
7.3 Noise estimation . . . 63
7.4 Spatial prior . . . 65

8 Supplementary material regarding matrices **65**
8.1 Proving that kmeans is related to an optimization problem . . . 65

1 Classification of hyperspectral images

1.1 Hyperspectral images

Exercise. 1 *What image is this showing?*

```
R=[1;1;0]; G=[0.5;1;1]; B=[0;1;0];
im=cat(3,R,G,B),
figure(1); imshow(im);
```

Exercise. 2

1. *Find on the web the hyperspectral image Pine and retrieve it in Octave, using for instance*

   ```
   https://www.ehu.eus/ccwintco/index.php/
     Hyperspectral_Remote_Sensing_Scenes
   https://engineering.purdue.edu/~biehl/
     MultiSpec/hyperspectral.html
   ```

2. *Find the size of each bandwidth image*
3. *Find the number of bandwidths*

Answer. *Correction of exercise 2 I followed the following steps*

- *Retrieve* `Indian_pines_corrected.mat` *and* `Indian_pines_gt.mat`
- `T=load` *to retreive the image*
- `fieldnames(T)` *to get the name of the variable*
- `size` *to get the answers.*

```
ans =
{
  [1,1] = indian_pines_corrected
}

ans =

   145   145   200
```

The size of the image is 145×145. *The number of bandwidths is* 200.

```
function ex37()
  T=load('../dataset/Indian_pines_corrected.mat');
  fieldnames(T),
  size(T.indian_pines_corrected),
end
```

Exercise. 3

1. *Retrieve the calibration information*

   ```
   https://www.ehu.eus/ccwintco/index.php/
     Hyperspectral_Remote_Sensing_Scenes
   https://engineering.purdue.edu/~biehl/
     MultiSpec/hyperspectral.html
   ```

2. *Considering a horizontal line located at the center of the image, find the coordinate of its left most point.*
3. *Plot the spectral intensities as a function of the bandwidths number.*
4. *Plot the spectral intensities in terms of radiance and as a function of the center wavelength.*

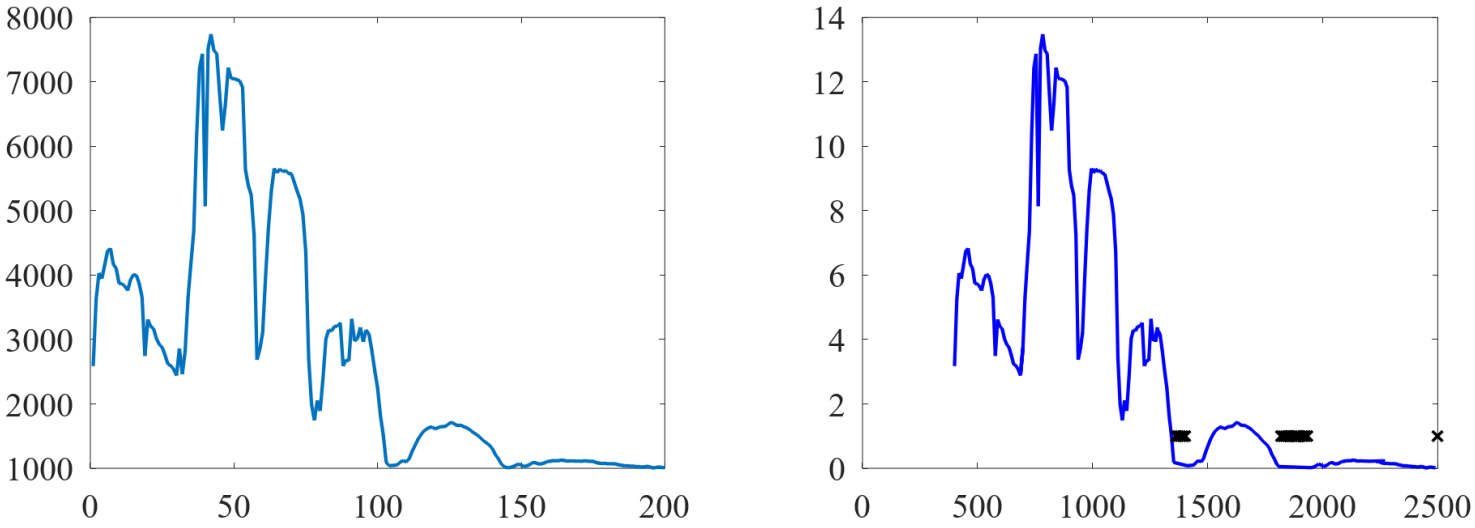


Figure 1: Left: spectral intensities as a function of the bandwidths number. Right: spectral intensities in terms of radiance and as a function of the center wavelength. Exercise 3

Answer. *Correction of exercise 3*

1. *Using the second website, I went to the following websites*

   ```
   https://purr.purdue.edu/publications/1947/1
   https://purr.purdue.edu/publications/1947/supportingdocs?v=1
   ```

   and got the following text file named `Calibration_Information_for_220_Channel_Data_Band_Set.txt`

   ```
   Information on 220 Channel
   AVIRIS Data Set
   Location

   This data is from the AVIRIS (Airborne Visible/Infrared Imaging Spectrometer)
   built by JPL and flown by NASA/Ames on June 12, 1992. The scene is over an area
   6 miles west of West Lafayette. The scene is a subset of a significantly larger
   image file.

   These data are calibrated data. In other words the data values in the scene are
   ```

```
proportional to radiance. 1000 has been added to the calibrated data so that all
data values in this scene are positive. To convert the scene data values (SDV)
to radiance values (RV), one must first subtact 1000 and then divide by the gain_
factor that JPL used which is 500.

RV = (SDV-1000) / 500.

The RV units are W * cm^-2 * nm^-1 * sr^-1.

This information is dated November 1992. It came with the tape documentation
from JPL for this AVIRIS data.
Spectral Calibration

FWHM = Full Width Half Maximum. In other words it is the width of the spectral band.
```

AVIRIS Band #	Data Channel #	Center Wavelength (nm)	FWHM (nm)	Center Uncertainty (nm)	FWHM Uncertainty (nm)
1	(not used - the band was all 0's)				
2	1	400.02	9.78	0.92	0.50
3	2	409.82	9.82	0.92	0.50
4	3	419.62	9.85	0.93	0.50
5	4	429.43	9.89	0.94	0.50
6	5	439.25	9.92	0.95	0.50
7	6	449.07	9.94	0.95	0.50
8	7	458.90	9.97	0.96	0.50
9	8	468.73	9.99	0.97	0.50
10	9	478.57	10.01	0.97	0.50
11	10	488.41	10.02	0.98	0.50
12	11	498.26	10.04	0.99	0.50
13	12	508.12	10.05	1.00	0.50
14	13	517.98	10.05	1.00	0.50
15	14	527.85	10.06	1.01	0.50
16	15	537.72	10.06	1.02	0.50

2. *The center horizontal line is at the line number* 73*:*

$$(72-1)+1=(145-74+1)=72 \textit{ and } 2\times 72+1=145$$

The left most point is at $m=73$*,* $n=1$*.*

3. *It is shown on the left of figure 1.*

```
figure(1); plot(im(73,1,:),'linewidth',2);
```

4. *It is shown on the right of figure 1. The following bands have been removed* $[104-108],[150-163],220$ *as indicated in the dataset because of the water absorption. We get the horizontal scale using the following steps.*

   - *Open the text file*
   - *Convert each line into arrays of numbers (a line starting with a letter is converted into a void array).*
   - *Check if the array is non-empty and if its second number is not included in the list of removed bandwidths.*
   - *Stack in a vector the third component of each non-void array.*

We get the vertical scale by making the following affine transform

$$RV = \frac{SDV - 1000}{500}$$

```
function ex38()
  question3();
  question4();
end

function question3()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  figure(1); plot(im(73,1,:),'linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/ex38_fig1.png');
end

function question4()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  RV = (im(73,1,:)-1000) / 500;
  [sc,sc_not]=scale_nm();
  figure(2); plot(sc,RV,'b-','linewidth',2,sc_not,1,'kx','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(2,'../../images/ex38_fig2.png');
end

function  [sc,sc_not]=scale_nm()
  sc=[]; sc_not=[];
  bd_supp=[104:108, 150:163, 220];
  file=fopen('../dataset/Calibration_Information_for_220_Channel_Data_Band_Set.txt');
  cpt=0;
  while(1)
    line=fgetl(file);
    if -1==line break; end
    array=str2num(line);
    if ~isempty(array)
      if ismember(array(2),bd_supp)
        sc_not=[sc_not array(3)];
        continue,
      end
      if length(array)>=3
        cpt+=1;
        sc(cpt)=array(3);
      end
    end
  end
  fclose(file);
end
```

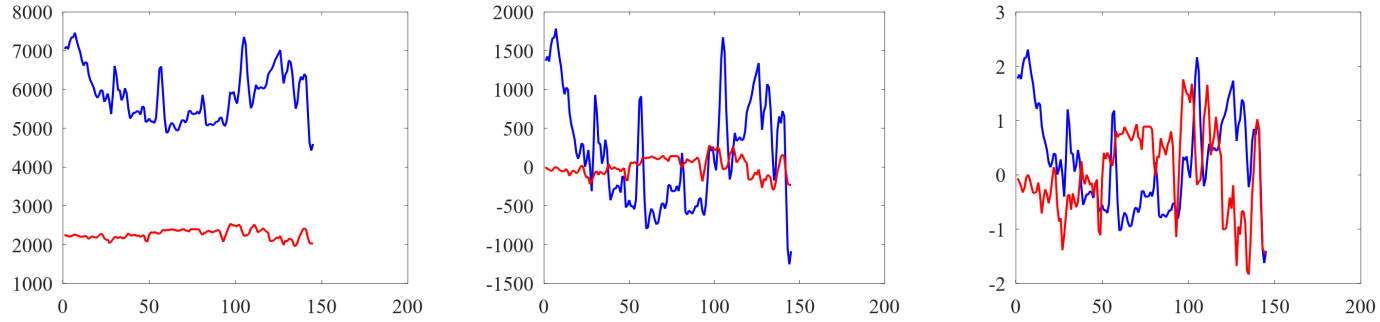
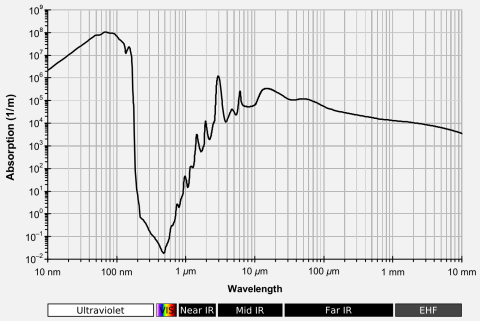
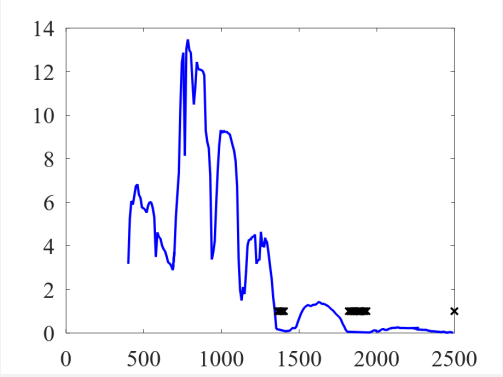


Figure 2: Left: non-centered profile line. Middle: centered profile line. Right: normalized profile line. Blue is $k_1 = 50$, red is $k_2 = 100$.

Exercise. 4





1. *Explain the reason for removing the bandwidths indicated with black crosses on the right.*

- $\mathcal{I}$ is a hyperspectral image, it is a rank 3 tensor, $I(m_1, m_2, k)$ is a component.
- m_1, m_2, k are the row, column and bandwidth indexes.
- M_1, M_2, K are the number of columns, rows and bandwidths.
- μ is the mean of a set of numbers.
- σ is the standard deviation of a set of numbers.

Exercise. 5 *We consider again the central horizontal line. For each questions, use appropriate notations to express the computed quantities.*

1. *Plot the profile line considering the spectral intensity of the bandwidth number* 50 *and the bandwidth number* 100.
2. *Center both lines. We here consider that centering assume that pixels at a given bandwidth should be processed in a similar manner.*
3. *Normalize them so that their variance is equal to one.*

Answer. *Correction of exercise 5*

1. *Let us call*

$$m_c = 73 \quad k_1 = 50 \quad k_2 = 100$$

The two profile lines are

$$n \mapsto \mathcal{I}(m_c, n, k_1) \quad \textit{and} \quad n \mapsto \mathcal{I}(m_c, n, k_1)$$

2. *The two average intensities are*

$$\mu_1 = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathcal{I}(m,n,k_1) \quad \mu_2 = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\mathcal{I}(m,n,k_2)$$

The transformed profile lines are

$$n \mapsto \mathcal{I}(m_c,n,k_1) - \mu_1 \quad \textit{and} \quad n \mapsto \mathcal{I}(m_c,n,k_1) - \mu_2$$

Proof.

$$\sum_{m=1}^{M}\sum_{n=1}^{N}(\mathcal{I}(m,n,k_1) - \mu_1)) = \sum_{m=1}^{M}\sum_{n=1}^{N}\mathcal{I}(m,n,k_1) - MN\mu_1 = 0$$

□

3. *The two standard deviations are*

$$\sigma_1 = \sqrt{\frac{1}{MN-1}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathcal{I}(m,n,k_1) - \mu_1\right)^2}$$
$$\sigma_2 = \sqrt{\frac{1}{MN-1}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathcal{I}(m,n,k_2) - \mu_2\right)^2}$$

The transformed profile lines are

$$n \mapsto \frac{\mathcal{I}(m_c,n,k_1) - \mu_1}{\sigma_1} \quad \textit{and} \quad n \mapsto \frac{\mathcal{I}(m_c,n,k_1) - \mu_2}{\sigma_2}$$

Proof.

$$\sqrt{\frac{1}{MN-1}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\frac{\mathcal{I}(m,n,k_1)-\mu_1}{\sigma_1}\right)^2} = \frac{1}{\sigma_1}\sqrt{\frac{1}{MN-1}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(\mathcal{I}(m,n,k_1) - \mu_1\right)^2} = 1$$

□

```
function ex40()
  question1();
  question2();
  question3();
end

function question1()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  N=size(im,2); mc=73; k1=50; k2=100;
  figure(1); plot(1:N,im(73,:,k1),'b-','linewidth',2,...
    1:N,im(73,:,k2),'r-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/ex40_fig1.png');
end

function question2()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  N=size(im,2); mc=73; k1=50; k2=100;
```

```
  mu1=mean(im(:,:,k1)(:)); mu2=mean(im(:,:,k2)(:));
  figure(2); plot(1:N,im(73,:,k1)-mu1,'b-','linewidth',2,...
    1:N,im(73,:,k2)-mu2,'r-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(2,'../../images/ex40_fig2.png');
end

function question3()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  N=size(im,2); mc=73; k1=50; k2=100;
  mu1=mean(im(:,:,k1)(:)); mu2=mean(im(:,:,k2)(:));
  sigma1=std(im(:,:,k1)(:)); sigma2=std(im(:,:,k2)(:));
  figure(3); plot(1:N,(im(73,:,k1)-mu1)/sigma1,'b-','linewidth',2,...
    1:N,(im(73,:,k2)-mu2)/sigma2,'r-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(3,'../../images/ex40_fig3.png');
         assert(std(im(:,:,k1)(:)-mu1)/sigma1,1,1e-4);
         assert(mean(im(:,:,k1)(:)-mu1)/sigma1,0,1e-4);
end
```

1.2 Supervised and unsupervised classification of hyperspectral images

- lower case indicates scalars: $f_{m_1 m_2}$, except $I_{m_1 m_2}$.
- Bold lower case indicates row vectors: $\mathbf{x}$.
- Capital letters indicate column vectors: Y.
- Bold capital letters indicate matrices: $\mathbf{X}$, $\mathbf{I}$.
- Sets are in calligraphic fonts: $\mathscr{C}, \mathscr{N}$.
- $n \in \{0 \dots N-1\}$ is the index of sample $\mathbf{x}_n$.
- Image intensities are here considered as a data set x_n
- Bandwidths are now considered as features $\mathbf{x} = [x_0 \dots x_{F-1}]$.
- Land use and land covers are indicated with $yn \in \{0 \dots C-1\}$.
- Ground truth map and classification map: $\mathbf{I}_{\text{gd}}, \mathbf{I}_{\text{c}}$.

Data Set

$$(\mathbf{X}, Y) = \left(\begin{bmatrix} x_{11} & x_{12} & \dots \\ x_{21} & x_{22} & \dots \\ \vdots & & \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} \right) \tag{1}$$

Exercise. 6 *Draw and code with Octave the scatter plot of the following dataset*

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 2 & 0 \\ 2 & 1 \\ 2 & 2 \end{bmatrix} \quad Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

```
function fig_classification()
  X=[repelem((0:2)',3,1)...
    repmat((0:2)',3,1)];
  Y=(X(:,2)>=X(:,1));
  ind1=find(Y==1);
  ind0=find(Y==0);
  figure(1);
    plot(X(ind1,1),...
    X(ind1,2),'g+',...
    'LineWidth',3,...
    X(ind0,1),...
    X(ind0,2),'ro',...
    'LineWidth',3,...
    0.5,1.5,'bs','LineWidth',3);
  legend('y=1','y=0','y ?');
  axis([-0.1 2.1 -0.1 2.1]);
  xlabel('x_1'),ylabel('x_2'),
  set(gca, "linewidth", 3, "fontsize", 14)
  print ("-r600", "./images/fig_classification.png");
end
```

Exercise. 7

1. *Denoting $\mathscr{C}$ the collection of classes that are soybean, $0 \dots C-1$ the total set of classes, write the pseudo-code of an algorithm yielding figure 5.*

Code of exercise 7

```
function ex53()
  question1();
end

function question1()
  T=load('../dataset/Indian_pines_gt.mat');
  im=T.indian_pines_gt;
  figure(1); imshow(im1);
  imwrite(im1,'../../images/ex53_fig1.png');
end
```

1.3 Simple predictors

- Predicted output: $\widehat{y}$ (it depends on $\mathbf{x}$).

Figure 3: Classification map indicating in white the soybean.

- $\mathbf{1}$: Iversion bracket ($\mathbf{1}(0=1)=0$ and $\mathbf{1}(2+2=4)=1$).
- Θ: the whole set of parameters.
- parameters: $\theta_F, \theta_x, \theta_y$.
- Threshold on intensity θ_x.
- $\cdot$ scalar product.
- $\|\quad\|$ norm of the scalar product.
- $\mathbf{x}^T$ is a column vector and ^{T} is the transpose.

Exercise. 8 *We are considering the following predictor which is an example of decision stump.*

$$f_{a,b}(x) = (2a-1)\mathbf{1}(x \leq b) + 1 - a$$

with a and b as parameters.

1. *Compute $f_{1,2}(0.5)$, $f_{1,0.5}(2)$.*
2. *Prove that*

$$\begin{aligned} f_{x,y}(z) = f_{x,z}(y)\mathbf{1}(y=z) \\ +(1-f_{x,z}(y))\mathbf{1}(y \neq z) \end{aligned}$$

Exercise. 9 *We consider a predictor f defined as*

$$f(\mathbf{x}) = \mathbf{1}(2x_1 + x_2 \leq 2) \tag{2}$$

1. *Rewrite f using the scalar product.*
2. *Rewrite f using matrix operations.*
3. *Plot $x_1 \mapsto f([x_1, 0])$.*
4. *Plot $x_2 \mapsto f([0, x_2])$.*

We are considering two sets

$$\mathscr{X}_0 = \{\mathbf{x} \,|\, f(\mathbf{x}) = 0\} \text{ and } \mathscr{X}_1 = \{\mathbf{x} \,|\, f(\mathbf{x}) = 1\}$$

6. *Plot a line separating the two sets and indicate which set is where?*

1.4 Accuracy and loss functions

- Accuracies: OA, AA and A.
- Output and inputs of global extrema: $\max, \min, \operatorname{argmin}, \operatorname{argmax}$.
- Loss function: L.
- Labels: Y and $\widehat{Y}$ stacking y_n and $\widehat{y}_n$.

Exercise. 10 *We are considering the predictor* $f_{a,b}(x)$ *defined as*

$$f_{a,b}(x) = (2a-1)\mathbf{1}(x \leq b) + 1 - a$$

with a *and* b *as parameters. and the following database* $\mathcal{S}_1$

$$\begin{array}{ll} x_1 = 1 & y_1 = 1 \\ x_2 = 1.5 & y_2 = 0 \\ x_3 = 6 & y_3 = 1 \\ x_4 = 3 & y_4 = 1 \\ x_5 = 0.5 & y_5 = 0 \end{array}$$

1. *Plot the function defined by* $b \mapsto A(\mathcal{S}_1, f_{1,b})$.
2. *Plot the function defined by* $b \mapsto A(\mathcal{S}_1, f_{0,b})$.
3. *Select values for* a *and* b *maximizing* $A(\mathcal{S}_1, f_{a,b})$.
4. *Find the corresponding maximum value of* $A(\mathcal{S}_1, f_{a,b})$.
5. *Use* argmax *and* $\max$ *to write the answers to the two last questions.*

1.5 Training, testing and validation sets

- Machine learning tools: SPLIT, LEARN, TEST
- optimal value of a parameter: opt.

Exercise. 11 *Given a certain data set* $\mathcal{S}_3 \bigcup \mathcal{S}_4$ *with* $\mathcal{S}_3$ *as labeled and* $\mathcal{S}_4$ *not labeled.*

1. *Improve the following algorithm using validation sets.*

Require: $\mathcal{S}_3, \mathcal{S}_4$*: data sets*
Ensure: $\mathbf{a}, b$*: linear classifier*

$\mathcal{S}_{opt} = \mathcal{S}_3$.
$(\mathbf{a}_{opt}, b_{opt}) =$ *LEARN*$(\mathcal{S}_{opt})$
Compute A_{opt} *with* $(\mathbf{a}_{opt}, b_{opt})$ *and* $\mathcal{S}_{opt}$.
repeat
    $(\mathbf{x}, (\mathbf{x}', y')) = \operatorname{argmin}_{\mathbf{x} \in \mathcal{S}_4, (\mathbf{x}', y') \in \mathcal{S}_3} d(\mathbf{x}', \mathbf{x})$
    Set $\mathcal{S} = \mathcal{S}_{opt} \bigcup (\mathbf{x}, y')$
    $(\mathbf{a}, b) =$ *LEARN*$(\mathcal{S})$
    Compute $A =$ *TEST*$(\mathscr{S}, (\mathbf{a}, b))$
    if $A > A_{opt}$ ***then***
        $(\mathbf{a}_{opt}, b_{opt}) = (\mathbf{a}, b)$, $\mathcal{S}_{opt} = \mathcal{S}$, $A_{opt} = A$.
until $A <= A_{opt}$

1.6 Confusion matrix

- Confusion matrix $\mathbf{C} = [c_{ij}]$.
- Column vector of predicted labels: $\widehat{Y}$.

Exercise. 12 *We consider the following confusion matrix.*

$$\mathbf{C} = \begin{bmatrix} 5,1 \\ 1,5 \end{bmatrix}$$

1. *Give an example of* Y *and* $\widehat{Y}$ *consistent with* $\mathbf{C}$.
2. *Given* $Y^T = [0,0,0,0,0,0,1,1,1,1,1,1]$, *how many different* $\widehat{Y}$ *are consistent with* $\mathbf{C}$*?*

Exercise. 13 *We are considering the following matrix*

$$\mathbf{C} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

1. *How many classes are there?*
2. *How many samples have been tested?*
3. *Up to some renumbering, what are the values of* y_n*?*
4. *Using the same ordering, what are the values of* $\widehat{y}_n$*?*
5. *Compute the OA?*
6. *Compute the AA?*
7. *Show that* $OA = \frac{c_{00}+c_{11}+c_{22}}{N}$.
8. *Show that* $AA = \frac{1}{3}\left(\frac{c_{00}}{c_{00}+c_{01}+c_{02}} + \frac{c_{11}}{c_{10}+c_{11}+c_{12}} + \frac{c_{22}}{c_{20}+c_{21}+c_{22}}\right)$

2 Image processing

2.1 Segmentation

- Image, slices and components: $\mathscr{I}$, $\mathbf{I}$ instead of $\mathbf{I}_k$, $I(n)$ instead of $I(n,k)$
- $f_{\mathbf{I}}$ and $f_{\mathbf{I}}^{-1}$, using $\{\ldots \mid \ldots\}$ to define a set.
- Sets of pixels: $\mathscr{N}$, $\mathscr{N}_{\mathbf{a}}$, $\mathscr{N}_{\mathbf{b}}$ and $\mathscr{C}_c$.
- $\bigcup$, $\bigcap$,$\emptyset$, partition, $\subset$.
- Cardinality of a set: $|\mathscr{S}|$
- Rounding notations: $\lfloor \ldots \rfloor$, $\lfloor \ldots \rceil$, $\lceil \ldots \rceil$.

Exercise. 14 *To investigate the choice of the threshold, we are investigating the properties of the following curves. Given an image* $\mathbf{I}$*, let* $f_{\mathbf{I}}$ *be defined as*

$$f_{\mathbf{I}}(T) = |\{n \in \mathscr{N} | I(n) \geq T\}|$$

1. *Is* $f_{\mathbf{I}}$ *increasing, decreasing, or...?*

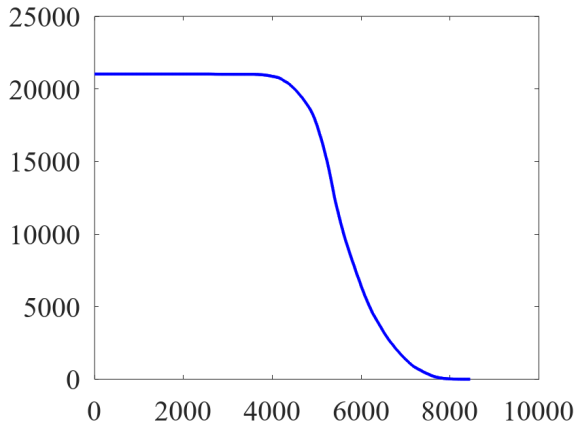


Figure 4: Example of $f_{\mathbf{I}}$-function as defined in exercise 14 for the Indian's Pine hyperspectral image using the bandwidth number 50.

Exercise. 2. *Compute* $f_{\mathbf{I}}(0)$, $\lim_{+\infty} f_{\mathbf{I}}$

Let $\mathbf{I}_r$ *be the centered and normalized image* $\mathbf{I}$ *and* $f_{\mathbf{I}_r}$ *the corresponding function.*

$$I_r(n) = I(n) - \mu \quad \textit{where} \quad \mu = \frac{1}{N}\sum_{n=0}^{N-1} I(n)$$

3. *What is the relation between* $f_{\mathbf{I}}$ *and* $f_{\mathbf{I}_r}$*?*

Exercise. 15 *Based on the definition of a decision stump in machine learning and using the L2-loss function applied to real valued predictors, how could a threshold be computed?*

Exercise. 16

1. *Looking at figure 4, what does it tell us on the hyperspectral image?*
2. *Show on figure 4, the first, second and third quartiles.*

Exercise. 17 *We consider two sets* $\mathscr{N}_{\mathbf{a}}$ *and* $\mathscr{N}_{\mathbf{b}}$ *defined as the set of pixels being closer to* $\mathbf{a},\mathbf{b}$ *than of* $\mathbf{b},\mathbf{a}$*.*

1. *Show that* $\mathscr{N}_{\mathbf{a}}$ *and* $\mathscr{N}_{\mathbf{b}}$ *are segmentations of* $\mathbf{I}$ *in the sense of equation (4).*

We denote $\mathbf{X}$ *the dataset obtained using the intensities of* $\mathscr{I}$ *at the different bandwidths as defined in equation (1).*

2. *Show that there exists* U *and* b *such that* $\mathbf{1}(\mathbf{X}U \leq b)$ *is a binary column vector indicating the membership of each row to* $\mathscr{N}_{\mathbf{a}}$*. Show that* $\mathbf{1}(\mathbf{X}U > b)$ *indicates that of* $\mathscr{N}_{\mathbf{b}}$*.*

```
function im2=edge_det1(im1)
  im21=filter2([1 0; 0 -1],im1);
  im22=filter2([0 1; -1 0],im1);
  im2=abs(im21)+abs(im22);
end
```

2.2 Edges as a mean for segmentation

- $\partial\mathbf{I}$ is here a contour, that is a binary image.

- Ψ and Φ are here images whose values are angles.
- $*$ is here the 2D-convolution product. It is actually a sum-product of a sliding window. It uses here $\square$ to indicate how the sliding window is to be positioned.
- $|\ldots|$ means the absolute value when applied to a numerical value or function.
- Four examples of filtering operators to find edges: $\mathrm{F}_1, \mathrm{F}_2, \mathrm{F}_3, \mathrm{F}_4$.
- One example of a smoothing operator reducing noise: G.
- $\mathrm{PSNR}_{\mathrm{dB}}$ is a metric used in image processing.
- $\log_{10}$.

Exercise. 18 *We consider the following image*

$$I = \begin{bmatrix} 1 & 6 & 3 & 3 \\ 2 & 6 & 2 & 4 \\ 1 & 1 & 1 & 5 \\ 5 & 6 & 4 & 1 \end{bmatrix}$$

1. *Compute the resulting edge-image obtained with the magnitude of the gradient obtained using the Roberts operators.*
2. *Compute the angle of the edge detector.*

```
function fig13()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,50);
  %im=filter2([1 2 1; 2 4 2; 1 2 1]/16,im);
  im2=edge_det2(im);
  T=quantile(im2(:),0.75);
  figure(2); imshow(im2>=T);
  imwrite(im2>=T,'../../images/fig13.png');
end

function fig12()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,50);
  im=filter2([1 2 1; 2 4 2; 1 2 1]/16,im);
  im2=edge_det2(im);
  T=quantile(im2(:),0.75);
  figure(1); imshow(im2>=T);
  imwrite(im2>=T,'../../images/fig12.png');
end

function fig14()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,k1);
  im=(im-min(im(:)))/(max(im(:))-min(im(:)));
```

```
  sigma_l=1e-2:1e-2:0.6;
  psnr1_l=zeros(size(sigma_l));
  psnr2_l=zeros(size(sigma_l));
  psnr3_l=zeros(size(sigma_l));
  for sigma_=1:length(sigma_l)
    im1=im+sigma_l(sigma_)*randn(size(im));
    im2=filter2([1 2 1; 2 4 2; 1 2 1]/16,im1);
    psnr1_l(sigma_)=PSNR(im1,im);
    psnr2_l(sigma_)=PSNR(im2,im1);
    psnr3_l(sigma_)=PSNR(im2,im);
  end
  figure(1); plot(sigma_l,psnr1_l,'b-','linewidth',2,...
    sigma_l,psnr2_l,'r-','linewidth',2,...
    sigma_l,psnr3_l,'g-','linewidth',2);
  set(gca, 'FontSize', 13, 'fontName','Times');
  legend('d(I_1,I)','d(I_2,II_1)','d(I_2,I)')
  saveas(1,'../../images/fig14.png');
end

function val=PSNR(im1,im2)
  val=10*log10(1./mean((im1(:)-im2(:)).^2));
end

function fig15()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,k1);
  T1=quantile(im(:),0.75);
  figure(1); imshow(im>=T1);
  imwrite(im>=T1,'../../images/fig15a.png');
  im1=filter2([1 2 1; 2 4 2; 1 2 1]/16,im);
  im2=abs(im-im1);
  T2=quantile(im2(:),0.75);
  figure(2); imshow(im2>=T2);
  imwrite(im2>=T2,'../../images/fig15b.png');
end

function fig14()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,k1);
  im=(im-min(im(:)))/(max(im(:))-min(im(:)));
  sigma_l=1e-2:1e-2:0.6;
  psnr1_l=zeros(size(sigma_l));
  psnr2_l=zeros(size(sigma_l));
  psnr3_l=zeros(size(sigma_l));
  for sigma_=1:length(sigma_l)
    im1=im+sigma_l(sigma_)*randn(size(im));
    im2=filter2([1 2 1; 2 4 2; 1 2 1]/16,im1);
    psnr1_l(sigma_)=PSNR(im1,im);
    psnr2_l(sigma_)=PSNR(im2,im1);
    psnr3_l(sigma_)=PSNR(im2,im);
  end
```

```
  figure(1); plot(sigma_l,psnr1_l,'b-','linewidth',2,...
    sigma_l,psnr2_l,'r-','linewidth',2,...
    sigma_l,psnr3_l,'g-','linewidth',2);
  set(gca, 'FontSize', 13, 'fontName','Times');
  legend('d(I_1,I)','d(I_2,II_1)','d(I_2,I)')
  saveas(1,'../../images/fig14.png');
end

function val=PSNR(im1,im2)
  val=10*log10(1./mean((im1(:)-im2(:)).^2));
end
```

2.3 Detection of connected components

- $\mathbb{R}, \mathbb{N}$
- $\forall$ and $\exists$

Exercise. 19

1. *What neighborhood system is the pseudocode using?*
2. *How can we use the pseudocode to test is a given set is connected?*
3. *Give the intermediate values of* $\mathbf{I}'$ *when* $\mathbf{I}$ *is defined as*

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

```
function fig16()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,k1);
  T1=quantile(im(:),0.75);
  im_conn=connect_comp(im>=T1);
  figure(1); imshow(im_conn/max(im_conn(:)));
  imwrite(im_conn/max(im_conn(:)),'../../images/fig16a.png');
  area_l=count_area(im_conn);
  [area2_l,nb_l]=count2_nb(area_l);
  figure(2); loglog(area2_l,nb_l,'linewidth',2);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(2,'../../images/fig16b.png');
  im_conn2=big(im_conn,area_l,5);
  figure(3); imshow(im_conn2/max(im_conn2(:)));
  imwrite(im_conn2/max(im_conn2(:)),'../../images/fig16c.png');
end

function area_l=count_area(im_conn)
  list_l=unique(im_conn);
  if any(list_l==0) list_l=list_l(list_l~=0); end
  area_l=zeros(size(list_l,1),2);
  for list_=1:length(list_l)
```

```
    val=sum(im_conn(:)==list_l(list_));
    area_l(list_,1)=val;
    area_l(list_,2)=list_l(list_);
  end
end

function im_conn2=big(im_conn,area_l,nb)
  list_l=unique(im_conn);
  im_conn2=zeros(size(im_conn));
  for n=1:nb
    [~,ind]=max(area_l(:,1));
    num=area_l(ind,2);
    area_l(ind,1)=0;
    im_conn2=im_conn2+n*(im_conn==num);
  end
end

function [area2_l,nb_l]=count2_nb(area_l)
  area1_l=sort(area_l(:,1));
  Mx=max(area1_l);
  Mn=min(area1_l);
  area2_l=(0:200)/200*(Mx-Mn)+Mn;
  nb_l=zeros(size(area2_l));
  for area2_=1:length(area2_l)
    nb_l(area2_)=sum(area1_l>=area2_l(area2_));
  end
end
```

Exercise. 20 *Give a pseudo transforming a binary image with edges into the corresponding regions.*

```
function ex66()
  im1=im_edge();
  im2=1-im1;
  figure(1); imshow(im1);
  im_conn=connect_comp(im2);
  imwrite(im1,'../../images/ex66_fig1.png');
  figure(2); imshow(im_conn/max(im_conn(:)));
  imwrite(im_conn/max(im_conn(:)),'../../images/ex66_fig2.png');
  area_l=count_area(im_conn);
  im_conn2=big(im_conn,area_l,5);
  figure(3); imshow(im_conn2/max(im_conn2(:)));
  imwrite(im_conn2/max(im_conn2(:)),'../../images/ex66_fig3.png');
end

function im=im_edge()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im=im(:,:,50);
  im=filter2([1 2 1; 2 4 2; 1 2 1]/16,im);
  im2=edge_det2(im);
  T=quantile(im2(:),0.75);
  im=(im2>=T);
end
```

```
function area_l=count_area(im_conn)
  list_l=unique(im_conn);
  if any(list_l==0) list_l=list_l(list_l~=0); end
  area_l=zeros(size(list_l,1),2);
  for list_=1:length(list_l)
    val=sum(im_conn(:)==list_l(list_));
    area_l(list_,1)=val;
    area_l(list_,2)=list_l(list_);
  end
end

function im_conn2=big(im_conn,area_l,nb)
  list_l=unique(im_conn);
  im_conn2=zeros(size(im_conn));
  for n=1:nb
    [~,ind]=max(area_l(:,1));
    num=area_l(ind,2);
    area_l(ind,1)=0;
    im_conn2=im_conn2+n*(im_conn==num);
  end
end
```

2.4 Use of iterated algorithms

- $\sum_{n\in\mathscr{N}_0} I(n) = \sum_n I(n)\mathbf{1}(n\in\mathscr{N}_0)$
- The average is $\mu_0 = \frac{1}{|\mathscr{N}_0|}\sum_{n\in\mathscr{N}_0} I(n)$

Exercise. 21 *We consider the following image*

$$\mathscr{I} = \begin{bmatrix} 1 & 6 & 3 & 3 \\ 2 & 6 & 2 & 4 \\ 1 & 1 & 1 & 5 \\ 5 & 6 & 4 & 1 \end{bmatrix}$$

1. *Give the segmented image using the thresholding algorithm.*

```
function im2=seg_thresholding1(im1)
  M1=size(im1,1); M2=size(im1,2); center=[ceil(M1/2) ceil(M2/2)];
  T=0.5*(mean(im1([1 M1],[1 M2])(:))+im1(center(1),center(2)));
  T_old=0;
  while(abs(T_old-T)<0.1)
    mu1=sum(sum(im1.*(im1<=T)))/sum(sum(im1<=T));
    mu2=sum(sum(im1.*(im1>T)))/sum(sum(im1>T));
    T=(mu1+mu2)/2;
    T,
  end
  im2=(im1>T);
end
```

Code of figure 4

```
function fig3()
%\ref{fig:iterated_algorithms_fig3_fig4}
  x=data();
  dt=(max(x)-min(x))/1000;
  T_l=min(x):dt:max(x);
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
    val_l(T_)=eval_loss(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(1,'../../images/fig3.png');

end

function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end

function y=eval_loss_old(x,T)
  ind1=find(x<=T); x1=x(ind1);
  ind2=find(x>T); x2=x(ind2);
  y=length(ind1)*sum((x1-mean(x1)).^2)+length(ind2)*sum((x2-mean(x2)).^2);
  y=sqrt(y/(length(ind1)^2+length(ind2)^2));
end

function y=eval_loss(x,T)
  ind1=find(x<=T); x1=x(ind1);
  ind2=find(x>T); x2=x(ind2);
  y=sum((x1-mean(x1)).^2)+sum((x2-mean(x2)).^2);
  y=sqrt(y/length(x));
end
```

Code of figure 4

```
function fig4()
%%\ref{fig:iterated_algorithms_fig3_fig4}
  x=data();
  dt=(max(x)-min(x))/1000;
  T_l=min(x):dt:max(x);
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
    val_l(T_)=eval_action(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2,...
     T_l,T_l,'k:','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(1,'../../images/fig4.png');
end
```

```
function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end

function T2=eval_action(x,T1)
  ind1=find(x<=T1); x1=x(ind1);
  ind2=find(x>T1); x2=x(ind2);
  mu1=mean(x1);
  mu2=mean(x2);
  T2=(mu1+mu2)/2;
end
```

Exercise. 22 *We are going to prove formulas in steps 3 and 4 used in algorithm* `seg_thresholding1`. *We assume a function to be minimized*

$$J = \sqrt{\frac{\sum_{n\in\mathcal{N}_0}(I_n-\mu_0)^2 + \sum_{n\in\mathcal{N}_1}(I_n-\mu_1)^2}{N}}$$

1. *Show that given $\mathcal{N}_0$ and $\mathcal{N}_1$, J is minimal when*

$$\mu_0 = \frac{\sum_{n\in\mathcal{N}_0} I_n}{|\mathcal{N}_0|} \text{ and } \mu_1 = \frac{\sum_{n\in\mathcal{N}_1} I_n}{|\mathcal{N}_1|}$$

2. *Show that given μ_0 and μ_1, J is minimal when*

$$\mathcal{N}_0 = \{I_n \leq \frac{\mu_0+\mu_1}{2}\} \text{ and } \mathcal{N}_1 = \{I_n > \frac{\mu_0+\mu_1}{2}\}$$

```
function im2=seg_thresholding2(im1)
  M1=size(im1,1); M2=size(im1,2); center=[ceil(M1/2) ceil(M2/2)];
  T=0.5*(mean(im1([1 M1],[1 M2])(:))+im1(center(1),center(2)));
  mu1=sum(sum(im1.*(im1<=T)))/sum(sum(im1<=T));
  mu2=sum(sum(im1.*(im1>T)))/sum(sum(im1>T));
  x=im1(:);
  mu1_old=0; mu2_old=0;
  while(abs(mu1_old-mu1)+abs(mu2_old-mu2)<0.1)
    q=abs(x-mu1)./(abs(x-mu1)+abs(x-mu2));
    mu1=sum(q.*x)/sum(q);
    mu2=sum((1-q).*x)/sum(1-q);
  end
  q=abs(im1-mu1)./(abs(im1-mu1)+abs(im1-mu2));
  im2=q;
end
```

Exercise. 23 *Find a formula for q_n such that the second algorithm behaves like the first one.*

Code of figure 5

```
function fig7()
%%\ref{fig:iterated_algorithms_fig5_fig6_fig7}
```

```
  pkg load image
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  figure(3); imshow(mat2gray(im(:,:,k1)));
  imwrite(mat2gray(im(:,:,k1)),'../../images/fig7.png');
end
```

Code of figure 5

```
function fig5()
%%\ref{fig:iterated_algorithms_fig5_fig6_fig7}
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im2=seg_thresholding1(im(:,:,k1));
  figure(1); imshow(im2);
  imwrite(im2,'../../images/fig5.png');
end
```

Code of figure 5

```
function fig6()
%%\ref{fig:iterated_algorithms_fig5_fig6_fig7}
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  im2=seg_thresholding2(im(:,:,k1));
  figure(2); imshow(im2);
  imwrite(im2,'../../images/fig6.png');
end
```

2.5 Clustering regarded as an optimization problem

```
function fig_kmeans()
  close all
  name='fig_kmeans_';
  X=data_preparation();
  graph(name,X,[],1);
  %figure(1); plot(X(:,1),X(:,2),'+','linewidth',3);
  Y_l=kmeans(X);
  for k=1:size(Y_l,2)
    graph(name,X,Y_l(:,k),k+1);
  end
end

function X=data_preparation()
  X=0.7*randn(10,2)+ones(10,1)*[0 1];
  X=[X;randn(10,2)+ones(10,1)*[1 0]];
end

function [Y_l]=kmeans(X)
  while(1)
    ind_l=randperm(size(X,1),2);
    x0=X(ind_l(1),:); x1=X(ind_l(2),:);
    Y_l=[];
```

```
    n=0;
    while(1)
      n=n+1;
      Y=(dist2(X,x0)>  dist2(X,x1));
      if all(1==Y)||all(0==Y) break; end
      ind0=find(Y==0); ind1=find(Y==1);
      x0=sum(X(ind0,:),1)/length(ind0);
      x1=sum(X(ind1,:),1)/length(ind1);
      Y_l=[Y_l Y];
      if 1==n continue, end
      if (all(Y_l(:,end)==Y_l(:,end-1))) return; end
    end
  end
end

function graph(name,X,Y,n)
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  if ~isempty(Y)
    ind0=find(Y==0); ind1=find(Y==1);
    x0=sum(X(ind0,:),1)/length(ind0);
    x1=sum(X(ind1,:),1)/length(ind1);
    figure(n); plot(X(ind0,1),X(ind0,2),'g+','linewidth',3,X(ind1,1),X(ind1,2),'ro','linew
    text(x0(1),x0(2),'C(Y=0)','color','green'); text(x1(1),x1(2),'C(Y=1)','color','red');
    set(gca, "linewidth", 3, "fontsize", 16)
    legend('Y=0','Y=1','location','northwest'),
  else
    figure(n); plot(X(:,1),X(:,2),'+','linewidth',3);
    set(gca, "linewidth", 3, "fontsize", 16)
  end
  title(['kmeans n=',num2str(n),' \it ',name_title]),
  xlabel('x1'); ylabel('x2');
  prin(n);
end

function D=dist2(X,xa)
  D=sum((X-ones(size(X,1),1)*xa).^2,2);
end
```

Exercise. 24 *We consider a set of points* $\mathbf{X}$ *and two clusters. Two points are first randomly selected. Then the two following iterations are repeated.*

- *Each point is assigned to the closest point.*
- *Each geometric center is updated with its new and removed members.*

1. *Give the algorithm*

3 Learning regarded as an optimization problem

3.1 Optimization problem

Exercise. *We are considering the following* 2*-feature data set denoted* $\mathcal{S}_2$.

$$\begin{array}{lll} x_{11}=2 & x_{12}=0.5 & y_1=1 \\ x_{21}=1 & x_{22}=2 & y_2=0 \\ x_{31}=0 & x_{32}=0 & y_3=1 \end{array}$$

We consider a family of predictors $f_{\mathbf{a},b}$ *defined as*

$$f_{\mathbf{a},b}(\mathbf{x}) = \mathbf{1}(\mathbf{a}.\mathbf{x} \leq b)$$

with $\mathbf{a} = [a_1, a_2]$.
We define $J(a_1, a_2, b) = L(\mathcal{S}_2, f_{\mathbf{a},b})$

1. *Compute* $J(a_1, a_2, b)$ *as the sum of three quadratic expressions. And explain why* 0 *an obvious lower bound of J is likely to be reached.*
2. *Show that* $J(a_1, a_2, b) = 0$ *if this system is solved.*

$$\begin{cases} 2a_1 + 0.5a_2 - b = -1 \\ a_1 + 2a_2 - b = 1 \\ b = 1 \end{cases}$$

3. *Solve the system and show that* $a_1 = -\frac{2}{7}$, $a_2 = \frac{8}{7}$ *and* $b = 1$.

- L is here the L2-loss function.
- $f^v(\mathbf{x}) \in \mathbb{R}$ whereas $f(\mathbf{x}) \in \{0, 1\}$.

3.2 Simulated annealing

- L is different from L. It is the last best value obtained.
- $\Delta\Theta$: modification of the parameter values.

```
function [theta,tab]=simulated_annealing(cost_function,dim,option,init_value,option2)
%cost_function is the name of a cost function
%the output is a number if it is possible or NaN if it is impossible
%dim of theta
%option='silent' to suppress any display

  is_init_value=0; silence=0; store_all=0;
  if nargin>=5
    if ~silence silence=strcmp(option2,'silent'); end
    if ~store_all store_all=strcmp(option2,'store_all'); end
    if ~is_init_value is_init_value=strcmp(option2,'init_value'); end
  end
  if nargin>=4
    if ~is_init_value is_init_value=strcmp(option,'init_value'); end
  end
  if nargin>=3
    if ~silence silence=strcmp(option,'silent'); end
    if ~store_all store_all=strcmp(option,'store_all'); end
```

```
  end
  theta=zeros(dim,1); L=Inf; tab=[];
  if is_init_value
    theta=init_value(:); L=cost_function(theta);
    if store_all tab=[tab; L theta' k]; end
  else
  end
  vert_b=is_vert(cost_function,dim);
  for k=1:1e5
    r=rand(dim,1)*6;
    sigma=10.^(-r);
    if rand(1)<0.5
      delta_theta=randn(dim,1).*r;
    elseif rand(1)<0.5
      delta_theta=-theta(:)+randn(dim,1).*r;
    else
      p=ceil(rand(1)*dim);
      delta_theta=zeros(dim,1);
      delta_theta(p)=randn(1).*r(1);
    end
    try
      if vert_b
        L_try=cost_function(theta+delta_theta);
      else
        L_try=cost_function(theta'+delta_theta');
      end
      if abs(imag(L_try))>0                                    error('L imag'), end
      if NaN==L_try continue, end
      if (L_try<L)
        theta=theta+delta_theta;
        L=L_try;
        if ~silence disp(['L=',num2str(L)]), end
        if store_all tab=[tab; L theta' k]; end
      end
    catch
      error('pb'),
    end_try_catch
  end
end


function test=is_vert(fun,dim)
  try
    theta=ones(dim,1)*1e-8;
    L_try=fun(theta);
    test=1;
    return;
  catch
    L_try=fun(theta');
    test=0;
    return;
  end_try_catch;
  error('pb'),
end
```

Exercise. 26 *Give the Octave code that uses* `simulated_annealing` *to find an approximation of* $\mathbf{a}$ *and* a *of exercise 3.1 which tells*

We are considering the following 2*-feature data set denoted* $\mathcal{S}_2$.

$$\begin{array}{lll} x_{11}=2 & x_{12}=0.5 & y_1=1 \\ x_{21}=1 & x_{22}=2 & y_2=0 \\ x_{31}=0 & x_{32}=0 & y_3=1 \end{array}$$

We consider a family of predictors $f_{\mathbf{a},b}$ *defined as*

$$f_{\mathbf{a},b}(\mathbf{x}) = \mathbf{1}(\mathbf{a}.\mathbf{x} \leq b)$$

with $\mathbf{a} = [a_1, a_2]$.
We define

$$J(a_1, a_2, b) = L(\mathcal{S}_2, f_{\mathbf{a},b}) = \frac{1}{2}\sum_{n=0}^{N-1}(f^v(\mathbf{x}_n) - \widetilde{y}_n)^2$$

```
function fig_simulated_annealing()
  [~,dim,msg]=J1(NaN);
  disp(msg),
  cost_function=@(theta)J1(theta);
  [theta,tab]=simulated_annealing(cost_function,dim,'store_all');
  theta,
  graph(tab);
end

function graph(tab)
  name='fig_simulated_annealing_';
  name_title=name; name_title(name_title=='_')=' ';
  name_data=['./prg/',name,'data.mat'];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  L_l=tab(:,1);
  b_l=tab(:,2);
  a1_l=tab(:,3);
  a2_l=tab(:,4);
  k_l=tab(:,5);
  it_l=1:size(tab,1);
  keyboard,
  figure(1); semilogy(it_l,L_l,'linewidth',2); title('L');
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('number changes'); ylabel('loss function');
  saveas(1,'../../images/fig_simulated_annealing_1.png');
  b=1; a1=-2/7; a2=8/7;
  figure(2); plot(it_l,b_l-b,'b','linewidth',2,it_l,a1_l-a1,'r','linewidth',2,...
    it_l,a2_l-a2,'g','linewidth',2); title('b a1 a2');
  set(gca, "linewidth", 2, "fontsize", 16)
  legend('b-b^*','a_1-a^*_1', 'a_2-a^*_2');
  xlabel('number changes'); ylabel('error on parameter values');
  saveas(2,'../../images/fig_simulated_annealing_2.png');
  figure(3); semilogy(it_l,k_l,'linewidth',2); title('k');
  set(gca, "linewidth", 2, "fontsize", 16)
  xlabel('number changes'); ylabel('number of random trials');
  saveas(3,'../../images/fig_simulated_annealing_3.png');
```

```
end

function [J,dim,msg]=J1(theta)
  dim=3; msg='b=theta(1); a1=theta(2); a2=theta(3); ';
  if isnan(theta) J=NaN; return; end
  x1=[2 0.5]; y1=1;
  x2=[1 2]; y2=0;
  x3=[0 0]; y3=1;
  tilde=@(y)2*y-1;
  b=theta(1); a1=theta(2); a2=theta(3);
  J=(b-a1*x1(1)-a2*x1(2)-tilde(y1))^2;
  J=J+(b-a1*x2(1)-a2*x2(2)-tilde(y2))^2;
  J=J+(b-a1*x3(1)-a2*x3(2)-tilde(y3))^2;
end
```

3.3 Method of least squares

- Extended vector $\overset{\triangle}{\mathbf{x}}$
- Extended matrix $\overset{\triangle}{\mathbf{X}}$
- Unique vector $\mathbf{w}$ for linear classifier instead of $[-\mathbf{a}, b]$.
- Derivation w.r.to a row vector $\frac{\partial}{\mathbf{w}}$ or a column vector $\frac{\partial}{\mathbf{w}^T}$.
- $\mathbf{w}^*$ global minimum of the loss function.

Exercise. 27 *We consider once again exercise 3.1 to solve without using the trick of zeroing J which usually does not work.*

$$\begin{array}{lll} x_{11}=2 & x_{12}=0.5 & y_1=1 \\ x_{21}=1 & x_{22}=2 & y_2=0 \\ x_{31}=0 & x_{32}=0 & y_3=1 \end{array}$$

We consider a linear family of predictors $f_{\mathbf{a},b}$ *defined as*

$$f_{\mathbf{a},b}(\mathbf{x}) = \mathbf{1}(\mathbf{a}.\mathbf{x} \leq b)$$

with $\mathbf{a} = [a_1, a_2]$. *We consider an L2-loss function* $J(a_1, a_2, b) = L(\mathcal{S}_2, f_{\mathbf{a},b}) = \frac{1}{2}\sum_{n=1}^{N}(f^v(\mathbf{x}_n) - \widetilde{y}_n)^2$

1. *Define* $\mathbf{w}$ *with respect to* $\mathbf{a}$ *and* b *and* $\overset{\triangle}{\mathbf{x}}$ *with respect to* x_1 *and* x_2.
2. *Compute* $\mathbf{X}$, $\overset{\triangle}{\mathbf{X}}$ *and* $\overset{\triangle}{\mathbf{X}}^T\overset{\triangle}{\mathbf{X}}$.

Exercise. *3. Compute Y, $\widetilde{Y}$ and $\overset{\Delta}{\mathbf{X}}^T \widetilde{Y}$*

4. *Show that when $a_1 = -\frac{2}{7}$, $a_2 = \frac{8}{7}$ and $b = 1$, we have indeed that $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$.*
5. *Let us suppose that we have an extra sample in $\mathcal{S}_2$. What are the sizes of the different vectors and matrices involved here.*
6. *Assuming that $\mathbf{w}^*$ that cancels the J-derivative is a global minimum, show that*

$$\min_{\mathbf{w}} J(\mathbf{w}) = \widetilde{Y}^T\widetilde{Y} - \widetilde{Y}^T\overset{\Delta}{\mathbf{X}}\left(\overset{\Delta}{\mathbf{X}}^T\overset{\Delta}{\mathbf{X}}\right)^{-1}\overset{\Delta}{\mathbf{X}}^T\widetilde{Y}$$

4 Predicting the learning performances and probabilistic framework

- $\mathcal{X}$ is here a random **vector**.
- $\mathcal{P}(\ldots \& \ldots)$ joint probability
- $\mathcal{P}(\ldots | \ldots)$ conditional probability

Exercise. 28 *Let $\mathcal{Y}$ be a uniform binary random variable and $\mathcal{X}$ when conditioned to $\mathcal{Y}$ be a 2D-gaussian variable with mean $\boldsymbol{\mu}_0 \in \mathbb{R}^2$ or $\boldsymbol{\mu}_1 \in \mathbb{R}^2$ and standard deviation $\sigma_0 > 0$ or $\sigma_1 > 0$.*

1. *What is the probability that $\mathcal{Y} = 0$ on a given experiment?*
2. *What is the probability density function that $\mathcal{X} = [x_1, x_2]$ given $\mathcal{Y} = 0$ and then given $\mathcal{Y} = 1$?*
3. *We now assume that $\sigma_0 = \sigma_1 = \sigma$, show that a straight line separates points that are more likely when $\mathcal{Y} = 1$ from the more likely points when $\mathcal{Y} = 0$.*

$$f_{\mathcal{X}|\mathcal{Y}=1}(\mathbf{x}) \geq f_{\mathcal{X}|\mathcal{Y}=0}(\mathbf{x}) \Leftrightarrow (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)\mathbf{x}^T \geq (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)(\frac{1}{2}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_0)^T$$

4.1 Linear discriminant analysis

- $\overset{r}{\mathbf{x}}$ is a random vector.
- $f_{\overset{r}{\mathbf{x}}}(\mathbf{x})$ is the probability density of $\mathcal{P}(\overset{r}{\mathbf{x}})$.
- $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ are the mean of the probability distributions of classes 0 and 1. They are estimated using averaging operators on the training set. Their estimates is $\widehat{\boldsymbol{\mu}}_0$ and $\widehat{\boldsymbol{\mu}}_1$.
- $\boldsymbol{\Sigma}$ is the common covariance matrix of both Gaussian probability distributions. It is estimated using the whole training set. Its estimation is denoted $\widehat{\boldsymbol{\Sigma}}$.
- $\det(\mathbf{A})$ is the determinant of $\mathbf{A}$, it is a scalar.

Exercise. 29 *We consider here a data set defined by a probability distribution.*

$$P(y=0)=P(y=1)=0.5 \text{ and } \begin{cases} f_{\mathbf{x}|y=0}(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_0)(\mathbf{x}-\boldsymbol{\mu}_0)^T} \\ f_{\mathbf{x}|y=1}(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_1)(\mathbf{x}-\boldsymbol{\mu}_1)^T} \end{cases}$$

with $\boldsymbol{\mu}_0=[1,0], \boldsymbol{\mu}_1=[0,1]$ *and* $\sigma=2$.

1. *Write an algorithm to check that these expressions are probability distributions. Use the independence between the two components to reduce the numerical complexity.*
   $\int_{x_1}\int_{x_2} f(x_1)f(x_2)dx_1dx_2 = \int_{x_1} f(x_1)dx_1 \int_{x_2} f(x_2)dx_2$
2. *Show that with this model,* $y=1$ *is more likely than* $y=0$ *iff*

$$\boldsymbol{\mu}_0\boldsymbol{\mu}_0^T - \boldsymbol{\mu}_1\boldsymbol{\mu}_1^T - (\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)\mathbf{x}^T \geq 0$$

3. *Draw in the feature space the domains for which* $y=1$ *or* $y=0$ *is more likely.*

4.2 Predicting the true probabilities

- $\text{A} \rightarrow \text{B}$: means that

$$\mathcal{P}(\text{A},\text{B}) = \mathcal{P}(\text{B}|\text{A})\mathcal{P}(\text{A})$$

- $\binom{n}{p}$ means the number of different subsets of size p that can be drawn out of a set of size n.

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

Exercise. 30 *We assume here an experiment of 12 samples, 6 labeled positively and 6 negatively. We observed for each label, that* 5 *of them are correctly predicted.*

1. *Write an algorithm computing an approximation of the probability distributions that could best explain this experiment: the probability of a negative label to be correctly labeled* $f_0(p)$ *and that of a positive to be correctly labeled* $f_1(p)$.
2. *Given* p_0 *and* p_1, *and a column vector* $Y^T = [0,0,0,0,0,0,1,1,1,1,1,1]$, *show that the probability to have* $\widehat{Y}$ *consistent with the confusion matrix is*

$$\binom{6}{1}p_0^5(1-p_0)\times\binom{6}{1}p_1^5(1-p_1)$$

4.3 Prior and Bayes formula

- $\neg\text{A}$ is the alternative event to A.

```
function fig_modeling_prior()
  name='fig_modeling_prior_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  time=10000;
  if ~exist(name_data)
    [p_l,fZ0_l,fZ1_l,n]=prior(time);
    save(name_data,'p_l','fZ0_l','fZ1_l','n');
```

```
    nZ=n;
  else
    load(name_data);
    nZ=n;
  end
  figure(1); plot(p_l,fZ0_l/sum(fZ0_l),'linewidth',3,p_l,fZ1_l/sum(fZ1_l),'linewidth',3)
  title(['Prior n=',num2str(n),' \it ',name_title]),
  set(gca, "linewidth", 3, "fontsize", 12)
  legend('f_0(p)','f_1(p)','location','northwest'),
  prin(1);
  name1='fig_modeling_inferences_';
  name1_data=['./prg/',name1,'data.mat'];
  S1=load(name1_data);
  fZh_l=[(1-S1.Ep0)*(1-S1.Ep1),  (1-S1.Ep0)*S1.Ep1+S1.Ep0*(1-S1.Ep1)+(1-S1.Ep0)*(1-S1.Ep1
  [fZ_prior,acc_l]=use_prior(p_l,fZ0_l,fZ1_l);
  name2='fig_testing_inferences_';
  name2_data=['./prg/',name2,'data.mat'];
  S2=load(name2_data);
  bar(S2.acc_l',[S2.fZ_l;fZ_prior;fZh_l]');
  title(['C_{11}=C_{22}=5; C_{12}=C_{21}=1; n=',num2str(nZ),' \it ',name_title]),
  set(gca, "linewidth", 3, "fontsize", 12)
  legend('A','A prior','A model','location','northwest'),
  prin(2);
  [fZ_ML,acc_l]=use_prior(5/6,1,1);
  bar(S2.acc_l',[S2.fZ_l;fZ_ML;fZ_prior;fZh_l]');
  title(['C_{11}=C_{22}=5; C_{12}=C_{21}=1; n=',num2str(nZ),' \it ',name_title]),
  set(gca, "linewidth", 3, "fontsize", 12)
  legend('A','Max Likelihood','A prior','A model','location','northwest'),
  prin(3);



end

function [fZ_prior,acc_l]=use_prior(p_l,fZ0_l,fZ1_l)
  acc_l=[0 0.5 1]; fZ_prior=zeros(1,3);
  fZ_prior(3)=(sum(p_l.^5.*(1-p_l).*fZ0_l.*p_l)/sum(p_l.^5.*(1-p_l).*fZ0_l))*...
    (sum(p_l.^5.*(1-p_l).*fZ1_l.*p_l)/sum(p_l.^5.*(1-p_l).*fZ1_l));
  fZ_prior(2)=(sum(p_l.^5.*(1-p_l).*fZ0_l.*p_l)/sum(p_l.^5.*(1-p_l).*fZ0_l))*...
    (sum(p_l.^5.*(1-p_l).*fZ1_l.*(1-p_l))/sum(p_l.^5.*(1-p_l).*fZ1_l))+...
    (sum(p_l.^5.*(1-p_l).*fZ0_l.*(1-p_l))/sum(p_l.^5.*(1-p_l).*fZ0_l))*...
    (sum(p_l.^5.*(1-p_l).*fZ1_l.*(p_l))/sum(p_l.^5.*(1-p_l).*fZ1_l));
  fZ_prior(1)=(sum(p_l.^5.*(1-p_l).*fZ0_l.*(1-p_l))/sum(p_l.^5.*(1-p_l).*fZ0_l))*...
    (sum(p_l.^5.*(1-p_l).*fZ1_l.*(1-p_l))/sum(p_l.^5.*(1-p_l).*fZ1_l));
  fZ_prior=fZ_prior/sum(fZ_prior);
end

function [p_l,fZ0_l,fZ1_l,n]=prior(time)
  [p_l,fZ0_l,fZ1_l]=start_fZ();
  tic,
  n=0;
  while(1)
    n=n+1;
    [mu0,mu1,sigma0,sigma1,w,b]=draw_pb();
```

```
    y=zeros(1e3,1);
    yh=draw_data(y,mu0,mu1,sigma0,sigma1,w,b);
    p0=mean(y==yh);
    fZ0_l=adapt(fZ0_l,p0,p_l);
    y=ones(1e3,1);
    yh=draw_data(y,mu0,mu1,sigma0,sigma1,w,b);
    p1=mean(y==yh);
    fZ1_l=adapt(fZ1_l,p1,p_l);
    if toc>time break; end
  end
  fZ0_l=fZ0_l/sum(fZ0_l);fZ1_l=fZ1_l/sum(fZ1_l);
end


function [p_l,fZ0_l,fZ1_l]=start_fZ()
  Q=1e-2;
  p_l=0:Q:1;
  fZ0_l=zeros(size(p_l));
  fZ1_l=fZ0_l;
end

function fZ_l=adapt(fZ_l,p,p_l)
  Q=p_l(2)-p_l(1);
  p_=1+round(p/Q);
  fZ_l(p_)=fZ_l(p_)+1;
end


function yh=draw_data(y,mu0,mu1,sigma0,sigma1,w,b)
  if ~(size(y,2)<=1)                                    error('draw_data'), end
  yh=zeros(size(y));
  for k=1:size(y,1)
    if 1==y(k)
      x=mu0+sigma0*randn(1);
    else
      x=mu1+sigma1*randn(1);
    end
    yh(k)=(b-w*x'>=0);
  end
end

function [mu0,mu1,sigma0,sigma1,w,b]=draw_pb()
  mu0=randn(1,2);
  mu1=randn(1,2);
  sigma0=rand(1);
  sigma1=rand(1);
  w=randn(1,2);
  b=randn(1);
end
```

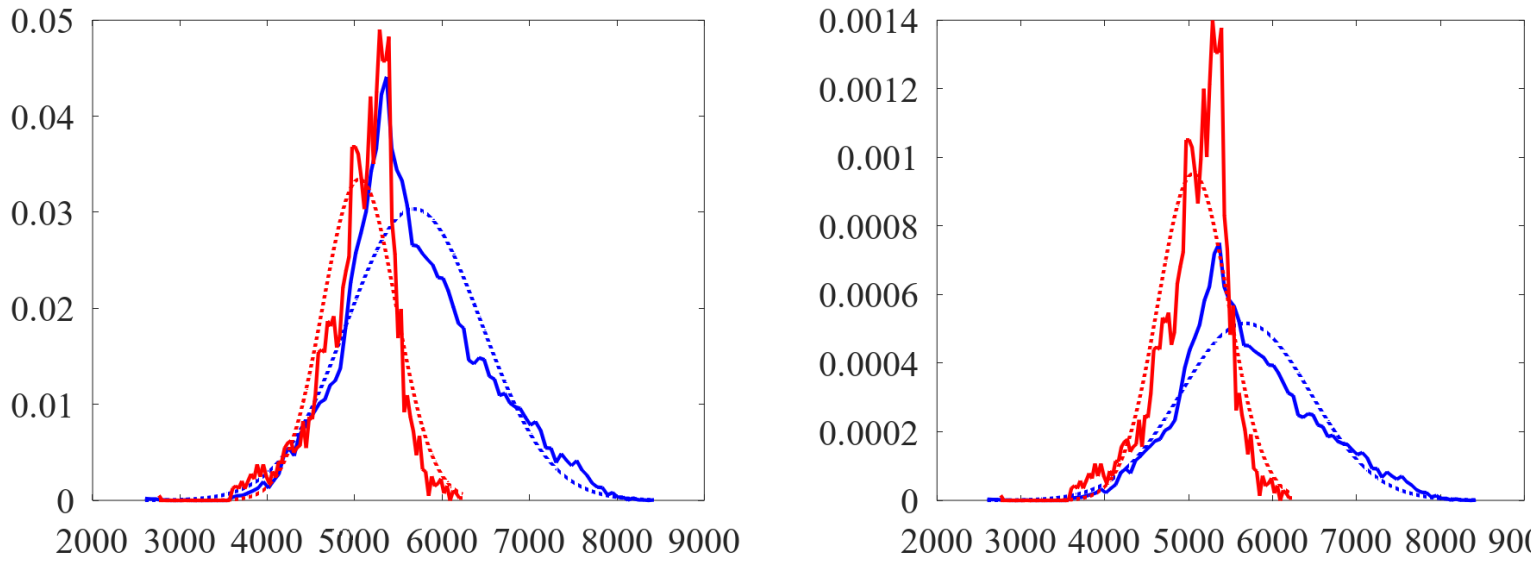


Figure 5: Empirical distributions of the bandwidth number 50 considering all pixels in blue and only pixels showing soybean in red. The dotted curves are the approximate Gaussian distributions.

5 More in depth with probabilities

5.1 Probabilities

Exercise. 34

1. *Write the pseudo-code of an algorithm yielding figure 5, empirical distributions are such that their sums equal* 1.
2. *Write the pseudo-code of an algorithm yielding figure 5, empirical distributions are such that their approximate integral equal* 1.

```
function ex54()
  question1and2();

end

function question1and2()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  gauss=@(x,mu,sigma)exp(-(x-mu).^2/sigma^2/2)/sqrt(2*pi)/sigma;
  sum1=@(y)y/sum(y);
  sum2=@(x,y)y/sum(diff(x).*(0.5*y(1:end-1)+0.5*y(2:end)));

  k1=50;
  im=im(:,:,k1);
  [Ng,g_l]=hist(im(:),100);
  moy1=mean(im(:)); sigma1=std(im(:));
  im_c=gt_im; im2=im(gt_im==1);
  [Ng2,g2_l]=hist(im2(:),100);
  moy2=mean(im2(:)); sigma2=std(im2(:));
  figure(1); plot(g_l,Ng/sum(Ng),'b-','linewidth',2,...
    g_l,sum1(gauss(g_l,moy1,sigma1)),'b:','linewidth',2,...
    g2_l,Ng2/sum(Ng2),'r-','linewidth',2,...
    g2_l,sum1(gauss(g2_l,moy2,sigma2)),'r:','linewidth',2);
```

```
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/ex54_fig1.png');
  figure(3); plot(g_l,Ng,'b-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(3,'../../images/ex54_fig3.png');
  figure(2); plot(g_l,sum2(g_l,Ng),'b-','linewidth',2,...
    g_l,gauss(g_l,moy1,sigma1),'b:','linewidth',2,...
    g2_l,sum2(g2_l,Ng2),'r-','linewidth',2,...
    g2_l,gauss(g2_l,moy2,sigma2),'r:','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(2,'../../images/ex54_fig2.png');
  figure(4); plot(g_l,Ng/sum(Ng),'b-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(4,'../../images/ex54_fig4.png');
  figure(5); plot(g_l,sum2(g_l,Ng),'b-','linewidth',2);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(5,'../../images/ex54_fig5.png');

end

function im_c=gt_im
  T=load('../dataset/Indian_pines_gt.mat');
  im_gt=T.indian_pines_gt;
  im_c=(im_gt==10)|(im_gt==11)|(im_gt==12);
end
```

Exercise. 35 *Let us call* $\mathtt{X}$ *and* $\mathtt{Y}$ *two empirical distributions obtained using the intensity values at bandwidth number* 50 *and conditionally to being actually soybean (i.e. labels* $10, 11, 12$ *of the groundtruth map).*

1. *Transform* $\mathtt{X}$ *and* $\mathtt{Y}$ *into centered and normalized random variables denoted* $\mathtt{X}_r$ *and* $\mathtt{Y}_r$.
2. *Plot as a function of* $c \in [0, 2]$ *the left side of equation (6) for* $\mathtt{X}_r$ *and* $\mathtt{Y}_r$. *Plot the right side of equation (6) and that of equation (5).*

Exercise. 36 *A simple proof of the Chebychev arises from the following steps.*

1. *Prove the Markov inequality which states*

$$\mathcal{P}[|\mathtt{X}| \geq c] \leq \frac{\mathcal{E}|\mathtt{X}|}{c}$$

   To do so, introduce a new random variable $\mathtt{Y} = c\mathbf{1}(|\mathtt{X}| \geq c)$ *and show that it is upper bounded by* $\mathtt{X}$ *and compute its expectancy.*

2. *Apply the Markov inequality to* $\mathtt{Z} = (\mathtt{X} - \mathcal{E}\mathtt{X})^2$.

5.2 Using Gaussians

- $g_{\mu,\sigma}(x)$ is the Gaussian deterministic function.
- L is here the likelihood, it is used as the opposite of a loss function.

Exercise. 37

1. *Prove that step 3 in* `seg_thresholding3.m` *is*

$$\mathrm{T} = \frac{\sigma_1\mu_0 + \sigma_0\mu_1}{\sigma_0 + \sigma_1} \text{ and } \mathscr{N}_0 = \{n | I_n \leq \mathrm{T}\}$$

2. *Prove that in steps 5, 6, μ_0 and μ_1 should the average of samples in $\mathscr{N}_0$ and $\mathscr{N}_1$.*

3. *Prove that in steps 5, 6, σ_0 and σ_1 should the standard deviation of samples in $\mathscr{N}_0$ and $\mathscr{N}_1$.*

Code of figure 7

```
function fig9()
%%\ref{fig:gaussian_fig9_fig8}
  x=data();
  dt=(max(x)-min(x))/1000;
  T_l=min(x)+dt:dt:max(x)-dt;
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
    val_l(T_)=eval_loss(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/fig9.png');
end

function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end


function y=eval_loss(x,T)
  ind1=find(x<=T); x1=x(ind1);
  ind2=find(x>T); x2=x(ind2);
  mu1=mean(x1);
  mu2=mean(x2);
  sigma1=sqrt(mean((x1-mu1).^2));
  sigma2=sqrt(mean((x2-mu2).^2));
  y=sum(log(fgauss(x1,mu1,sigma1)))+sum(log(fgauss(x2,mu2,sigma2)));
end

function p=fgauss(x,mu,sigma)
  p=1/sqrt(2*pi)/sigma*exp(-0.5*(x-mu).^2/sigma^2);
end
```

Code of figure 7

```
function fig8()
%%\ref{fig:gaussian_fig9_fig8}
  x=data();
```

```
  dt=(max(x)-min(x))/1000;
  T_l=min(x):dt:max(x);
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
    val_l(T_)=eval_action(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2,...
     T_l,T_l,'k:','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/fig8.png');
end

function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end

function T2=eval_action(x,T1)
  ind1=find(x<=T1); x1=x(ind1);
  ind2=find(x>T1); x2=x(ind2);
  mu1=mean(x1);
  mu2=mean(x2);
  sigma1=sqrt(mean((x1-mu1).^2));
  sigma2=sqrt(mean((x2-mu2).^2));
  T2=(sigma2*mu1+sigma1*mu2)/(sigma1+sigma2);
end
```

Exercise. 38

1. *Using right of figure 7, define the catchment areas (a.k.a. basins of attraction): the set of values of* T *such that the algorithm converges to a given value.*

5.3 Probabilities as a loss function designer

- Entropy: $\mathcal{H}(p) = -\sum_{x_i} p(x_i) \ln p(x_i) \geq 0$ There exists also a differential definition of $\mathcal{H}$.
- KL-divergence: $\mathcal{D}_{KL}(q||p) = \sum_{x_i} q(x_i) \ln \frac{q(x_i)}{p(x_i)} \geq 0$ There exists also a differential definition of $\mathcal{D}_{KL}$.
- ELBO $= \ln \left(\frac{p(x,y)}{q(y)} \right)$

Exercise. 39 *We consider a statistical model for a binary classification problem:*

- *The intensity of each pixel follows a Gaussian random variable.*
- *There is a unique standard deviation σ.*
- *The mean value depending on its class membership μ_0, μ_1.*
- *Conditionally to their classes, the random variables are independent.*

We use this model to infer the parameters' values involved in the model and the hidden parameters by observing only the pixel values.

1. *List the variables whose values are known and those whose values are to find by maximizing the likelihood.*
2. *Write the likelihood of a given pixel's intensity and that of all N pixels, assuming we know which pixels follows which Gaussian variable.*

Exercise. 40 *We consider the statistical model of exercise 39*

1. *Write the prior probability of $\mathtt{Y}_n$ knowing parameters of the last iteration (i.e. t-iteration).*
2. *Write the posterior probability of $\mathtt{Y}_n$ knowing parameters of the last iteration (i.e. t-iteration) and using the pixel intensity values.*
3. *Write the expectation step of the E-M algorithm, assuming $\mu_1 > \mu_0$.*
4. *Write the maximization step of the E-M algorithm.*

Code of figure 8

```
function fig10()
%%\ref{fig:algo_probability_fig10_fig11}
  x=data();
  dt=(max(x)-min(x))/1000;
  T_l=min(x)+dt:dt:max(x)-dt;
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
     val_l(T_)=eval_loss(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 20, 'fontName','Times');
  saveas(1,'../../images/fig10.png');
end

function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end


function y=eval_loss(x,T)
  N=length(x);
```

```
  ind0=find(x<=T); x0=x(ind0);
  ind1=find(x>T); x1=x(ind1);
  mu0=mean(x0); mu1=mean(x1); sigma=std(x);
  qn=length(ind0)/length(x);
  qnp=qn.*fgauss(x,mu0,sigma)./(qn.*fgauss(x,mu0,sigma)+...
    (1-qn).*fgauss(x,mu1,sigma));
  mu0=sum(qnp.*x)/sum(qnp);
  mu1=sum((1-qnp).*x)/sum((1-qnp));
  sigma=sqrt((sum(qnp.*(x-mu0).^2)+sum((1-qnp).*(x-mu1).^2))/N);
  y=sum(log(qnp.*fgauss(x,mu0,sigma)))+sum(log((1-qnp).*fgauss(x,mu1,sigma)));
end

function p=fgauss(x,mu,sigma)
  p=1/sqrt(2*pi)/sigma*exp(-0.5*(x-mu).^2/sigma^2);
end
```

Code of figure 8

```
function fig11()
%%\ref{fig:algo_probability_fig10_fig11}
  x=data();
  dt=(max(x)-min(x))/1000;
  T_l=min(x):dt:max(x);
  val_l=zeros(size(T_l));
  for T_=1:length(T_l)
    val_l(T_)=eval_action(x,T_l(T_));
  end
  figure(1); plot(T_l,val_l,'b-','linewidth',2,...
     T_l,T_l,'k:','linewidth',2);
  %axis([0 3.5 0 1]);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(1,'../../images/fig11.png');
end

function x=data()
  T=load('../dataset/Indian_pines_corrected.mat');
  im=T.indian_pines_corrected;
  k1=50;
  x=im(:,:,k1);
  x=x(:);
end



function y=eval_loss(x,T)
  N=length(x);
  ind0=find(x<=T); x0=x(ind0);
  ind1=find(x>T); x1=x(ind1);
  mu0=mean(x0); mu1=mean(x1); sigma=std(x);
  qn=length(ind0)/length(x);
  qnp=qn.*fgauss(x,mu0,sigma)./(qn.*fgauss(x,mu0,sigma)+...
    (1-qn).*fgauss(x,mu1,sigma));
  mu0=sum(qnp.*x)/sum(qnp);
  mu1=sum((1-qnp).*x)/sum((1-qnp));
  sigma=sqrt((sum(qnp.*(x-mu0).^2)+sum((1-qnp).*(x-mu1).^2))/N);
  y=sum(log(qnp.*fgauss(x,mu0,sigma)))+sum(log((1-qnp).*fgauss(x,mu1,sigma)));
```

```
end

function p=fgauss(x,mu,sigma)
  p=1/sqrt(2*pi)/sigma*exp(-0.5*(x-mu).^2/sigma^2);
end

function T2=eval_action(x,T1)
  N=length(x);
  ind0=find(x<=T1); x0=x(ind0);
  ind1=find(x>T1);  x1=x(ind1);
  mu0=mean(x0);
  mu1=mean(x1);
  sigma=std(x);
  qn=length(ind0)/length(x);
  qnp=qn.*fgauss(x,mu0,sigma)./(qn.*fgauss(x,mu0,sigma)+...
    (1-qn).*fgauss(x,mu1,sigma));
  mu0=sum(qnp.*x)/sum(qnp);
  mu1=sum((1-qnp).*x)/sum((1-qnp));
  sigma=sqrt((sum(qnp.*(x-mu0).^2)+sum((1-qnp).*(x-mu1).^2))/N);
  T2=(mu0+mu1)/2;
end
```

Exercise. 41 *We consider the following pseudo-code, what is the probability distribution that is being sampled.*

Require: N

Ensure: $I_0 \dots I_{N-1}$

1: ***for*** *n=0:N-1* ***do***
2: *Draw k a binary integer*
3: *Draw x a random value using $\mathcal{N}(0,1)$*
4: ***if*** $k == 0$ ***then***
5: $I_n = x$
6: $I_n = x + 3$

Exercise. 42 *Write a pseudocode simulating $\mathcal{P}(\sigma)$ and $\mathcal{Q}(\sigma|\sigma^{(t)})$*

```
function ex61()
  x=data();
  sigma_l=0.25:1e-2:4;
  p1_l=c_p_th(x,sigma_l);
  [p2_l,err_l,sigma2_l,p2_init_l,sigma_init_l,err_init_l]=c_p_EM(x,sigma_l);
  qp=(x<1.5);
  N=length(x);
  figure(1);
  semilogy(sigma_l,p1_l,'b-','linewidth',2,...
    sigma_l,p2_l,'r-','linewidth',2,...
    sigma_l,p2_init_l,'m-','linewidth',0.7);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(1,'../../images/ex61_fig1.png');
  figure(2);
  plot(sigma_l,err_l,'r-','linewidth',2,...
    sigma_l,err_init_l,'m-','linewidth',0.7);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(2,'../../images/ex61_fig2.png');
  figure(3);
```

```
  plot(sigma_l,sigma2_l,'g-','linewidth',2,...
    sigma_l,sigma_init_l,'c-','linewidth',0.7);
  set(gca, 'FontSize', 13, 'fontName','Times');
  saveas(3,'../../images/ex61_fig3}.png');
end

function x=data()
  N=1e4;
  k=(rand(1,N)>0.5);
  x=randn(1,N)+3*k;
end

function [p_l,err_l,sigma2_l,p_init_l,sigma_init_l,err_init_l]=c_p_EM(x,sigma_l)
  p_l=zeros(size(sigma_l));
  err_init_l=zeros(size(sigma_l));
  for sigma_=1:length(sigma_l)
    y=(rand(size(x))>0.5);
    y_old=zeros(size(y));
    sigma=0.5+4*rand(1);
    q=mean(y==0);
    qp=q*fgauss(x,0,sigma)...
      ./(q*fgauss(x,0,sigma)+(1-q)*fgauss(x,3,sigma));
    p_init_l(sigma_)=-length(x)*log(sqrt(2*pi)*sigma_l(sigma_))...
     -sum(qp.*(x-0).^2)./sigma_l(sigma_).^2/2-sum((1-qp).*(x-3).^2)/sigma_l(sigma_).^2/2;
    sigma_init_l(sigma_)=sqrt(sum(qp.*(x-0).^2+(1-qp).*(x-3).^2)/length(x));
    y_init_l=(qp<=0.5);
    err_init_l(sigma_)=mean(y_init_l~=(x>1.5));
    while(any(y_old~=y))
      y_old=y;
      q=mean(y==0);
      qp=q*fgauss(x,0,sigma)...
        ./(q*fgauss(x,0,sigma)+(1-q)*fgauss(x,3,sigma));
      sigma=sqrt(sum(qp.*(x-0).^2+(1-qp).*(x-3).^2)/length(x));
      y=(qp<0.5);
    end
    N=length(x);
    q=mean(y==0);
    qp=q*fgauss(x,0,sigma)...
      ./(q*fgauss(x,0,sigma)+(1-q)*fgauss(x,3,sigma));

    p_l(sigma_)=-length(x)*log(sqrt(2*pi)*sigma_l(sigma_))...
     -sum(qp.*(x-0).^2)./sigma_l(sigma_).^2/2-sum((1-qp).*(x-3).^2)/sigma_l(sigma_).^2/2;
    err_l(sigma_)=mean(y~=(x>1.5));
    sigma2_l(sigma_)=sigma;
  end
end

function p_l=c_p_th(x,sigma_l)
  p_l=zeros(size(sigma_l));
  for sigma_=1:length(sigma_l)
    y_l=0.5*fgauss(x,0,sigma_l(sigma_))+0.5*fgauss(x,3,sigma_l(sigma_));
    p_l(sigma_)=sum(log(y_l));
  end
end
```

```
function p=fgauss(x,mu,sigma)
  p=1/sqrt(2*pi)/sigma*exp(-0.5*(x-mu).^2/sigma^2);
end
```

Exercise. 43 *We consider again the statistical model of exercise 39. We consider a discrete probability distribution depending on a parameter T for a given sample n, denoting here x_n and y_n as x and y.*

$$q(y) = \begin{cases} \mathbf{1}(y=0) & \text{if} \quad x \leq T \\ \mathbf{1}(y=1) & \text{if} \quad x > T \end{cases}$$

1. *Show that q is indeed a probability distribution whose entropy is zero.*
2. *Given the posterior of the probability distribution computed in exercise 40,*

$$p'(y=0) = \frac{\alpha g_{0,\sigma}(x)}{\alpha g_{0,\sigma}(x) + g_{3,\sigma}(x)} \text{ and } p'(y=1) = 1 - p'(y=0)$$

where σ denotes the estimated value $\sigma^{(t)}$. Compute the KL-divergence between the q and p'.

Exercise.

3. *Show that*

$$p'(y=0) > \frac{1}{2} \Leftrightarrow x < \frac{3}{2} + \frac{\sigma^2 \ln(\alpha)}{3}$$

4. *Show that the KL-divergence is minimized when $T = \frac{3}{2} + \frac{\sigma^2 \ln(\alpha)}{3}$.*

We now consider the whole dataset.

5. *Compute ELBO $- \mathcal{H}_{\mathcal{Q}}$*
6. *Compute $\sigma^{(t+1)}$ maximizing ELBO $- \mathcal{H}_{\mathcal{Q}}$ as a function of $\mathcal{N}_0^{(t)}$ and $\mathcal{N}_1^{(t)}$ which are the set containing the samples $y_n^{(t)} = 0$ and $y_n^{(t)} = 1$. Show that*

$$\sigma^{(t+1)} = \sqrt{\frac{1}{N}\left(\sum_{n \in \mathcal{N}_0^{(t)}} x_n^2 + \sum_{n \in \mathcal{N}_1^{(t)}} (x_n - 3)^2\right)}$$

6 Curse of dimensionality, regularization and sparsity

6.1 Data preparation

- diag($\mathbf{A}$) is a diagonal matrix composed of the diagonal components of $\mathbf{A}$.
- diag($\mathbf{A}$)$^\alpha$ when the diagonal components are positive is equal to A_{ii}^α.

```
function fig_centering()
  name='fig_centering';
  ax=[-2.2 2.2 -2.2 2.2];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  X=2*rand(1e4,2);
  ind=find((X(:,1)<=1)|(X(:,2)<=1));
  figure(1); plot(X(ind,1),X(ind,2),'.');
  axis(ax);
  pbaspect ([1 1 1]);
```

```
  xlabel('x_1'),ylabel('x_2'),
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(1);
  Xc=[mean(X(ind,1)),mean(X(ind,2))];
  X1=X(ind,:)-Xc;
  mean(X1),
  figure(2); plot(X1(:,1),X1(:,2),'.');
  axis(ax);
  pbaspect ([1 1 1]);
  xlabel('x_1'),ylabel('x_2'),
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(2);
  alpha=1./[sqrt(mean(X1(:,1).^2)) sqrt(mean(X1(:,2).^2))];
  X2=X1*diag(alpha);
  figure(3); plot(X2(:,1),X2(:,2),'.');
  axis(ax);
  pbaspect ([1 1 1]);
  xlabel('x_1'),ylabel('x_2'),
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(3);
  std(X2), mean(X2),
end
```

Exercise. **44** *Let* $\mathbf{X}$ *be a feature matrix. Show that there exists* β_f *such that* $\mathbf{X}' = \mathbf{X} - [\beta_1 \dots \beta_F]$ *is centered.*

Exercise. **45** *Given a data set* $X = [x_{nf}]$*, compute a value* α_f *such that*

$$\frac{1}{N}\sum_{n=1}^{N} {x'}_{nf}^2 = 1$$

where $x'_{nf} = \alpha_f x_{nf}$

Exercise. **46** *The exercises 44 and 45 provided formulas to center and normalize the samples in the feature space. The goal here is to express these transformations with matrices. An interesting side-effect is the simplification of the implementation.*
We consider here a dataset described with a matrix $\mathbf{X}$ *of size* $N \times F$ *and a column vector* Y *of size* $N \times 1$*.*

1. *Define a matrix* $\mathbf{H}$ *of size* $N \times N$ *such that* $\mathbf{HX}$ *is centered (i.e. the sums of each column of* $\mathbf{HX}$ *are null).*
2. *Show that* $\mathbf{HX}\left(diag(\mathbf{X}^T\mathbf{H}^2\mathbf{X})\right)^{-\frac{1}{2}}$ *is centered and normalized.*
3. *Write the Matlab/Octave implementation of* $\mathbf{HX}\left(diag(\mathbf{X}^T\mathbf{H}^2\mathbf{X})\right)^{-\frac{1}{2}}$

$$(diag(A))_{ij} = a_{ij}\mathbf{1}(j=i) \text{ and } ((diag(A))_{ij})^{-\frac{1}{2}} = \frac{1}{\sqrt{a_{ii}}}\mathbf{1}(j=i)$$

6.2 Feature construction

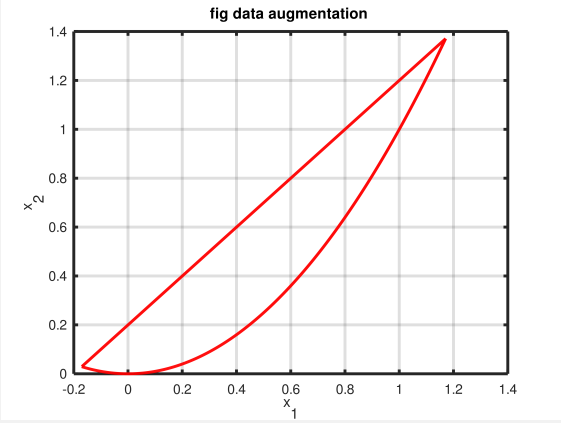
- $\mathscr{F}$ and $\overset{\omega}{\mathscr{F}}$
- $\mathbf{x}$ and $\overset{\omega}{\mathbf{x}}$
- $\| \ \|$ and $\| \ \|_\omega$

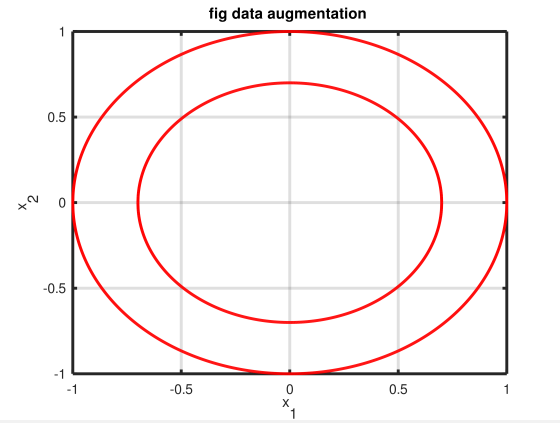
- ω

```
function fig_data_augmentation()
  name='fig_data_augmentation_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  N=100;
  [X,Y]=data1(N);
  ind1=find(Y==1); ind0=find(Y==0);
  figure(1); plot(X(ind1,1),X(ind1,2),'g+','linewidth',3,X(ind0,1),X(ind0,2),'bo','linewi
  title(name_title),
  xlabel('x_1'),ylabel('x_2'),
  legend('y=1','y=0');
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(1);
  [X,Y]=data2(N);
  ind1=find(Y==1); ind0=find(Y==0);
  figure(2); plot(X(ind1,1),X(ind1,2),'g+','linewidth',3,X(ind0,1),X(ind0,2),'bo','linewi
  title(name_title),
  xlabel('x_1'),ylabel('x_2'),
  legend('y=1','y=0');
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(2);
  [x1_l,x2_l]=contour1();
  figure(3); plot(x1_l,x2_l,'r-','linewidth',3);
  title(name_title),
  xlabel('x_1'),ylabel('x_2'),
  grid,
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(3);
  [x11_l,x21_l]=contour2(0.7);
  [x12_l,x22_l]=contour2(1);
  figure(4); plot(x11_l,x21_l,'r-','linewidth',3,x12_l,x22_l,'r-','linewidth',3);
  title(name_title),
  xlabel('x_1'),ylabel('x_2'),
  grid,
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(4);

end

function [X,Y]=data1(N);
  X=0.35*randn(N,2)+0.3*ones(N,1)*[1 1];
  Y=zeros(N,1);
  ind=find((X(:,2)>=X(:,1).^2)&(X(:,2)<=0.2+X(:,1)));
  Y(ind)=1;
end

function [x1_l,x2_l]=contour1()
  x1_lim=roots([1 -1 -1/5]);
  x1a_l=min(x1_lim):1e-3:max(x1_lim);
  x2a_l=x1a_l.^2;
  x1b_l=max(x1_lim):-1e-3:min(x1_lim);
  x2b_l=0.2+x1b_l;
```

```
  x1_l=[x1a_l x1b_l];
  x2_l=[x2a_l x2b_l];
end


function [X,Y]=data2(N);
  X=0.35*randn(N,2)+0.5*ones(N,1)*[1 1];
  Y=zeros(N,1);
  r=sqrt(X(:,1).^2+X(:,2).^2);
  ind=find((r>=0.7)&(r<=1));
  Y(ind)=1;
end

function [x1_l,x2_l]=contour2(r)
  x1a_l=-r:1e-3:r;
  x2a_l=sqrt(r^2-x1a_l.^2);
  x1b_l=r:-1e-3:-r;
  x2b_l=-sqrt(r^2-x1b_l.^2);
  x1_l=[x1a_l x1b_l];
  x2_l=[x2a_l x2b_l];
end
```

Exercise. 47 *The goal is to write linear classifiers corresponding to these domains in the feature space composed of two dimensions.*





1. *Write equations delimiting the area of the left figure.*
2. *Write equations delimiting the area of the right figure.*
3. *Define the added features.*

Exercise. 4. *Define two linear classifiers bounding the left area using also the added features.*

$$f(\vec{\mathbf{x}}) = \mathbf{1}(b_1 - \mathbf{a}_1.\vec{\mathbf{x}})\mathbf{1}(b_2 - \mathbf{a}_2.\vec{\mathbf{x}})$$

with $f(\vec{\mathbf{x}}) = 1$ *iff* $\mathbf{x}$ *is inside the domain.*

5. *Define two linear classifiers bounding the right area using also the added features.*

$$f(\vec{\mathbf{x}}) = \mathbf{1}(b_1 - \mathbf{a}_1.\vec{\mathbf{x}})\mathbf{1}(b_2 - \mathbf{a}_2.\vec{\mathbf{x}})$$

with $f(\vec{\mathbf{x}}) = 1$ *iff* $\mathbf{x}$ *is inside the domain.*

```
function fig_norm_feature_construction4()
```

```
%for  vectors whose norms have other values$
  name='fig_norm_feature_construction4_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  norm_value_l=0.3:0.3:3;
  F=2;
  tab_l=[];
  if ~exist(name_data)
    for cpt=1:length(norm_value_l)
      norm_value=norm_value_l(cpt);
      x_max=simulated_annealing(@(x)1e4-fun1(x,norm_value),2,'silent');
      rho_max=fun1(x_max,norm_value);
      x_min=simulated_annealing(@(x)fun1(x,norm_value),2,'silent');
      rho_min=fun1(x_min,norm_value);
      if ~(rho_min<rho_max)                                    error('pb'), end
      disp(num2str([norm_value,rho_min,rho_max])),
      tab_l=[tab_l; norm_value, rho_max,rho_min];
    end
    save(name_data,'tab_l','norm_value_l','F');
  else
    load(name_data);
  end
  rho_min_th=@(norm_x)sqrt(norm_x.^2+1);
  rho_1_th=@(norm_x)norm_x.*sqrt(1+3/4*norm_x.^2);
  rho_2_th=@(norm_x)norm_x.*sqrt(1+norm_x.^2);
  rho_max_th=@(norm_x)sqrt(1.5*norm_x.^2+1);
  figure(1);
  % plot(tab_l(:,1),tab_l(:,2),'b-','linewidth',3,tab_l(:,1),tab_l(:,2),'m-','linewidth',
    % tab_l(:,1),rho_min_th(tab_l(:,1)),'linewidth',3,tab_l(:,1),rho_max_th(tab_l(:,1)),'
  plot(tab_l(:,1),tab_l(:,2),'b-','linewidth',3,tab_l(:,1),tab_l(:,3),'m-','linewidth',3)
    %tab_l(:,1),rho_1_th(tab_l(:,1)),'linewidth',3,tab_l(:,1),rho_2_th(tab_l(:,1)),'linew
  axis([0 10 0 10])
  legend('\rho_{max}','\rho_{min}');
  set(gca, "linewidth", 3, "fontsize", 14)
  title(name_title),
  prin(1);
end


function y=fun1(x,norm_value)
%x is in F and y is the norm of aug vector x
  x=x(:)';
  if norm(x)>1e-10
    x=norm_value/norm(x)*x;
    if ~(abs(norm_value-norm(x))<1e-6)                     error('pb'), end
  end
  y=norm(constr1(x));
end

function xc=constr1(x)
  x=x(:).';
  xc=[x x(:,1).^2 x(:,1).*x(:,2) x(:,2).^2];
end
```

6.3 Kernel trick

Exercise. 48 *We consider a small dataset*

$$\mathbf{x}_1 = [1,\ 0]$$
$$\mathbf{x}_2 = [0,\ 1]$$
$$\mathbf{x}_3 = [1,\ 1]$$

We consider three new features x_1^2, x_1x_2 and x_2^2 and its corresponding mapping ω. We consider a first kernel K

$$K(\mathbf{x}, \mathbf{x}') = \omega(\mathbf{x}).\omega(\mathbf{x}')$$

1. *Express K as function of $[x_1, x_2]$ and $[x'_1, x'_2]$. Is it left-linear, right-linear?*
2. *Compute $\mathbf{K} = [K(\mathbf{x}_m, \mathbf{x}_n)]_{m,n}$*
3. *Show that the inverse of $\mathbf{K}$ is defined?*

Exercise. *The inverse of $\mathbf{K}$ is*

$$K^{-1} = \begin{bmatrix} 1.5 & 1 & -1 \\ 1 & 1.5 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

We define

$$K(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1), K(\mathbf{x}, \mathbf{x}_2), K(\mathbf{x}, \mathbf{x}_3)]\mathbf{K}^{-1} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}$$

4. *Compute $K(\mathbf{x}_1)$, $K(\mathbf{x}_2)$ and $K(\mathbf{x}_3)$.*
5. *Show that there exists $\mathbf{x}$ such that $\omega(\mathbf{x}) \notin \text{span}(\omega(\mathbf{x}_1), \omega(\mathbf{x}_2), \omega(\mathbf{x}_3))$. Explain how we could manage to avoid this problem?*
6. *Compute $K(\mathbf{x}_1 - \mathbf{x}_2)$.*

```
function fig_kernel1()
  name='fig_kernel1_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  if ~exist(name_data)
    N_l=3:10;
    norm_mean=zeros(1,length(N_l));
    for N_=1:length(N_l)
      N=N_l(N_);
      norm_mean_l(N_)=do_task(N);
    end
    save(name_data,'N_l','norm_mean_l');
  else
    load(name_data);
  end
  figure(1); semilogy(N_l,norm_mean_l,'linewidth',3)
  title(name_title),
  set(gca, "linewidth", 3, "fontsize", 16)
  prin(1);
```

```
end


function norm_mean=do_task(N);
  It=1e4;
  X=randn(N,2);
  augm=@(x)[x(1),x(2),x(1)^2,x(1)*x(2),x(2)^2];
  augm_=@(X)[X(:,1),X(:,2),X(:,1).^2,X(:,1).*X(:,2),X(:,2).^2];
  Xaugm=augm_(X);
  kernel=@(xa,xb)sum(augm(xa).*augm(xb));
  K=zeros(N);
  epsi=1e-5;
  for m=1:N
    for n=1:N
      K(m,n)=kernel(X(m,:),X(n,:));
    end
  end
  Ki=inv(K+epsi*eye(N));
  norm_mean=0;
  for it=1:It
    x=randn(1,2); x=x/sqrt(sum(x.^2));
    xb=Kfun(x,X,Ki,N)*Xaugm;
    norm_mean=norm_mean+norm(xb-augm(x));
  end
  norm_mean=norm_mean/It,
end

function xb=Kfun(x,X,Ki,N)
  augm=@(x)[x(1),x(2),x(1)^2,x(1)*x(2),x(2)^2];
  kernel=@(xa,xb)sum(augm(xa).*augm(xb));
  Kvect=zeros(1,N);
  for n=1:N
    Kvect(n)=kernel(x,X(n,:));
  end
  xb=Kvect*Ki;
end
```

Exercise. 49 *Write an algorithm to test the representation theorem on the kernel derived from* $\mathbf{x} \mapsto \omega(\mathbf{x})$.

6.4 Curse of dimensionality and feature extraction

```
function fig_hughes()
  name='fig_hughes_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  if ~exist(name_data)
    [dim_l,a_l,a2_l,a3_l]=do_task1();
    save(name_data,'dim_l','a_l','a2_l','a3_l');
  else
    load(name_data);
  end
  figure(1); plot(dim_l,a_l,'linewidth',3,dim_l,a2_l,'linewidth',3,dim_l,a3_l,'linewidth'
```

```
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([0 Inf 0 1])
  legend('learning from x_f','learning from average of x_f',...
    'learning from x_0 x_1 and some copies','location','SouthEast');
  title(['average accuracy ',name_title]),
  prin(1);
end


function [dim_l,a_l,a2_l,a3_l]=do_task1();
  dim_l=1:15; a_l=zeros(size(dim_l)); a2_l=a_l; a3_l=a_l;
  N=10; E=500;
  for dim_=1:length(dim_l)
    a=0; a2=0; a3=0;
    for exp_=1:E
      dim=dim_l(dim_);
      [Xl,Yl]=prepa1(N,dim);
      w=L2solver(Xl,Yl);
      [Xt,Yt]=prepa1(N,dim);
      Yh=predict(Xt,w);
      a=a+acc(Yh,Yt);
      Xl2=sum(Xl,2)/size(Xl,2);
      w2=L2solver(Xl2,Yl);
      Yh2=predict(sum(Xt,2)/size(Xt,2),w2);
      a2=a2+acc(Yh2,Yt);
      [X3,Y3]=prepa2(N,dim);
      w3=L2solver(X3,Y3);
      [Xt3,Yt3]=prepa2(N,dim);
      Yh3=predict(Xt3,w3);
      a3=a3+acc(Yh3,Yt3);
    end
    a_l(dim_)=a/E; a2_l(dim_)=a2/E; a3_l(dim_)=a3/E;
  end
end

function [X,Y]=prepa1(N,dim)
  mu1=0.5; mu0=-0.5; sigma=0.5;
  Y=rand(N,1)>0.5;
  ind1=find(Y==1); ind0=find(Y==0);
  X=zeros(N,dim);
  X(ind1,:)=sigma*randn(length(ind1),dim)+mu1;
  X(ind0,:)=sigma*randn(length(ind0),dim)+mu0;
end

function [X,Y]=prepa2(N,dim)
  mu1=0.5; mu0=-0.5; sigma=0.5;
  Y=rand(N,1)>0.5;
  ind1=find(Y==1); ind0=find(Y==0);
  N1=length(ind1); N0=length(ind0);
  X=zeros(N,dim);
  x1=sigma*randn(N1,1)+mu1;
  x0=sigma*randn(N0,1)+mu0;
  X(ind1,:)=(x1*ones(1,dim)).*(1-0.01*randn(N1,dim))+0.01*randn(N1,dim);
  X(ind0,:)=(x0*ones(1,dim)).*(1-0.01*randn(N0,dim))+0.01*randn(N0,dim);
```

```
end
```

```
function w=L2solver(X,Y)
  Xe=[X ones(size(X,1),1)];
  Ytilde=2*Y-1;
  w=(inv(Xe'*Xe)*(Xe'*Ytilde))';
end

function Yh=predict(X,w)
  Xe=[X ones(size(X,1),1)];
  Yh=zeros(size(X,1),1);
  for n=1:size(X,1)
    Yh(n)=(sum(w.*Xe(n,:))>=0);
  end
end

function A=acc(Y,Yh)
  A=mean(Y==Yh);
end
```

6.5 Principal Component Analysis

Exercise. 50 *We consider a tiny dataset with*

$$\mathbf{x}_1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$
$$\mathbf{x}_2 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

1. *Compute* $\mathbf{X}$ *and* $\mathbf{X}^T\mathbf{X}$

We assume that using a PCA-algorithm we found $\mathbf{P}$ *and* $\mathbf{D}$

$$\mathbf{P} = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } \mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{9} \end{bmatrix}$$

2. *Write the analysis and synthesis equations and check that we have a perfect reconstruction.*

Exercise.

3. *Considering that we keep only one component, write the approximation scheme.*
4. *Check the orthogonality property.*
5. *Compute* $\|\mathbf{x}\|^2$, $\|\mathbf{x} - PCA_{\mathcal{T}}(\mathbf{x})\|^2$
6. *Compute* $\mathscr{A}_{\mathcal{T}PCA}$
7. *Check the* $\mathbf{X}$*-signification of* $\mathscr{A}_{\mathcal{T}PCA}$

Exercise. 51 *We consider two independant Gaussian random variable* $\overset{r}{z}_1$ *and* $\overset{r}{z}_2$ *centered and normalised.*

$$\overset{r}{z}_1 \sim \mathcal{N}(0,1) \text{ and } \overset{r}{z}_2 \sim \mathcal{N}(0,1)$$

We define a random vector

$$\overset{r}{\mathbf{x}} = \left[\frac{2}{3}\overset{r}{z}_1 + \frac{1}{3}\overset{r}{z}_2, \quad \frac{1}{3}\overset{r}{z}_1 + \frac{2}{3}\overset{r}{z}_2\right]$$

1. *Compute the covariance matrix using* $\Sigma = \mathcal{E}\left[(\overset{r}{\mathbf{x}} - \boldsymbol{\mu})^T(\overset{r}{\mathbf{x}} - \boldsymbol{\mu})\right]$

Exercise. 52 *We consider a centered multivariate normal distribution*

$$\overset{r}{\mathbf{x}} \sim \mathcal{N}(0,\Sigma) \text{ and } \Sigma = \begin{bmatrix} \frac{5}{9} & \frac{4}{9} \\ \frac{4}{9} & \frac{5}{9} \end{bmatrix}$$

We want to find the locus of equal density probability of $\mathbf{x}$.

1. *Show that this locus fullfills*

$$J = \frac{1}{2}\mathbf{x}\Sigma^{-1}\mathbf{x}^T$$

   with a probability density of $\frac{9}{2\pi}e^{-J}$

2. *Check that*

$$\Sigma^{-1} = \begin{bmatrix} 5 & -4 \\ -4 & 5 \end{bmatrix}$$

3. *Defining* $\mathbf{x}$ *with coordinates:* $\mathbf{x} = [x_1 \ x_2]$*, show that they fullfill*

$$2J = 5x_1^2 - 8x_1x_2 + 5x_2^2$$

Exercise. 4. *We now use polar coordinates* $x_1 = r\cos(\theta)$ *and* $x_2 = r\sin(\theta)$*. Show that*

$$r(\theta) = \frac{\sqrt{2J}}{\sqrt{5 - 4\sin(2\theta)}}$$

and hence that a parametric description of the contour is

$$\begin{cases} x(\theta) = r(\theta)\cos(\theta) \\ y(\theta) = r(\theta)\sin(\theta) \end{cases}$$

5. *Describe the contour and find its closest and farthest points.*

6. *Find a unit vector along the* **farthest** *point's direction. We will see that this is the first eigenvector and hence the first column of the* P*-matrix.*

```
function fig_ex25()
  name='fig_ex25_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  theta_l=(0:1e-3:2*pi)';
```

```
  J=1;
  r=@(theta)sqrt(2)*sqrt(J)./sqrt(5-4*sin(2*theta));
  x_l=[r(theta_l).*cos(theta_l) r(theta_l).*sin(theta_l)];
  figure(1); plot(x_l(:,1), x_l(:,2),'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  %axis([0 7 0 Inf])
  %legend('Probability distribution of variance',['Mean=',num2str(M)]);
  title(name_title),
  prin(1);
  X=draw_points(1000);
  figure(2); plot(X(:,1),X(:,2),'.','linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('x_1'); ylabel('x_2');
  title(name_title),
  prin(2);
end

function X=draw_points(N)
  z1=randn(N,1); z2=randn(N,1);
  x1=2/3*z1+1/3*z2; x2=1/3*z1+2/3*z2;
  X=[x1 x2];
end

function fig_explain_variance()
  name='fig_explain_variance_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  if ~exist(name_data)
    V_exp_l=do_task();
    save(name_data,'V_exp_l');
  else
    load(name_data);
  end
  M=mean(V_exp_l),
  [n,v_exp]=hist(V_exp_l,1000);
  n_h=n/sum(n);
  figure(2); plot(v_exp,n_h,'linewidth',3,[M M],[0 max(n_h)],'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([0 3.2 0 Inf])
  legend('Probability distribution of variance',['Mean=',num2str(M)]);
  title(name_title),
  prin(1);

end

function V_exp_l=do_task()
  It=1e7; N=5;
  V_exp_l=zeros(1,It);
  for cpt=1:It
    Sigma_2=mk_prob(N);
    %if ~(abs(1-trace(Sigma_2'*Sigma_2))<1e-6)                    error('pb'), end
    x=randn(1,N)*Sigma_2;
    V_exp_l(cpt)=x*x';
  end
```

```
end



function fig_explain_variance_old()
  It=1e3; N=20;
  V_th_l=zeros(1,It);
  V_exp_l=zeros(1,It);
  V_exp2_l=zeros(1,It);
  V_exp3_l=zeros(1,It);
  for cpt=1:It

    Sigma_2=mk_prob(N);
    %Sigma_2=eye(N)/sqrt(N);
    if ~(abs(1-trace(Sigma_2'*Sigma_2))<1e-6)                    error('pb'), end
    x=randn(1,N)*Sigma_2;
    %x=x-mean(x);
    V_th_l(cpt)=trace(Sigma_2'*Sigma_2);
    V_exp_l(cpt)=x*x';
    z=0;
    for l=1:10
      x=randn(1,N)*Sigma_2;
      z=z+x*x'/10;
    end
    X=randn(10,N)*Sigma_2;
    %X=X-sum(X,2)/N;
    V_exp3_l(cpt)=trace(X'*X)/10;
    V_exp2_l(cpt)=z;
  end
  [~,ind]=sort(V_th_l);
  %figure(1); plot(V_th_l(ind),V_exp_l(ind),'.');
  mean(V_exp_l),
  [n,v_exp]=hist(V_exp_l,[0.2:0.2:3]);
  figure(2); plot(v_exp,n/sum(n));
  [n2,v_exp2]=hist(V_exp2_l,[0.2:0.2:3]);
  figure(3); plot(v_exp2,n2/sum(n2));
  [n3,v_exp3]=hist(V_exp3_l,[0.2:0.2:3]);
  figure(4); plot(v_exp3,n3/sum(n3));

end

function Sigma=mk_prob(N)
%the true Sigma is Sigma'*Sigma
  Sigma=rand(N);
  Sigma=Sigma/sqrt(trace(Sigma'*Sigma));
end

function Sigma=mk_prob2(N)
%the true Sigma is Sigma'*Sigma
  Sigma=diag(rand(1,N));
  Sigma=Sigma/sqrt(trace(Sigma'*Sigma));
end
```

Exercise. 53 *We consider a covariance matrix*

$$\Sigma = \frac{1}{9}\begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

We are trying to solve the eigenvalue problem.

1. *Write the second order polynomial yielding the eigenvalues and find them.*
2. *Find the eigenvectors and write the equation.*

Exercise. 54 *We consider the same centered multivariate normal distribution as defined in exercise 52.*

$$\overset{r}{\mathbf{x}} \sim \mathcal{N}(0, \Sigma) \text{ and } \Sigma = \begin{bmatrix} \frac{5}{9} & \frac{4}{9} \\ \frac{4}{9} & \frac{5}{9} \end{bmatrix}$$

We assume that using a PCA-algorithm we found P and D

$$\mathbf{P} = \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } \mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{9} \end{bmatrix}$$

1. *Write the equations of the whitening process transforming $\overset{r}{\mathbf{x}}$ into $\overset{r}{\mathbf{z}'}$.*

We now assume as in exercise 51 that actually $\overset{r}{\mathbf{x}}$ comes from two centered normalized Gaussian random variable $\overset{r}{z}_1$ and $\overset{r}{z}_2$.

$$\overset{r}{x}_1 = \frac{2}{3}\overset{r}{z}_1 + \frac{1}{3}\overset{r}{z}_2 \text{ and } \overset{r}{x}_2 = \frac{1}{3}\overset{r}{z}_1 + \frac{2}{3}\overset{r}{z}_2$$

2. *Check that $\overset{r}{\mathbf{z}}$ is indeed white.*

Exercise. 55 *We consider the tiny dataset of exercise 50with*

$$\mathbf{x}_1 = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$
$$\mathbf{x}_2 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

1. *Compute the correlation matrix.*

6.6 Supervised feature extraction

6.7 Regularization

Exercise. 56

1. *What is doing this code?*

```
function fig_cond()
  N=10; F=10; cd=zeros(3); X=randn(N,F);
  for m=1:4
    Xn=X; X=smooth(X')';
    for n=1:3
      Xn=smooth(Xn); cd(m,n)=cond(Xn);
    end
  end
  disp(num2str(round(cd))),
end
function X2=smooth(X1)
  N=size(X1,2); X2=[X1(:,1) (X1(:,1:N-1)+X1(:,2:N))/2];
end
```

Exercise. 57 *Solve analytically the new optimization problem with the regularized L_2-loss function.*

```
function fig_regularization3()
%less bizarre random dataset
%called problem B
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  close all;
  delete(name_data);
  if ~exist(name_data)
    [lambda_l,acc_l]=do_task();
    save(name_data,'acc_l','lambda_l');
  else
    load(name_data);
  end
  figure(1); semilogx(lambda_l,acc_l,'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  %axis([0 Inf 0 1])
  xlabel('\lambda'); ylabel('average accuracy');
  title(['Accuracy as a function of lambda ',name_title]),
  prin(1);
end


function [lambda_l,acc_l]=do_task()
  lambda_l=10.^(-4:0.1:3);
  acc_l=zeros(size(lambda_l));
  dim=10;
  E=300;
  for exp=1:E
```

```
    mu1=2*randn(1,dim); mu0=2*randn(1,dim);
    sigma=rand(dim); sigma=(sigma+sigma')/2;
    [Xl,Yl]=prepa1(10,mu0,mu1,sigma);
    mu=sum(Xl,1)/size(Xl,1);
    [Xq,Yq]=prepa1(50,mu0,mu1,sigma);
    for lambda_=1:length(lambda_l)
      lambda=lambda_l(lambda_);
      w=L2solver(Xl,Yl,lambda);
      Yhq=predict(Xq,w);
      acc_l(lambda_)=acc_l(lambda_)+acc(Yhq,Yq);
    end
  end
  acc_l=acc_l/E;
end


function [X,Y]=prepa1(N,mu0,mu1,sigma)
  dim=size(mu0,2);
  Y=rand(N,1)>0.5;
  ind1=find(Y==1); N1=length(ind1);  ind0=find(Y==0); N0=length(ind0);
  X=zeros(N,dim);
  X(ind1,:)=randn(N1,dim)*sigma+ones(N1,1)*mu1;
  X(ind0,:)=randn(N0,dim)*sigma+ones(N0,1)*mu0;
end


function w=L2solver(X,Y,lambda)
  Xe=[X ones(size(X,1),1)];
  Sigma=Xe'*Xe+lambda*eye(size(X,2)+1);
  Ytilde=2*Y-1;
  w=(inv(Sigma)*(Xe'*Ytilde))';
end

function Yh=predict(X,w)
  Xe=[X ones(size(X,1),1)];
  Yh=zeros(size(X,1),1);
  for n=1:size(X,1)
    Yh(n)=(sum(w.*Xe(n,:))>=0);
  end
end

function A=acc(Y,Yh)
  A=mean(Y==Yh);
end
```

Exercise. 58 *We consider a regression problem, that is we want to predict* **values** *instead of labels. The values are represented by* Y*. For the sake of simplicity, we consider here only one feature, so the data matrix* $\mathbf{X}$ *is here a column vector* X*.* **a** *is a scalar,* a*.*

$$Y = aX + \eta$$

a *and* η *are here regarded as a random variable and vector.*

$$\overset{r}{a} \sim \mathcal{N}(0, \sigma_a) \text{ and } \overset{r}{\eta} \sim \mathcal{N}(0, \sigma_\eta \mathbf{I}_N)$$

1. *Write the likelihood of* Y *given* X *and* a*.*
2. *Write the posterior probability* a *given* X *and* Y *as a function of the likelihood and a prior.*

Exercise. 3. *Show that* $\widehat{a}$ *maximizing the posterior probability is defined as*

$$\widehat{a} = \arg\min_a (Y - aX)^T (Y - aX) + \frac{\sigma_\eta^2}{\sigma_a^2} a^2$$

```
function fig_regularization4()
%prior of problem B
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  close all;
  %delete(name_data);
  if ~exist(name_data)
    a_l=do_task();
    save(name_data,'a_l');
  else
    load(name_data);
  end
  f_gauss=@(x,sig)1/sqrt(2*pi)/sig*exp(-0.5*x.^2/sig^2);
  f_laplace=@(x,b)1/2/b*exp(-abs(x)/b);
  [x_norm,fx_norm,sigma_norm,m_norm]=dist_est(sqrt(sum(a_l.^2,2)));
  sigma_norm,
  figure(1); plot(x_norm,fx_norm,'linewidth',3,...
  x_norm,f_gauss(x_norm-m_norm,sigma_norm),'linewidth',3,...
  x_norm,f_laplace(x_norm-m_norm,sigma_norm/2),'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16);
  legend('estimated','Gaussian','Laplace');
  xlabel('Norm of w'); ylabel('Probability distribution');
  title(['',name_title]),
  axis([0 2*sigma_norm+m_norm 0 Inf]),
  prin(1);

  data=a_l(:,1);
  [x,fx,sigma,m]=dist_est(data);
  sigma,
  figure(2); plot(x,fx,'linewidth',3,...
  x,f_gauss(x-m,sigma),'linewidth',3,...
  x,f_laplace(x,sigma/2),'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
```

```
  legend('estimated','Gaussian','Laplace');
  xlabel('First component of w'); ylabel('Probability distribution');
  title(['',name_title]),
  axis([-2*sigma+m 2*sigma+m 0 Inf]),
  prin(2);



end



function [x,fx,sigma,m]=dist_est(data);
  [n,x]=hist(data,1000);
  fx=n/sum(n)/(x(2)-x(1));
  sigma=std(data);
  m=mean(data);
end


function a_l=do_task()
  dim=10;
  E=1e4;
  a_l=[];
  for exp=1:E
    mu1=2*randn(1,dim); mu0=2*randn(1,dim);
    sigma=rand(dim); sigma=(sigma+sigma')/2;
    [Xl,Yl]=prepa1(100,mu0,mu1,sigma);
    w=estimate_w(Xl,Yl);
    a_l=[a_l; w];
  end
end


function w=estimate_w(X,Y)
  N=size(X,1);
  Xe=[X ones(N,1)];
  w=(inv(Xe'*Xe)*(Xe'*Y))';
end

function [X,Y]=prepa1(N,mu0,mu1,sigma)
  dim=size(mu0,2);
  Y=rand(N,1)>0.5;
  ind1=find(Y==1); N1=length(ind1);  ind0=find(Y==0); N0=length(ind0);
  X=zeros(N,dim);
  X(ind1,:)=randn(N1,dim)*sigma+ones(N1,1)*mu1;
  X(ind0,:)=randn(N0,dim)*sigma+ones(N0,1)*mu0;
end



function w=L2solver(X,Y,lambda)
  Xe=[X ones(size(X,1),1)];
  Sigma=Xe'*Xe+lambda*eye(size(X,2)+1);
  Ytilde=2*Y-1;
```

```
  w=(inv(Sigma)*(Xe'*Ytilde))';
end

function Yh=predict(X,w)
  Xe=[X ones(size(X,1),1)];
  Yh=zeros(size(X,1),1);
  for n=1:size(X,1)
    Yh(n)=(sum(w.*Xe(n,:))>=0);
  end
end

function A=acc(Y,Yh)
  A=mean(Y==Yh);
end
function fig_regularization6()
%likelihood of problem B
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  close all;
  %delete(name_data);
  if ~exist(name_data)
    load ./prg/fig_regularization5_data.mat
    [noise1,fnoise1,sigma_noise1,m_noise1]=dist_est(noise_l(:,1));
    norm_noise=sqrt(sum(noise_l.^2,2)/size(noise_l,2));
    [noise_norm,fnoise_norm,sigma_noise_norm,m_norm]=dist_est(norm_noise);
    save(name_data,'noise1','fnoise1','sigma_noise1','m_noise1','noise_norm','fnoise_norm
  else
    load(name_data);
  end
  sigma_norm, sigma_noise1,
    f_gauss=@(x,sig)1/sqrt(2*pi)/sig*exp(-0.5*x.^2/sig^2);
  f_laplace=@(x,b)1/2/b*exp(-abs(x)/b);
  figure(1); plot(noise1,fnoise1,'linewidth',3,noise1,f_gauss(noise1,sigma_noise1),'linew
    noise1,f_laplace(noise1,sigma_noise1/2),'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([-2*sigma_noise1 2*sigma_noise1 0 Inf]),
  xlabel('first component of noise'); ylabel('probability density');
  prin(1);

  figure(2); plot(noise_norm,fnoise_norm,'linewidth',3,noise_norm,f_gauss(noise_norm-m_no
    noise_norm,f_laplace(noise_norm-m_norm,sigma_norm/2),'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([-2*sigma_norm+m_norm 2*sigma_norm+m_norm 0 Inf]),
  xlabel('norm of noise'); ylabel('probability density');
  prin(2);
end




function [x,fx,sigma,m]=dist_est(data);
  [n,x]=hist(data,1000);
```

```
  fx=n/sum(n)/(x(2)-x(1));
  sigma=std(data);
  m=mean(data);
end
```

6.8 Feature selection

Exercise. 59 *We consider again exercise 3.1 and the proposed solution in exercise 27 where*

$$\overset{\triangle}{\mathbf{X}} = [\mathbf{X}1], \quad \mathbf{w} = [-\mathbf{a}\ b] \text{ and } \mathbf{w}^T = \left(\overset{\triangle}{\mathbf{X}}^T \overset{\triangle}{\mathbf{X}}\right)^{-1} \overset{\triangle}{\mathbf{X}}^T \widetilde{Y}$$

with

$$f_{\mathbf{a},b}(\mathbf{x}) = \mathbf{1}(\mathbf{a}.\mathbf{x} \leq b)$$

1. *Let us suppose that the first component of all samples in $\mathcal{S}_2$ is constant, why would this be a problem in these equations. Suggest an experiment studying this question.*
2. *What should we think of this situation?*
3. *What could we do?*

```
function fig_feature_selection2()
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  close all;
  %delete(name_data);
  if ~exist(name_data)
    [lambda_l,w_l,acc_l]=do_task();
    save(name_data,'lambda_l','w_l','acc_l');
  else
    load(name_data);
  end
  norm_L1_w_l=sum(abs(w_l),2);
  figure(1);
  semilogx(lambda_l,norm_L1_w_l,'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('\lambda'); ylabel('sum of abs of w');
  prin(1);

  figure(2); hold on;
  set(gca, "linewidth", 3, "fontsize", 16)
  line=['legend('];
  for w_=1:size(w_l,2)
    semilogx(lambda_l,w_l(:,w_),'linewidth',3);
    line=[line,'''w_',num2str(w_),''','];
  end
  line=[line(1:end-1),');'];
  xlabel('\lambda'); ylabel('Components of w');
  hold off;
  prin(2);
```

```
  figure(3);
  semilogx(lambda_l,acc_l,'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('\lambda'); ylabel('accuracy');
  prin(3);


end

function w=estimate_ridge_w(X,Y,lambda)
  N=size(X,1); F=size(X,2);
  Xe=[X ones(N,1)];
  w=(inv(Xe'*Xe+lambda*eye(F+1))*(Xe'*Y))';
end


function w=estimate_lasso_w(X,Y,lambda,w_init)
  N=size(X,1); F=size(X,2);
  Xe=[X ones(N,1)];
  horiz=@(w)w(:)';
  J=@(w)horiz(w)*Xe'*Xe*horiz(w)'-2*horiz(w)*Xe'*(2*Y-1)+lambda*sum(abs(w));
  if ~isnan(w_init)
    w=simulated_annealing(J,F+1,'init_value',w_init,'silent');
  else
    w=simulated_annealing(J,F+1,'silent');
  end
  disp(num2str([lambda,w(:)'])),
end

function Yh=predict(X,w)
  w=w(:)';
  Xe=[X ones(size(X,1),1)];
  Yh=zeros(size(X,1),1);
  for n=1:size(X,1)
    Yh(n)=(sum(w.*Xe(n,:))>=0);
  end
end


function [lambda_l,w_l,acc_l]=do_task()
  Nf=5; N=40;
  [mu0,mu1,sigma,F_s_true]=prb_chosen(Nf);
  [Xl,Yl]=prepa1(N,mu0,mu1,sigma);
  [Xq,Yq]=prepa1(N,mu0,mu1,sigma);
  lambda_l=10.^(-4:0.2:5);
  w_l=zeros(length(lambda_l),1+length(mu0));
  w=NaN;
  for lambda_=1:length(lambda_l)
    lambda=lambda_l(lambda_);
    w=estimate_lasso_w(Xl,Yl,lambda,w);
    w_l(lambda_,:)=w;
    Yh=predict(Xq,w);
    acc=mean(Yh==Yq);
    acc_l(lambda_)=acc;
```

```
  end
end

function [X,Y]=prepa1(N,mu0,mu1,sigma)
  dim=size(mu0,2);
  Y=rand(N,1)>0.5;
  ind1=find(Y==1); N1=length(ind1);  ind0=find(Y==0); N0=length(ind0);
  X=zeros(N,dim);
  X(ind1,:)=randn(N1,dim)*sigma+ones(N1,1)*mu1;
  X(ind0,:)=randn(N0,dim)*sigma+ones(N0,1)*mu0;
end


function [mu0,mu1,sigma,F_s_true]=prb_chosen(Nf)
  mu0=zeros(1,10);
  mu1=1:-0.1:0.1;
  sigma=eye(10);
  F_s_true=1:Nf;
end
```

7 Spatial context

7.1 Spatial context

7.2 Texture descriptors

```
function fig_texture1()
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  name_img=@(num)['./images/',name,num2str(num),'.png'];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  imwr=@(img,num)imwrite(img,name_img(num));
  name_data=['./prg/',name,'data.mat'];
  close all;
  [img,true_img]=chess_board();
  imwr(img,1);

  imwr(true_img,2);
  [Xl,Yl]=read_img(img,true_img);
  szImg=size(img);
  F_s=[1]; fig_n=3; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);
  F_s=[1 2]; fig_n=4; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);
  F_s=[1 3]; fig_n=5; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);
  F_s=[1 4]; fig_n=6; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);
  F_s=[1 5]; fig_n=7; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);
  F_s=[1 6]; fig_n=8; Yh=kmeans(Xl(:,F_s)); imwr(Y2img(Yh,size(img)),fig_n);


end

function img=Y2img(Y,szImg)
  img=zeros(szImg);
  M=szImg(1); N=szImg(2);
  for m=3:M-2
```

```
    for n=3:N-2
      k=(m-3)*(N-4)+n-3+1;
      img(m,n)=Y(k);
    end
  end
end

% function [img,true_img,do_task()
  % [img,true_img]=chess_board();
  % [Xl,Yl]=read_img(img,true_img);
  % Yh_a=kmeans(Xl(:,1));
% end

function [img,true_img]=chess_board()
  img_small=[ones(32) zeros(32); zeros(32) ones(32)];
  img=repmat(img_small,[4 4]);
  true_img=img;
  img=img+0.2*randn(size(img));
end

function [Xl,Yl]=read_img(img,true_img)
  M=size(img,1); N=size(img,2);
  F=6;
  Xl=zeros((M-4)*(N-4),F);
  Yl=zeros((M-4)*(N-4),1);
  for m=3:M-2
    for n=3:N-2
      k=(m-3)*(N-4)+n-3+1;
      Xl(k,1)=img(m,n);
      Yl(k)=true_img(m,n);
      wind=img(m-2:m+2,n-2:n+2);
      for f=1:F
        Xl(k,f)=read_feature(wind,f);
      end
    end
  end
  if ~(Xl(end,1)==img(M-2,N-2))                                    error('pb'), end

end

function [x,msg]=read_feature(img,f)
  std2_=@(mat)std(mat(:));
  mean2_=@(mat)mean(mat(:));
  if 1==f
    msg='pixel value';
    x=img(3,3);
  elseif 2==f
    msg='horizotonatfilter';
    x=sum(img(3,1:5))/5;
  elseif 3==f
    msg='variance';
    x=std(img(:))^2;
  elseif 4==f
    msg='diversity';
```

```
    [n,val]=hist(img(:),5);
    x=sum((n/sum(n)).^2);
  elseif 5==f
    msg='correlation';
    rho=mean2_((img(:,1:4).*img(:,2:5))/std2_(img(:,1:4))/std2_(img(:,2:5)));
    x=rho;
  elseif 6==f
    msg='mean';
    x=mean2_(img);
  else                                                    error('pb'), end
  if ~all([1 1]==size(x))                                 error('pb'), end
end

function Y=kmeans(X)
  while(1)
    ind_l=randperm(size(X,1),2);
    x0=X(ind_l(1),:); x1=X(ind_l(2),:);
    Y_l=[];
    n=0;
    while(1)
      n=n+1;
      Y=(dist2(X,x0)>  dist2(X,x1));
      if all(1==Y)||all(0==Y) break; end
      ind0=find(Y==0); ind1=find(Y==1);
      x0=sum(X(ind0,:),1)/length(ind0);
      x1=sum(X(ind1,:),1)/length(ind1);
      Y_l=[Y_l Y];
      if 1==n continue, end
      if (all(Y_l(:,end)==Y_l(:,end-1))) return; end
    end
  end
  Y=Y_l(:,end);
end

function D=dist2(X,xa)
  D=sum((X-ones(size(X,1),1)*xa).^2,2);
end
```

7.3 Noise estimation

```
function fig_noise1()
  name=[mfilename(),'_'];
  name_title=name; name_title(name_title=='_')=' ';
  name_img=@(num)['./images/',name,num2str(num),'.png'];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  imwr=@(img,num)imwrite(img,name_img(num));
  name_data=['./prg/',name,'data.mat'];
  close all;
  [img,img_noise]=do_task();
  imwr(img,1);
  imwr(img_noise,2);

end
```

```
function [img,img_noise]=do_task()
  [img,true_img]=chess_board2();
  [Xl,Yl]=read_img(img,true_img);
  X_noise=sqrt(Xl(:,3));
  img_noise=Y2img(X_noise,size(img));
end



function img=Y2img(Y,szImg)
  img=zeros(szImg);
  M=szImg(1); N=szImg(2);
  for m=3:M-2
    for n=3:N-2
      k=(m-3)*(N-4)+n-3+1;
      img(m,n)=Y(k);
    end
  end
end

function [img,true_img]=chess_board2()
  img_small=[ones(16) zeros(16); zeros(16) ones(16)];
  img=repmat(img_small,[4 4]);
  true_img=img;
  img=img+(0.2+img).*randn(size(img));
end



function [Xl,Yl]=read_img(img,true_img)
  M=size(img,1); N=size(img,2);
  F=6;
  Xl=zeros((M-4)*(N-4),F);
  Yl=zeros((M-4)*(N-4),1);
  for m=3:M-2
    for n=3:N-2
      k=(m-3)*(N-4)+n-3+1;
      Xl(k,1)=img(m,n);
      Yl(k)=true_img(m,n);
      wind=img(m-2:m+2,n-2:n+2);
      for f=1:F
        Xl(k,f)=read_feature(wind,f);
      end
    end
  end
  if ~(Xl(end,1)==img(M-2,N-2))                                  error('pb'), end

end

function [x,msg]=read_feature(img,f)
  std2_=@(mat)std(mat(:));
  mean2_=@(mat)mean(mat(:));
  if 1==f
    msg='pixel value';
    x=img(3,3);
```

```
  elseif 2==f
    msg='horizotonatfilter';
    x=sum(img(3,1:5))/5;
  elseif 3==f
    msg='variance';
    x=std(img(:))^2;
  elseif 4==f
    msg='diversity';
    [n,val]=hist(img(:),5);
    x=sum((n/sum(n)).^2);
  elseif 5==f
    msg='correlation';
    rho=mean2_((img(:,1:4).*img(:,2:5))/std2_(img(:,1:4))/std2_(img(:,2:5)));
    x=rho;
  elseif 6==f
    msg='mean';
    x=mean2_(img);
  else                                                    error('pb'), end
  if ~all([1 1]==size(x))                                 error('pb'), end
end
```

7.4 Spatial prior

8 Supplementary material regarding matrices

Exercise. 61 *Considering a binary dataset* $(\mathbf{X}, Y)$ *composed of* $N = 3$ *samples belonging to a feature space of size* F*, and considering a matrix* T *of size* 3×3 *defined as*

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

show that $(T\mathbf{X}, TY)$ *is the same dataset.*

8.1 Proving that kmeans is related to an optimization problem

Exercise. 62 *We consider a dataset* $(\mathbf{X}, Y)$ *and denote* $N_0, N_1, \boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ *the number of* 0*-labeled samples,* 1*-labeled samples, the geometric center of the* 0*-labeled samples and that of the* 1*-labeled samples.*

1. *Prove that*

$$N_0\boldsymbol{\mu}_0 + N_1\boldsymbol{\mu}_1 = N\boldsymbol{\mu} = \sum_{n=1}^{N} \mathbf{x}_n$$

   where $\boldsymbol{\mu}$ *is the geometric center of the samples in the feature space.*

2. *Let* $Y' = Y$ *except for* $n = n_0$ *where* $y_{n_0} = 0$ *and* $y'_{n_0} = 1$*. Show that*

$$N_0(Y') = N_0(Y) - 1, \quad N_1(Y') = N_1(Y) + 1,$$

Exercise. *3. Let $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ be the means of the 0 and 1-labeled samples before the modification. Let $\boldsymbol{\mu}'_0, \boldsymbol{\mu}'_1$ be the corresponding means after the modification. Show that*

$$\boldsymbol{\mu}'_0 - \mathbf{x} = \frac{N_0}{N_0 - 1}(\boldsymbol{\mu}_0 - \mathbf{x})$$

4. We denote by J and J' the values of loss function for $(\mathbf{X}, Y)$ and $(\mathbf{X}', Y')$. Using the adding-a-sample identity, show that

$$J' - J = \frac{N_1}{N_1 + 1}\|\boldsymbol{\mu}_1 - \mathbf{x}\|^2 - \frac{N_0}{N_0 - 1}\|\boldsymbol{\mu}_0 - \mathbf{x}\|^2$$

5. Show that $J' \leq J$, when Y' is modified according to kmeans, still assuming that here only **one** *component changes.*