Optimization

Charles Soussen, Ronan Rialland

October 2018

1 Training based on matlab slides

Please check the examples on the matlab slides `transpmatlab.pdf` and test the given examples : type the matlab command line and check out the visual printout.

In particular, it is important to learn how to :
- — write a matlab script (for instance, name it `example.m`) saving all matlab commands.
- — write a matlab function like the sinuscardinal function given in slide 11 (create matlab script `sincardinal.m` containing the code of the function).
- — call the matlab function with a matlab command (*e.g.,* call `sincpi = sinuscardinal(pi);` or directly `sinuscardinal(pi)`, see slide 11).

2 Use of the optimization toolbox of matlab

Generally speaking, the principle is to :

1. Define the objective function in Matlab :
   `f = objfun(x);`
   or
   `[f,G] = objfun(x);`
2. Define the linear constraints ($\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, $\boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{b}_{\text{eq}}$) : define matrices and vectors $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{A}_{\text{eq}}$, $\boldsymbol{b}_{\text{eq}}$.
3. Define the bound constraints $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$: define vectors $\boldsymbol{l}$ and $\boldsymbol{u}$.
4. Define the nonlinear constraints : write a function
   `[c,ceq] = confun(x);`
5. Call matlab function :
   `[x,fval,exitflag] = xxxx(@objfun, x0, A, B, Aeq, Beq, l, u, @confun, options);`
   where xxxx is the optimization solver (`fminunc`, `fmincon`, `lsqlin`, *etc.*).

Help : type

```
doc optim
optimtool
help optimoptions
help fminunc
doc fminunc
```

3 Knapsack problem

Jo goes hitch hiking. The maximum weight allowed in his knapsack is W. Each article $i = 1, \dots, n$ he can take weight w_i and has a usefulness u_i. What articles should be taken in the knapsack to maximize the usefulness ?

1. Binary case : each article can be taken at most once.

2. General case : each article can be taken several times.

Application. For $W = 25$, search for the ideal knapsack. Same question for $W = 26$.

I	w_i	u_i
1	25	40
2	12,5	35
3	11,25	18
4	5	4
5	2,5	10
6	1,25	2

4 Plot of a 2D function $z = f(x, y)$

```
% definition of x and y
x=-10:0.1:10;
y=-0.4:0.1:10;
% define a grid (x,y)
[xx,yy]=meshgrid(x,y);
%
% Evaluation of f(x,y) on this grid
%
zz = f(xx,yy);  %%%% TO DEFINE
% 3D surface
figure(1), surf(x,y,zz), colormap hsv
camlight;
shading interp
lighting gouraud
view(3)

% Visualize the level sets:
figure(2),
contour(x,y,zz,[0:1:10]);
%or contour3(x,y,zz,[0:1:10]);
```

Example : Plot Stybilinski-Tang function $f(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{n}(x_j^4 - 16x_j^2 + 5x_j)$ for $n = 2$.

5 2D unconstrained minimization

Find the minimizer of

$$f(x,y) \quad = \quad \left[y - \cos(2x) - \frac{x^2}{10}\right]^2 \; + \; \exp\left(\frac{x^2+y^2}{100}\right).$$

3. Visualize the objective function in 3D. Visualize its level sets.

4. Try different initial conditions and different optimization algorithms (quasi-Newton, least squares, simplex).

5. Check the exitflag status and understand why the algorithm stopped.

6. Display the progress of algorithm per iteration (set optimization option `Display` to `iter`).

7. Include the computation of the gradient in the objective function and modify the `SpecifyObjectiveGradient` option. Validate (temporarily) the gradient calculation by activating the `CheckGradients` option.

6 Constrained minimization

8. Same problem with constaint $4 - x \leq y$.

9. Same problem with the constaints $4 - x \leq y$ and $x^2 \leq y$ (compute the gradient of the nonlinear constraint).

10. In both cases, comment on the values found for the Lagrange multipliers.

7 Unconstrained problem of large dimension

Minimize the following cost function

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} 100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2$$

over $\boldsymbol{x} \in \mathbb{R}^n$ for $n = 2$, 10, 100, and 1000.

Use `tic` and `toc` to measure the execution time :

```
tic
% call optimization solver
....
toc
```

Compare the results obtained by exploiting the knowledge and sparse structure of the gradient and the Hessian matrix (Matlab commands : sparse, full). Comparisons are done in terms of accuracy and computation time.

Remark : the specific structure of cost function $f(\boldsymbol{x})$ enables to use different optimization solvers, in particular least-squares solvers. Compare the use of least-squares solvers with the use of `fmincon`.

8 Least squares

Write the following matlab program `generate_data.m` :

```
clear all
close all
randn('seed',3);  % always the same source of randomness

theta_1 = 5;      % ground truth
theta_2 = 1;      % ground truth

x = 0:0.3:19;
y = exp(-x/theta_1)-0.8*exp(-x/theta_2);
N = length(x);

noise = 0.03;     % noise level (can be changed)
z = y + noise*randn(1,N);

figure(1), clf, plot(x,z,'o','linewidth',2); grid on;
save 'data0.mat'
```

which generates simulated data (x, z). The data are saved in the file `data0.mat`. They can be loaded using `load data0.mat`

We would like to approximate the data y_k using the model $f(x; \alpha, \beta) = \alpha \exp(-x/\beta)$ with $\beta > 0$. Write another Matlab program `lsq_approximation.m` which numerically computes the values of α and β corresponding to the minimum squared error.

Same question using the model $f(x; \alpha_1, \beta_1, \alpha_2, \beta_2) = \alpha_1 \exp(-x/\beta_1) + \alpha_2 \exp(-x/\beta_2)$ with $\beta_1 > 0$ and $\beta_2 > 0$.

Conclusions ?