Lecture Notes

Supervised and Unsupervised Classification of Hyperspectral Images

An Introduction to the use of spatial information and of sparsity

Gabriel Dauphin

May 8, 2023

Contents

1 Classification of hyperspectral images

Data Set

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

We are using the following notations.

- lower case indicates scalars.
- Bold lower case indicates row vectors.
- Capital letters indicate column vectors.
- Bold capital letters indicate matrices.
- Script capital letters indicate sets.
- Calligraphic letters indicate functions to be minimized.
- N number of samples. `N=size(X,1)=length(Y)`.
- n sample counter: $1 \leq n \leq N$.
- F number of features. `F=size(X,2)`.
- f feature counter: $1 \leq f \leq F$.
- $\mathbf{x}$ is a sample written as a row vector.
- $\mathbf{X}$ is a a matrix, each line is a sample and each column is the set of all values found in the data set for a specific feature.
- Y is the set of labels written as a column binary vector.
- $\mathcal{S}(\mathbf{X})$ is the set containing the N rows of $\mathbf{X}$.
- $\mathcal{S}(\mathbf{X}, Y)$ is the set of pairs $(\mathbf{x}, y)$ containing a row of $\mathbf{X}$ and the corresponding value in Y.

Binary Classification Problem

$$y_n \in \{0, 1\} \tag{2}$$

Exercise 1. (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. (2) *Draw and code with Octave the scatter plot of the following dataset*

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

Exercise 3. (3) *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.

Exercise 4. (17) *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.*

It is generally thought that features should be **normalized**, meaning that for each feature f, the range of values should be similar. **Normalization** means that by multiplying by a certain value, feature values should fulfill

$$\forall f, \quad \frac{1}{N} \sum_{n=1}^{N} x_{nf}^2 = 1 \tag{3}$$

Exercise 5. (18) *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 $x'_{nf} = \alpha_f x_{nf}$

2 Statistical Inference as an Optimization Problem

2.1 Supervised Classification

Predictor

$$\widehat{y} = f(\mathbf{x}) \tag{4}$$

Iverson Bracket

$$\delta(\Pi) = \begin{cases} 1 \text{ if } \Pi \text{ is true} \\ 0 \text{ if not} \end{cases} \tag{5}$$

Accuracy (also called overall accuracy)

$$\mathcal{A}(Y, \widehat{Y}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\widehat{y}_n = y_n) \tag{6}$$

With the *opposite sign*, it is an example of **loss function** denoted as $\mathscr{L}(Y,\widehat{Y})$.

This accuracy is actually a function of the data set $\mathcal{S}$ and of a predictor $f_\Theta(\mathbf{x})$, a function of features depending on parameters denoted Θ.

$$\mathscr{A}(\mathcal{S},f_\Theta)=\mathcal{A}\left(Y,\begin{bmatrix}f_\Theta(\mathbf{x}_1)\\f_\Theta(\mathbf{x}_2)\\\vdots\end{bmatrix}\right)=\sum_{i=1}^{n}\delta(f_\Theta(\mathbf{x}_n)=y_n) \tag{7}$$

In binary classification, classes are generally labeled as -1 and 1 instead of 0 and 1. In terms of notations, the new labels are here denoted as $\widetilde{y}$

$$\widetilde{y}=2y-1 \tag{8}$$

I see the following reasons:

- To express that the class of y is swapped:

$$-\widetilde{y}=\widetilde{1-y}$$

- To express that y_1 swaps the class of y_2:

$$\widetilde{y}_1\widetilde{y}_2 \quad = \quad \widetilde{y_1y_2+(1-y_1)(1-y_2)} \quad = \quad \widetilde{(2y_1-1)y_2+1-y_1}$$

And we will see later another reason when defining loss functions.

Exercise 6. (4) *We are considering the following predictor which is an example of decision stump.*

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

with a and b as parameters.

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

$$f_{x,y}(z)=f_{x,z}(y)\delta(y=z)$$
$$+(1-f_{x,z}(y))\delta(y\neq z)$$

A **decision stump** makes a decision based on the value of a feature.

$$f_{\theta_F,\theta_x,\theta_y}(\mathbf{x})=(2\theta_y-1)\delta(x_{\theta_F}\leq\theta_x)+1-\theta_y \tag{9}$$

with $\theta_y\in\{0,1\}$, $\theta_F\in\{1\ldots F\}$ and $\theta_x\in\mathbb{R}$

We get an **optimization problem** here defined as

$$\Theta=\underset{\Theta}{\operatorname{argmin}}\,\mathscr{L}\left(Y,\begin{bmatrix}f_\Theta(\mathbf{x}_1)\\f_\Theta(\mathbf{x}_2)\\\vdots\end{bmatrix}\right) \tag{10}$$

Exercise 7. (6) *We are considering the predictor $f_{a,b}(x)$ defined as*

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

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

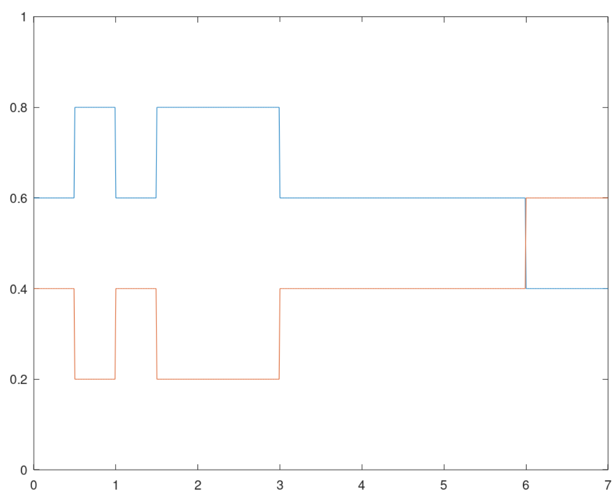


Figure 1: Accuracy obtained when thresholding the feature as a function of b. Exercise 6

The **feature space** is the set comprising all possible values of $\mathbf{x}$. We define on it a scalar product

$$\mathbf{x}.\mathbf{x}' = \sum_{f=1}^{F} x_f x'_f \tag{11}$$

A second kind of predictor is called **linear predictor** and defined as

$$f_{\mathbf{a},b}(\mathbf{x}) = \delta(\mathbf{a}.\mathbf{x} \leq b) \tag{12}$$

Exercise 8. (5) *We consider a predictor f defined as*

$$f(\mathbf{x}) = \delta(2x_1 + x_2 \leq 2) \tag{13}$$

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

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

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

We consider here a function J depending on several parameters listed in a column vector $\Theta = [\theta_1 \dots \theta_P]^T$.
A **global minimum** is an input Θ^* fulfilling

$$\forall \Theta, \quad \mathcal{J}(\Theta^*) \leq \mathcal{J}(\Theta) \tag{14}$$

A **local minimum** is an input Θ_0 fulfilling

$$\text{For all } \Theta \text{ close to } \Theta_0, \quad \mathcal{J}(\Theta_0) \leq \mathcal{J}(\Theta) \tag{15}$$

For a sufficiently smooth function, **inputs canceling the derivative** of a function are actually local minima or maxima of that function. A function that is not upper-bounded cannot have a unique local maxima. When there is a unique local minima, then it is a global minima.

A general idea to solve the optimization problem consists in finding $\mathbf{x}$ canceling the derivatives of the accuracy. However this accuracy does not have the expected regularities, so it is common practice to approximate this accuracy (or rather the lack of accuracy) by a more regular **Loss Function**.

An L_2**-loss function** is defined here as

$$\mathcal{L}(\mathcal{S}, f) = \frac{1}{2}\sum_{n=1}^{N}(f(\mathbf{x}_n) - y_n)^2 \tag{16}$$

We choose to use the following loss function derived from a real-valued predictors $f^v_{\mathbf{a},b} = b - \mathbf{a.x}$

$$\mathcal{L}(\mathcal{S}, f^v) = \frac{1}{2}\sum_{n=1}^{N}(f^v(\mathbf{x}_n) - \widetilde{y}_n)^2$$

Exercise 9. *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*

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

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

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

$$\left\{ \begin{array}{l} 2a_1 + 0.5a_2 - b = -1 \\ a_1 + 2a_2 - b = 1 \\ b = 1 \end{array} \right.$$

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

Simulated annealing is a generic old technique to find a solution minimizing a given cost function. We propose here to consider a simplified implementation.

Algorithm 1 Simplified implementation of simulated annealing

Require: $\mathscr{L}$
Ensure: Θ

Select randomly Θ and set $L := +\infty$.
for k=1:10000 **do**
  Select randomly r, a real in $[0, 6]$ and set $\sigma := 10^{-r}$.
  Select randomly $\Delta\Theta$ along a centered Gaussian distribution with σ as standard deviation.
  if $\mathscr{L}(\Theta + \Delta\Theta) < L$ **then**
    Set $\Theta := \Theta + \Delta\Theta$ and $L := \mathscr{L}(\Theta)$.
Display Θ.

2.2 Linear predictor using distances in the state space

A different technique to find a linear predictor. We define a cost function.

$$J(X, Y, f) = \sum_{\mathbf{x} \in \mathcal{S}(X)} \left\{ \delta(f(\mathbf{x}) = 1) \sum_{(\mathbf{x}', y') \in \mathcal{S}(X,Y)} \delta(y' = 1) \|\mathbf{x} - \mathbf{x}'\|^2 + \delta(f(\mathbf{x}) = 0) \sum_{(\mathbf{x}', y') \in \mathcal{S}(X,Y)} \delta(y' = 0) \|\mathbf{x} - \mathbf{x}'\|^2 \right\} \quad (17)$$

We define the quadratic distance between a sample and a set of samples

$$d(\mathbf{x}, \mathcal{S}) = \sum_{\mathbf{x}' \in \mathcal{S}} \|\mathbf{x} - \mathbf{x}'\|^2 \quad (18)$$

$$f(\mathbf{x}) = \left\{ \begin{array}{ll} 1 & \text{if } d(\mathbf{x}, \{(\mathbf{x}, y) \in \mathcal{S}(\mathbf{X}, Y) | y = 1\}) \leq d(\mathbf{x}, \{(\mathbf{x}, y) \in \mathcal{S}(\mathbf{X}, Y) | y = 0\}) \\ 0 & \text{if not} \end{array} \right. \quad (19)$$

Theorem 1. *For a given distance derived from a scalar product,* $\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = (\mathbf{x}_1 - \mathbf{x}_2).(\mathbf{x}_1 - \mathbf{x}_2)$, *we have*

$$\underset{\mathbf{x} \in \mathcal{S}}{\operatorname{argmin}}\, d(\mathbf{x}, \mathcal{S}) = \left\| \mathbf{x} - \frac{1}{\#\mathcal{S}} \sum_{\mathbf{x}' \in \mathcal{S}} \mathbf{x}' \right\|^2 \quad (20)$$

The predicting function obtained when minimizing $J(\mathbf{X}, Y, f)$ is a linear predictor. Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be respectively the means of the 0-labeled samples and the 1-labeled samples. Let $\mathbf{x}_I$ be the middle of $\mathbf{x}_0$ and $\mathbf{x}_1$.

$$f(\mathbf{x}) = \delta\left((\mathbf{x}_1 - \mathbf{x}_0).(\mathbf{x} - \mathbf{x}_I) \geq 0\right) \tag{21}$$

We denote this predictor $\mathbb{SD}_{\mathbf{X},Y}$ for Shortest Distance.

Unfortunately, this is not so straightforward in multiclass learning problems.

Link distance and matrix.

2.3 Unsupervised classification

The objective is a partition $\mathcal{S}(\mathbf{X}) = \{\mathcal{S}(\mathbf{X}) | f(\mathbf{x}) = 1\} + \{\mathcal{S}(\mathbf{X}) | f(\mathbf{x}) = 0\}$

We assume we know only $\mathbf{X}$ and not Y.

2.3.1 Main algorithm

Algorithm 2 Simplified implementation kmeans

Require: $\mathbf{X}$

Ensure: $f \in 0, 1^{\mathcal{X}}$

1: Select randomly Y a binary vector.
2: **repeat**
3:     **for** $\mathbf{x}_n \in \mathcal{S}(\mathbf{X})$ **do**
4:         Compute $y'_n = \mathbb{SD}_{\mathbf{X},Y}(\mathbf{x}_n)$ and $Y' = [y'_n]$
5: **until** Y'=Y

Exercise 10. **(8)** *Give the Octave code that uses* `simulated_annealing` *to find an approximation of* $\mathbf{a}$ *and* a *of exercise 9.*

Exercise 11. **(9)** *We consider once again exercise 9 to solve without using the trick of zeroing* $\mathcal{J}$ *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*

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

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

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* $\overset{\triangle}{\mathbf{X}}^T \widetilde{Y}$

4. *Show that when* $a_1 = -\frac{2}{7}$, $a_2 = \frac{8}{7}$ *and* $b = 1$, *we have indeed that* $\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = 0$.

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 $\mathcal{J}$-derivative is a global minimum, show that

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

Exercise 12. **(10)** *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*

Exercise 13. **(11)** *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.*

Exercise 14. **(12)** *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 $\mathscr{A}_{opt}$ *with* $(\mathbf{a}_{opt}, b_{opt})$ *and* $\mathcal{S}_{opt}$.
repeat
  $(\mathbf{x}, (\mathbf{x}', y')) = \text{argmin}_{\mathbf{x} \in \mathcal{S}_4, (\mathbf{x}', y') \in \mathcal{S}_3} d(\mathbf{x}', \mathbf{x})$
  Set $\mathcal{S} = \mathcal{S}_{opt} \bigcup (\mathbf{x}, y')$
  $(\mathbf{a}, b) = LEARN(\mathcal{S})$
  Compute $\mathscr{A}$ *with* $(\mathbf{a}, b)$ *and* $\mathcal{S}$
  if $\mathscr{A} > \mathscr{A}_{opt}$ ***then***
    $(\mathbf{a}_{opt}, b_{opt}) = (\mathbf{a}, b)$, $\mathcal{S}_{opt} = \mathcal{S}$, $\mathscr{A}_{opt} = \mathscr{A}$.
until $\mathscr{A} <= \mathscr{A}_{opt}$

Exercise 15. **(13)** *We consider the following confusion matrix.*

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

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 16. **(14)** *Let Y be a uniform binary random variable and X when conditioned to 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 $Y = 0$ on a given experiment?*
2. *What is the probability density function that $X = [x_1, x_2]$ given $Y = 0$ and then given $Y = 1$?*
3. *We now assume that $\sigma_0 = \sigma_1 = \sigma$, show that a straight line separates points that are more likely when $Y = 1$ from the more likely points when $Y = 0$.*

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

Exercise 17. **(15)** *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.*

Exercise 18. **(16)** *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*
   $$\binom{6}{1} p_0^5(1-p_0) \times \binom{6}{1} p_1^5(1-p_1)$$

Exercise 19. **(17)** *Let* $\mathbf{X}$ *be a feature matrix. Show that there exists* β_f *such that* $\mathbf{X}' = \mathbf{X} - [\beta_1 \ldots \beta_F]$ *is centered.*

Exercise 20. **(18)** *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 21. (19) *The exercises 17 and 18 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* $N \times 1$.

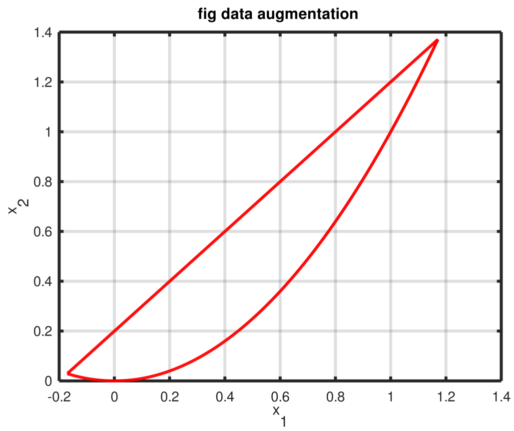
1. *Define a matrix* $\mathbf{H}$ *of size* $N \times N$ *such that* $\mathbf{HX}$ *is centered (i.e. the sums of each column of* $\mathbf{HX}$ *are null).*

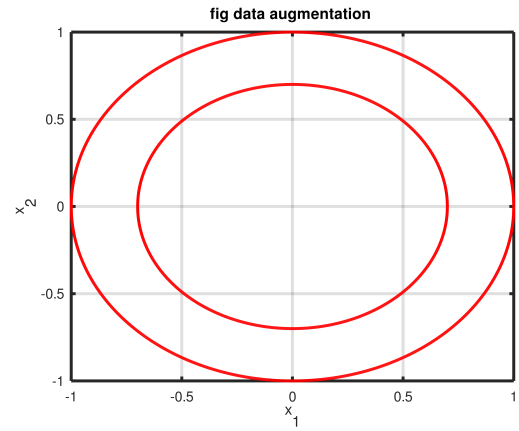
2. *Show that* $\mathbf{HX}\left(diag(\mathbf{X}^T\mathbf{H}^2\mathbf{X})\right)^{-\frac{1}{2}}$ *is centered and normalized.*

3. *Write the Matlab/Octave implementation of* $\mathbf{HX}\left(diag(\mathbf{X}^T\mathbf{H}^2\mathbf{X})\right)^{-\frac{1}{2}}$

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

Exercise 22. (20) *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.*

$$f(\overset{\omega}{\mathbf{x}}) = \delta(b_1 - \mathbf{a}_1.\overset{\omega}{\mathbf{x}})\delta(b_2 - \mathbf{a}_2.\overset{\omega}{\mathbf{x}})$$

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

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

$$f(\overset{\omega}{\mathbf{x}}) = \delta(b_1 - \mathbf{a}_1.\overset{\omega}{\mathbf{x}})\delta(b_2 - \mathbf{a}_2.\overset{\omega}{\mathbf{x}})$$

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

Exercise 23. (23) *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* $\|\mathbf{x}\|^2$, $\|\mathbf{x} - \mathit{PCA}_{\mathcal{T}}(\mathbf{x})\|^2$
6. *Compute* $\mathscr{A}_{\mathcal{T}PCA}$
7. *Check the* $\mathbf{X}$*-signification of* $\mathscr{A}_{\mathcal{T}PCA}$

Exercise 24. (24) *We consider two independant Gaussian random variable* $\overset{r}{z}_1$ *and* $\overset{r}{z}_2$ *centered and normalised.*

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

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* $\Sigma = E\left[(\overset{r}{\mathbf{x}} - \boldsymbol{\mu})^T(\overset{r}{\mathbf{x}} - \boldsymbol{\mu})\right]$

Exercise 25. (25) *We consider a centered multivariate normal distribution*

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

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.*

Exercise 26. (26) *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 27. (**27**) *We consider the same centered multivariate normal distribution as defined in exercise 25.*

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

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 24 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 28. (**28**) *We consider the tiny dataset of exercise 23with*

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

1. *Compute the correlation matrix.*

Exercise 29. (**32**) *We consider again exercise 9 and the proposed solution in exercise 9 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

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

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?*

Exercise 30. (21) *We consider a small dataset*

$$\begin{aligned}\mathbf{x}_1 &= [1,\ 0]\\ \mathbf{x}_2 &= [0,\ 1]\\ \mathbf{x}_3 &= [1,\ 1]\end{aligned}$$

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 $\mathcal{K}$

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

1. *Express $\mathcal{K}$ as function of $[x_1, x_2]$ and $[x_1', x_2']$. Is it left-linear, right-linear?*
2. *Compute $\mathbf{K} = [\mathcal{K}(\mathbf{x}_m, \mathbf{x}_n)]_{m,n}$*
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

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

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

2.3.2 Information we cannot retrieve

3 Using the Data Set to Test the Statistical Inferences

4 Dimensionality Reduction and Feature Selection

5 Spatial Context

6 Spatial Prior

7 Texture Descriptors

8 Abundance Classification

9 Supplementary

A **distance**, denoted here $d(\mathbf{x},\mathbf{x}')$ is a non-negative value indicating to what extent two samples are different. It fulfills the following properties:

$$d(\mathbf{x},\mathbf{x}') \geq 0, \quad d(\mathbf{x},\mathbf{x}') = d(\mathbf{x}',\mathbf{x}), \quad d(\mathbf{x},\mathbf{x}') = 0 \Rightarrow \mathbf{x} = \mathbf{x}', \quad d(\mathbf{x},\mathbf{x}') + d(\mathbf{x}',\mathbf{x}'') \geq d(\mathbf{x},\mathbf{x}'') \tag{22}$$

Exercise 31. *Use a computer to check that* $(\mathbf{x},\mathbf{x}')\mapsto(\mathbf{x}-\mathbf{x}')\centerdot(\mathbf{x}-\mathbf{x}')$ *is a distance.*

10 Probabilistic Framework

A Correction of exercises

Answer. *Correction of exercise 17 Let* β_f *be defined as*

$$\beta_f = \frac{1}{N}\sum_{n=1}^{N} X_{nf}$$

We then get for any $f\in\{1\dots F\}$

$$\sum_{n=1}^{N} X'_{nf} = \sum_{n=1}^{N}(X_{nf}-\beta_f) = \sum_{n=1}^{N} X_{nf} - N\beta_f = 0$$

Answer. *Correction of exercise 6*

1. *The blue curve is the upper curve shown in figure 1.*
2. *Red curve shown is the lower curve shown in figure 1.*

Answer. *Correction of exercise 18 For any given* $f\in\{1\dots F\}$*, let* α_f *be defined as*

$$\alpha_f = \frac{1}{\sqrt{\frac{1}{N}\sum\limits_{n=1}^{N} x_{nf}^2}}$$

we get

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

B Octave code for figures

Simulation of slide 133 and **??**

```
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
```

Simulation of slide 15

```
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
```

Simulation of slide 27
Simulation of slide 32
Code of slide 43

```
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
```

Code of slide 46

```
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);
  figure(1); plot(it_l,L_l,'linewidth',3); title('L');
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('number changes'); ylabel('loss function');
  prin(1);
  b=1; a1=-2/7; a2=8/7;
  figure(2); plot(it_l,b_l-b,'b',it_l,a1_l-a1,'r',it_l,a2_l-a2,'g','linewidth',3); title(
  set(gca, "linewidth", 3, "fontsize", 16)
  legend('b-b^*','a_1-a^*_1', 'a_2-a^*_2');
  xlabel('number changes'); ylabel('error on parameter values');
  prin(2);
  figure(3); plot(it_l,k_l,'linewidth',3); title('k');
  xlabel('number changes'); ylabel('number of iterations');
  set(gca, "linewidth", 3, "fontsize", 16)
  xlabel('number changes'); ylabel('error on parameter values');
  prin(3);
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
```

Code of slide 68

```
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','line
    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
```

Code of slide 101

```
function fig_testing_inferences()
  name='fig_testing_inferences_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];
  C=[5 1; 1 5];
  if ~exist(name_data)
    [acc_l,fZ_l,n]=mk_hZ(C,12000);
    nZ=n;
    save(name_data,'acc_l','fZ_l','n');
  else
    load(name_data); nZ=n;
  end
  name1='fig_modeling_inferences_';
  name1_data=['./prg/',name1,'data.mat'];
  load(name1_data);
  fZh_l=[(1-Ep0)*(1-Ep1),  (1-Ep0)*Ep1+Ep0*(1-Ep1)+(1-Ep0)*(1-Ep1), Ep0*Ep1];
  fZ_l,n,acc_l
  bar(acc_l',[fZ_l;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 model'),
  prin([]);
  figure(2);
  bar(acc_l',fZ_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'),
  prin(2);

end

function [acc_l,fZ_l,n]=mk_hZ(C,time)
  if 1==nargin time=50; end
  y=[zeros(sum(C(1,:)),1); ones(sum(C(2,:)),1)];
  [acc_l,fZ_l]=start_fZ();
  tic,
  n=0;
  while(toc<time)
    [mu0,mu1,sigma0,sigma1,w,b]=draw_pb();
    yh=draw_data(y,mu0,mu1,sigma0,sigma1,w,b);
    if is_ok_C(y,yh,C)
      n=n+1;
```

```
      yh2=draw_data([0;1],mu0,mu1,sigma0,sigma1,w,b);
      acc_moy=((yh2(1)==0)+(yh2(2)==1))/2;
      fZ_l=adapt(fZ_l,acc_moy);
    end
  end
  fZ_l=fZ_l/sum(fZ_l);
end

function [acc_l,fZ_l]=start_fZ()
  acc_l=[0 0.5 1];
  fZ_l=zeros(1,3);
end

function fZ_l=adapt(fZ_l,acc)
  if 0==acc
    fZ_l(1)=fZ_l(1)+1;
  elseif 1==acc
    fZ_l(3)=fZ_l(3)+1;
  else
    fZ_l(2)=fZ_l(2)+1;
  end
end

function ok=is_ok_C(y,yh,C)
  Cf=@(yq,yhq)sum((y==yq)&(yh==yhq));
  Ch=[Cf(0,0) Cf(0,1); Cf(1,0) Cf(1,1)];
  ok=all(all(C==Ch));
end

function yh=draw_data(y,mu0,mu1,sigma0,sigma1,w,b)
  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
```

Code of slide 107
Code of slide 117

```
function fig_modeling_inferences()
  name='fig_modeling_inferences_';
```

```
  name_title=name; name_title(name_title=='_')=' ';
  name_data=['./prg/',name,'data.mat'];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  C=[5 1; 1 5];
  if ~exist(name_data)
    [p_l,f0_l,f1_l,n]=mk_hist_p0p1(C,120);
    Ep0=sum(p_l.*f0_l), Ep1=sum(p_l.*f1_l),
    save(name_data,'p_l','f0_l','f1_l','Ep0','Ep1','n');
  else
    load(name_data);
  end
  Q=p_l(2)-p_l(1);
  Ep0=sum(p_l.*f0_l), Ep1=sum(p_l.*f1_l),
  Ep0*Ep1, (1-Ep0)*Ep1+Ep0*(1-Ep1)+(1-Ep0)*(1-Ep1), (1-Ep0)*(1-Ep1),
  f_th=p_l.^5.*(1-p_l); f_th=f_th/sum(f_th);
  figure(1); plot(p_l,f0_l/sum(f0_l)/Q,'linewidth',3,p_l,f1_l/sum(f1_l)/Q,'linewidth',3,p_
  title(['C_{11}=C_{22}=5; C_{12}=C_{21}=1; n=',num2str(n),' \it ',name_title]),
  set(gca, "linewidth", 3, "fontsize", 12)
  legend('f_0(p)','f_1(p)','f_{th}(p)','location','northwest'),
  prin([]);
end
```

Code of slide 125

```
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
```

Code of slide **??** and 244

```
function fig_ill_conditioned
  name='fig_ill_conditioned_';
  name_title=name; name_title(name_title=='_')=' ';
  name_data=['./prg/',name,'data.mat'];
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  x1=[2 0.5];
  x2=[1 2];
  x3=[0 0];
  y1=1; y2=0; y3=1;
  Xd=[[x1; x2; x3] ones(3,1)];
  Yt=2*[y1; y2; y3]-1;
  sigma_l=1./sqrt((1:1e3));
  val_l=zeros(size(sigma_l));
  kmax=length(sigma_l);
  for k=1:length(sigma_l)
    sigma=sigma_l(k);
    Xd(:,1)=2+sigma*randn(3,1);
    val_l(k)=max(max(inv(Xd'*Xd)));
    if rcond(inv(Xd'*Xd))<1e-12
      kmax=k; break;
    end
  end
  figure(1); semilogy(1./sigma_l(1:kmax),val_l(1:kmax),'b-','linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 12)
  title(name_title),
```

```
  prin(1);
end
```

Code of slide 144 and 20

```
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
```

Code of slide 151

```
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
```

Code of slide 152

```
function fig_norm_feature_construction3()
  name='fig_norm_feature_construction3_';
  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=0.3:0.5:3;
  F=2; Fc=5;
  tab_l=[];
  if ~exist(name_data)
    for cpt=1:length(norm_value_l)
      norm_value=norm_value_l(cpt);
      y_mean=0;
      for cpt2=1:100
        xc=randn(1,Fc);
        xc=norm_value*xc/norm(xc);
        y_mean=y_mean+dist_F(xc);
      end
      y_mean=y_mean/100;
      disp(num2str([norm_value,y_mean])),
      tab_l=[tab_l; norm_value, y_mean];
      save(name_data,'tab_l','norm_value_l','F');
    end
    save(name_data,'tab_l','norm_value_l','F');
  else
```

```
    load(name_data);
  end
  figure(1); plot(tab_l(:,1),tab_l(:,2),'+-','linewidth',3);
  title(name_title),
  axis([0 3 0 3]);
  xlabel('distance in the augmented space'),ylabel('average distance with the closest poi
  grid,
  set(gca, "linewidth", 3, "fontsize", 14)
  prin(1);
end


function y=dist_F(xc)
%computes the distance between a sample of the five-dimension space
%and the closest member of the constructed feature space.
  xc=xc(:).';
  fun=@(x)norm(constr1(x)-xc);
  x_min=simulated_annealing(fun,2,'silent');
  y=fun(x_min);
end

function y=fun1(xc,x,norm_value)
%x is in F and y is the norm of aug vector x
  xc=xc(:)';
  x=x(:)';
  if norm(xc)>1e-10
    x=norm_value/norm(x)*x;
  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
```

Code of slide 153

```
function fig_geodesic()
  name='fig_geodesic_';
  name_title=name; name_title(name_title=='_')=' ';
  prin=@(num)eval(['print (''-r600'', ''./images/',name,num2str(num),'.png'')']);
  name_data=['./prg/',name,'data.mat'];

  xa=[1 0];
  xb=[0 1];
  alpha_l=0:1e-2:1;
  x_l=zeros(length(alpha_l),2);
  if ~exist(name_data)
    for alpha_=1:length(alpha_l)
      alpha=alpha_l(alpha_);
      xc=alpha*constr1(xa)+(1-alpha)*constr1(xb);
      fun1=@(x)norm(xc-constr1(x(:)'));
      xmin=simulated_annealing(fun1,2,'silent');
      x_l(alpha_,:)=xmin(:)';
```

```
      disp(num2str([alpha,xmin(:)']))
      save(name_data,'alpha_l','x_l');
    end
  else load (name_data);
  end
  figure(1); plot([xa(1) xb(1)],[xa(2) xb(2)],'linewidth',3,x_l(:,1),x_l(:,2),'linewidth'
  title([' \it ',name_title]),
  set(gca, "linewidth", 3, "fontsize", 12)
  legend('original feature space','constructed feature space','location','northwest'),
  prin([]);

end


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

Code of slide 171

```
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
```

Code of slide 194

```
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
```

Code of slide 199

```
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
```

Code of slide 203

```
function fig_explain_variance2()
  name='fig_explain_variance2_';
  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,V_exp2_l,V_ass_l]=do_task();
    save(name_data,'V_exp_l','V_exp2_l','V_ass_l');
  else
    load(name_data);
  end
  M=mean(V_exp_l),
  [n,v_exp]=hist(V_exp_l,100);
  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)
  legend('Probability distribution',['Mean=',num2str(M)]);
  title(name_title),
  prin(1);

  M2=mean(V_exp2_l./V_ass_l),
  [n2,v_exp2]=hist(V_exp2_l./V_ass_l,100);
  n2_h=n2/sum(n2);
  figure(3); plot(v_exp2,n2_h,'linewidth',3,[M2 M2],[0 max(n2_h)],'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([0 5 0 Inf])
  legend('Probability distribution of the error',['Mean=',num2str(M2)]);
  title(name_title), grid,
  prin(2);

  M3=mean(V_exp2_l./V_exp_l./V_ass_l),
  [n3,v_exp3]=hist(V_exp2_l./V_exp_l./V_ass_l,100);
  n3_h=n3/sum(n3);
  figure(4); plot(v_exp3,n3_h,'linewidth',3,[M3 M3],[0 max(n3_h)],'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16,'xgrid','on','ygrid','on');
  axis([0 2 0 Inf])
  legend('Probability distribution of the relative error',['Mean=',num2str(M3)]);
  title(name_title), grid
  prin(3);


  M4=mean(V_exp2_l-V_ass_l),
  [n4,v_exp4]=hist(V_exp2_l-V_ass_l,100);
  %[n4,v_exp4]=hist(V_exp2_l-V_ass_l,[0.1:0.2:3].^2);
  n4_h=n4/sum(n4);
  figure(5); plot(v_exp4,n4_h,'linewidth',4,[M4 M4],[0 max(n4_h)],'linewidth',3);
```

```
  set(gca, "linewidth", 3, "fontsize", 16)

  legend('Probability distribution of the error',['Mean=',num2str(M4)]);
  title(name_title), grid,
  prin(4);

  M5=mean(V_exp2_l./V_exp_l-V_ass_l),
  %[n5,v_exp5]=hist(V_exp2_l./V_exp_l-V_ass_l,[0.1:0.2:3].^2);
  [n5,v_exp5]=hist(V_exp2_l./V_exp_l-V_ass_l,100);
  n5_h=n5/sum(n5);
  figure(6); plot(v_exp5,n5_h,'linewidth',3,[M5 M5],[0 max(n5_h)],'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16,'xgrid','on','ygrid','on');

  legend('Probability distribution of the relative error',['Mean=',num2str(M5)]);
  title(name_title), grid
  prin(5);


  M3=mean(sqrt(V_exp2_l./V_exp_l)./sqrt(V_ass_l)),
  [n3,v_exp3]=hist(sqrt(V_exp2_l./V_exp_l)./sqrt(V_ass_l),100);
  n3_h=n3/sum(n3);
  figure(7); plot(v_exp3,n3_h,'linewidth',3,[M3 M3],[0 max(n3_h)],'linewidth',3);
  set(gca, "linewidth", 3, "fontsize", 16)
  axis([0 5 0 Inf])
  legend('Probability distribution of the square root of relative error',['Mean=',num2str
  title(name_title), grid,
  prin(6);


end

function [V_exp_l,V_exp2_l,V_ass_l]=do_task()
  It=1e6; N=5;
  V_exp_l=zeros(1,It);
  V_exp2_l=zeros(1,It);
  V_ass_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';
    [P,v_ass]=pca1(Sigma_2);
    xpca=x*P(:,1);
    V_ass_l(cpt)=v_ass;
    V_exp2_l(cpt)=(x-xpca)*(x-xpca)';
  end
end


function [P,v_ass]=pca1(Sigma_2)
  [V1,D1,W]=eig(Sigma_2'*Sigma_2);
  [~,ind]=sort(diag(D1));
  P=V1*eye(size(D1))(ind,:);
  D=D1*eye(size(D1))(ind,:);
```

```
  if ~(abs(1-sum(diag(D)))<1e-6)                              error('pb'), end
  v_ass=1-D(1);
end

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

Code of slide 158

```
function check_ex30
  augm=@(x)[x(1),x(2),x(1)^2,x(1)*x(2),x(2)^2];
  kernel=@(xa,xb)sum(augm(xa).*augm(xb));
  X=[1 0; 0 1; 1 1];
  Xaugm=[augm(X(1,:));augm(X(2,:));augm(X(3,:))];
  K=zeros(3);
  for m=1:3
    for n=1:3
      K(m,n)=kernel(X(m,:),X(n,:));
    end
  end
  K,
  det(K)
  Ki=inv(K)
  Kfun=@(x)[kernel(x,X(1,:)),kernel(x,X(2,:)),kernel(x,X(3,:))]*Ki;
  Kfun(X(1,:)), Kfun(X(1,:))*X, Kfun(X(1,:))*Xaugm,
  Kfun(X(2,:)), Kfun(X(2,:))*X, Kfun(X(2,:))*Xaugm,
  Kfun(X(3,:)), Kfun(X(3,:))*X, Kfun(X(3,:))*Xaugm,
  x4=[1 -1];
  augm(x4),
  [augm(X(1,:))',augm(X(2,:))',augm(X(3,:))',augm(x4)',]
end
```

Code of slide **??**

```
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
```

Code of slide 233

```
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
```

Code of slide 238

```
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
```

Code of slide 239

```
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_regularization5()
%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)
    noise_l=do_task();
    save(name_data,'noise_l');
  else
    load(name_data);
  end
  [noise1,fnoise1,sigma_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_noise]=dist_est(norm_noise);
  clear noise_l; clear norm_noise;

  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);
  axis([-2*sigma_noise1 2*sigma_noise1 0 Inf]),
  prin(1);

  figure(2); plot(noise_norm,fnoise_norm,'linewidth',3,noise_norm,f_gauss(noise_norm,sigma
    noise_norm,f_laplace(noise_norm,sigma_noise_norm/2),'linewidth',3);
  axis([-2*sigma_noise_norm+m_noise 2*sigma_noise_norm+m_noise 0 Inf]),
  prin(2);

end

function noise_l=do_task()
  dim=10;
  E=1e4;
  noise_l=[];
  for exp=1:E
    if 0==mod(exp,100) disp(['progression=',num2str(exp/E)]), end
    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);
    [Xq,Yq]=prepa1(100,mu0,mu1,sigma);
    noise_l=[noise_l; c_noise(Xq,Yq,w)];
  end

end

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

```
end

function noise=c_noise(X,Y,w)
  Xe=[X ones(size(X,1),1)];
  noise=Y-w*Xe';
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 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
```

Code of slide 252

```
function fig_regularization2()
%prior of problem A
  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();
    %[lambda_l,acc_l,acc_mu_l]=do_task();
    save(name_data,'a_l');
  else
    load(name_data);
  end
  [n,norm_a_exp]=hist(sqrt(sum(a_l.^2,2)),1000);
  n_norm_a_exp=n/sum(n);
  figure(1); plot(norm_a_exp,n_norm_a_exp,'linewidth',3);
  [n,b_exp]=hist(a_l(:,end),1000);
  n_b_exp=n/sum(n);
  figure(2); plot(b_exp,n_b_exp,'linewidth',3);
  [n,a1_exp]=hist(a_l(:,1),1000);
```

```
  n_a1_exp=n/sum(n)/(a1_exp(2)-a1_exp(1));
  figure(3); plot(a1_exp,n_a1_exp,'linewidth',3);
  sigma_a1=std(a_l(:,1));
  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(4); plot(a1_exp,n_a1_exp,'linewidth',3,a1_exp,f_gauss(a1_exp,sigma_a1),'linewidt
    a1_exp,f_laplace(a1_exp,sigma_a1/2),'linewidth',3);
  axis([-2*sigma_a1 2*sigma_a1 0 Inf]),

  [n,a2_exp]=hist(a_l(:,2),1000);
  n_a2_exp=n/sum(n)/(a2_exp(2)-a2_exp(1));
  sigma_a2=std(a_l(:,2));
  figure(6); plot(a2_exp,n_a2_exp,'linewidth',3,a2_exp,f_gauss(a2_exp,sigma_a2),'linewidt
    a2_exp,f_laplace(a2_exp,sigma_a2/2),'linewidth',3);
  axis([-2*sigma_a2 2*sigma_a2 0 Inf]),

  [n,a3_exp]=hist(a_l(:,3),1000);
  n_a3_exp=n/sum(n)/(a3_exp(2)-a3_exp(1));
  sigma_a3=std(a_l(:,3));
  figure(7); plot(a3_exp,n_a3_exp,'linewidth',3,a3_exp,f_gauss(a3_exp,sigma_a3),'linewidt
    a3_exp,f_laplace(a3_exp,sigma_a3/2),'linewidth',3);
  axis([-2*sigma_a3 2*sigma_a3 0 Inf]),

  b_m=mean(a_l(:,end));
  [n,b_exp]=hist(a_l(:,end),1000);
  n_b_exp=n/sum(n)/(b_exp(2)-b_exp(1));
  sigma_b=std(a_l(:,end));
  figure(5); plot(b_exp,n_b_exp,'linewidth',3,b_exp,f_gauss(b_exp-b_m,sigma_b),'linewidth
    b_exp,f_laplace(b_exp-b_m,sigma_b/2),'linewidth',3);
  axis([-2*sigma_b+b_m 2*sigma_b+b_m 0 Inf]),


end


function a_l=do_task()
  dim=10;
  E=1e4;
  a_l=[];
  for exp=1:E
    mu1=2*rand(1,dim); mu0=2*rand(1,dim);
    supp=20*rand(1,dim);
    mu0=mu0+supp; mu1=mu1+supp;
    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
```