优化多变量无约束函数利器：fminunc

zcl.space

目录

1 fminunc

matlab提供了一个优化多变量无约束目标函数的利器：fminunc。

其实现的功能是：

$$\min_x f(x) \tag{1.1}$$

2 应用1： x = fminunc(fun,x0)

最简单的应用是：

```
x = fminunc(fun,x0)
```

其中 x = fminunc(fun,x0) 提供了一个起始点x_0，供 fminunc 使用。fminunc 试图为目标函数找到局部最小解。

举个例子，假设要寻找函数$f(x) = 3x_1^2 + 2x_1x_2 + x_2^2 - 4x_1 + 5x_2$的最小值。我们首先为这个函数提供一个匿名函数句柄：

```
fun = @(x)3*x(1)^2 + 2*x(1)*x(2) + x(2)^2 - 4*x(1) + 5*x(2);
```

调用 fminunc：

```
x0 =[1,1];
[x,fval] = fminunc(fun,x0);
```

经过若干次迭代后, x 返回最小值的位置， fval 返回目标函数在这个位置的最小值。

```
x,fval
```

```
x =

    2.2500   -4.7500


fval =

  -16.3750
```

3 应用2： x = fminunc(fun,x0,options)

当提供梯度结果的时候， fmiunc 的计算结果会大大加快。比如对于多变量Rosenbrock函数：

$$f(x) = 100(x_1 - x_1^2)^2 + (1 - x_1)^2 \tag{3.1}$$

其梯度为：

$$\partial f(x) = \begin{bmatrix} -400(x_1 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix} \tag{3.2}$$

matlab 代码为：

```
function [f,g] = rosenbrockwithgrad(x)
% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
```

```
if nargout > 1 % gradient required
    g = [-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
         200*(x(2)-x(1)^2)];
end
```

为这个目标函数的梯度提供一些参数：

```
options = optimoptions('fminunc','Algorithm','trust-region','SpecifyObje
```

设定初始值为 $[-1,2]$，然后调用 fminunc

```
x0 = [-1,2];
fun = @rosenbrockwithgrad;
x = fminunc(fun,x0,options)
```

当梯度的值小于预先设定的最小值时，优化过程结束，程序返回：

```
x =
  1.0000  1.0000
```

4 应用3： x = fminunc(problem)

对于应用2，我们可以用更加结构化的方式来描述。首先还是针对Rosenbrock函数：

$$f(x) = 100(x_1 - x_1^2)^2 + (1 - x_1)^2 \tag{4.1}$$

其梯度为：

$$\partial f(x) = \begin{bmatrix} -400(x_1 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix} \tag{4.2}$$

matlab代码跟之前一样：

```
function [f,g] = rosenbrockwithgrad(x)
% Calculate objective f
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;

if nargout > 1 % gradient required
    g = [-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
         200*(x(2)-x(1)^2)];
end
```

为目标函数梯度设定参数：

```
options = optimoptions('fminunc','Algorithm','trust-region','SpecifyObje
```

为所有的参数创建一个结构体：

```
problem.options = options;
problem.x0 = [-1,2];
problem.objective = @rosenbrockwithgrad;
problem.solver = 'fminunc';
```

然后调用 fminunc ：

```
x = fminunc(problem)
```

5 应用4：[x,fval] = fminunc(fun,x0)

fminunc 函数不仅可以找到最小值的位置，也可以返回最小值。只要对其添加第二个输出，就是局部最小值。

比如目标函数是：

$$f(x) = x(1)e^{-||x||_2^2} + ||x||_2^2/20 \tag{5.1}$$

其MATLAB代码为：

```
fun = @(x)x(1)*exp(-(x(1)^2 + x(2)^2)) + (x(1)^2 + x(2)^2)/20;
```

设定初始值，并调用 fminunc ：

```
x0 = [1,2];
[x,fval] = fminunc(fun,x0)
```

迭代结束后，返回：

```
x =


   -0.6691    0.0000


fval =


   -0.4052
```

6 总结

matlab已经把优化过程封装为一个函数了。我们要做的就是对问题建模，输出问题的数学模型，然后调用优化函数。

`fminunc` 的 `options` 选项还提供了更多的选择，比如是否显示迭代过程，采用的优化算法等等等等，这些可以通过查阅matlab的帮助文档悉知。