Algorithm A.10 Simple gradient ascent algorithm

```
    Procedure Gradient-Ascent (
      θ^1,    // Initial starting point
      f_obj,  // Function to be optimized
      δ       // Convergence threshold
    )
1     t ← 1
2     do
3       θ^{t+1} ← θ^t + η∇f_obj(θ^t)
4       t ← t + 1
5     while ||θ^t − θ^{t−1}|| > δ
6     return (θ^t)
```

search of algorithm A.5 (see appendix A.4.2). Using the Taylor expansion of a function, we know that, in the neighborhood of $\boldsymbol{\theta}^0$, the function can be approximated by the linear equation

$$f_{\text{obj}}(\boldsymbol{\theta}) \approx f_{\text{obj}}(\boldsymbol{\theta}^0) + (\boldsymbol{\theta} - \boldsymbol{\theta}^0)^T \nabla f_{\text{obj}}(\boldsymbol{\theta}^0).$$

Using basic properties of linear algebra, we can check that the slope of this linear function, that is, $\nabla f_{\text{obj}}(\boldsymbol{\theta}^0)$, points to the direction of the steepest ascent. This observation suggests that, if we take a step in the direction of the gradient, we increase the value of f_{obj}. This reasoning leads to the simple gradient ascent algorithm shown in algorithm A.10. Here, η is a constant that determines the *rate* of ascent at each iteration. Since the gradient ∇f_{obj} approaches 0 as we approach a maximum point, the procedure will converge if η is sufficiently small.

Note that, in order to apply gradient ascent, we need to be able to evaluate the function f_{obj} at different points, and also to evaluate its gradient. In several examples we encounter in this book, we can perform these calculations, although in some cases these are costly. Thus, a major objective is to reduce the number of points at which we evaluate f_{obj} or ∇f_{obj}.

The performance of gradient ascent depends on the choice of η. If η is too large, then the algorithm can "overshoot" the maximum in each iteration. For sufficiently small value of η, the gradient ascent algorithm will converge, but if η is too small, we will need many iterations to converge. Thus, one of the difficult points in applying this algorithm is deciding on the value of η. Indeed, in practice, one typically needs to begin with a large η, and decrease it over time; this approach leaves us with the problem of choosing an appropriate schedule for shrinking η.

A.5.2.2 Line Search

An alternative approach is to adaptively choose the step size η at each step. The intuition is that we choose a direction to climb and continue in that direction until we reach a point where we start to descend. In this procedure, at each point $\boldsymbol{\theta}^t$ in the search, we define a "line" in the direction of the gradient:

$$g(\eta) = \vec{\boldsymbol{\theta}^t} + \eta \nabla f_{\text{obj}}(\boldsymbol{\theta}^t).$$

line search

We now use a *line search* procedure to find the value of η that defines a (local) maximum of