SDS 385 Exercise Set 01

Kevin Song

September 17, 2016

1 The Wolfe Conditions

The Wolfe conditions have two components: one makes sure that we have searched far enough, and the second makes sure that we do not search too far.

The condition that we search far enough is given by the first Wolfe condition, referred to by Nodecal and Wright as the *sufficient decrease* condition.

The sufficient decrease condition says that the step size taken must result in a decrease in the value of the function that would be greater than following some multiple of the derivative at that point, that is

$$f(\vec{x_0} + r\vec{s}) \leq f(\vec{x_0}) + c_1 \nabla f(\vec{x})^T \vec{s}$$

The *curvature condition* says that the interval must have a directional derivative along the search direction which is greater than some multiple of the original directional derivative along the search direction. This is used to rule out steps that are too small. Mathematically, this is expressed as:

$$\nabla f(\vec{x_0} + r\vec{s}) \geq c_2 \nabla f(\vec{x_0})^T$$

We can satisfy the first condition with an incredibly simple backtracking search that simply chooses the first step size in a geometric series that satisfies the sufficient decrease conditions. The code for this in python is provided below:

```python
import numpy as np

def backtracking_search(grad_func, obj_func, guess, searchDir):
    """
    Uses backtracking search to find a good step size.

    grad_func:  (function) calculates the gradient
    obj_func:   (function) calculates the objective value
    guess:      (cvector)  current guess (beta)
    searchDir:  (cvector)  direction to search along
    """

    ratio = 0.7 # How much do we decrease the step size each time?
    step = 1     # Initial trial value (almost certainly wrong)
    c1 = 1e 4    # Value suggested by book


    while obj_func(guess + searchDir * step) > \
          obj_func(guess) + c1 * step * np.dot(grad_func(guess).T, searchDir):

        step *= ratio

    return step
```

../code/backtrack.py

This code, even though it does not incorporate the curvature conditions, already represents a huge improvement from constant-size search. Whereas the previous search would take it's full 10000 steps without converging, with intelligent step sizes, steepest descent converges in under 300 iterations, even at a tolerance of 10^{-5}, using a ratio (upon step size failure) of 0.55.

Examination of the step sizes produced by the code shows that while most of the steps are the same order of magnitude, they vary by up to a factor of 8.