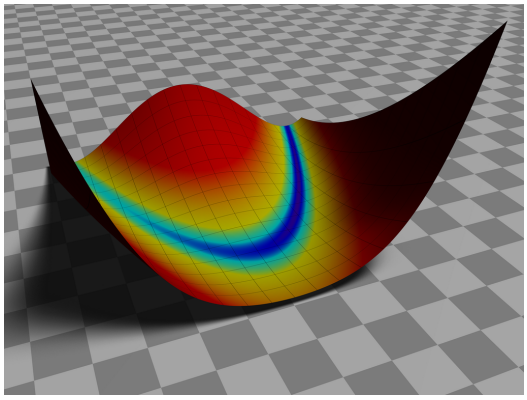
Unconstrained Optimization

Lecture 4



ME EN 575
Andrew Ning
aning@byu.edu

Outline

gradient

$$\nabla f(x) \equiv g(x) \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hessian

$$\nabla^2 f(x) \equiv H(x) \equiv \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{bmatrix}$$

A local quadratic approximation of a function (Taylor's series):

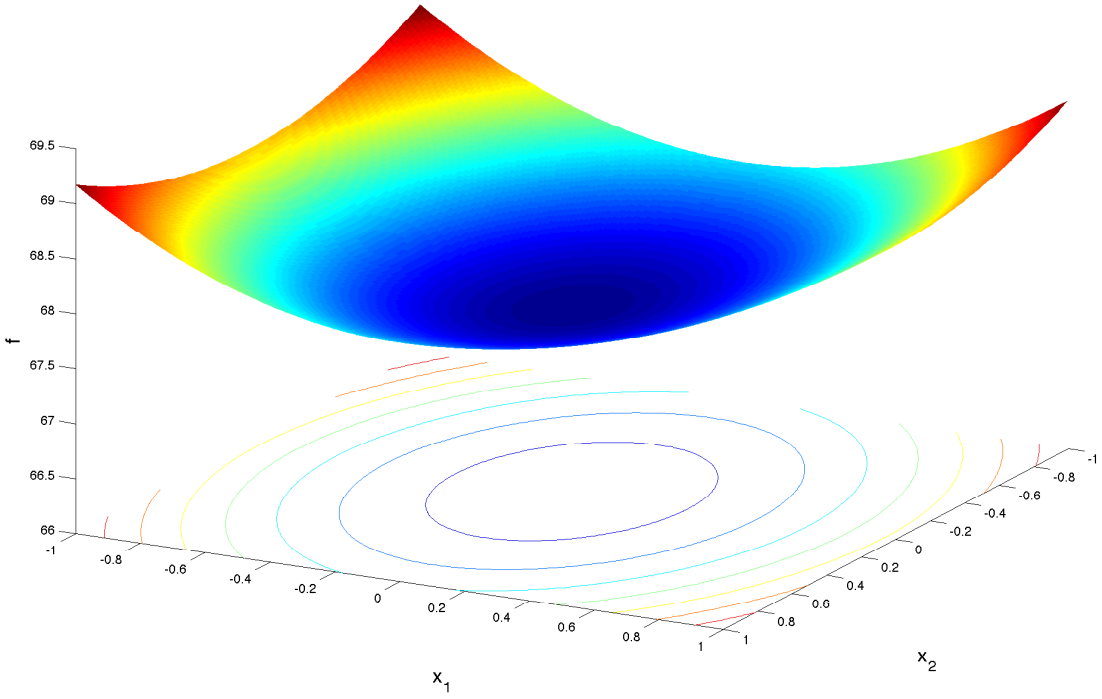
$$f(x_0 + \Delta x) = f(x_0) + g^T \Delta x + \frac{1}{2} \Delta x^T H \Delta x + \dots$$

Optimality Conditions

$$\|g(x^*)\| = 0$$

$H(x^*)$ is positive definite

Positive Definite



Mathematically:

$$x^T Ax > 0 \text{ for all } x \in \mathbb{R}^n$$

equivalently:

all the eigenvalues of A are positive

Unconstrained Optimization

1. Initial guess: x_0
2. Repeat until convergence criteria satisfied:
    3. Compute a search direction: p_k
    4. Find a step length α_k that satisfies Wolfe conditions (roughly $f(x_k + \alpha_k p_k) < f(x_k)$)
    5. Update the design variables: $x_{k+1} = x_k + \alpha_k p_k$

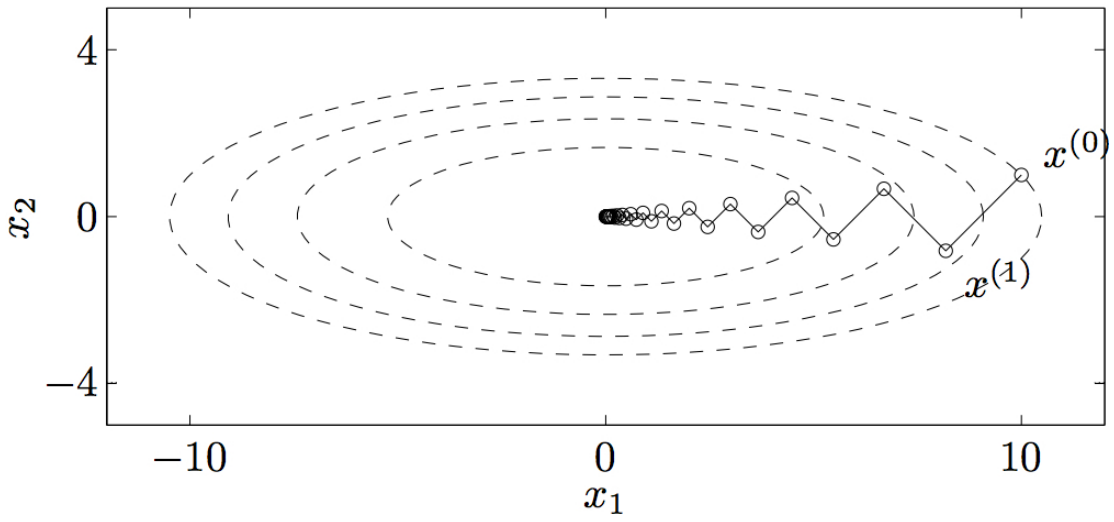
How do we choose p_k?

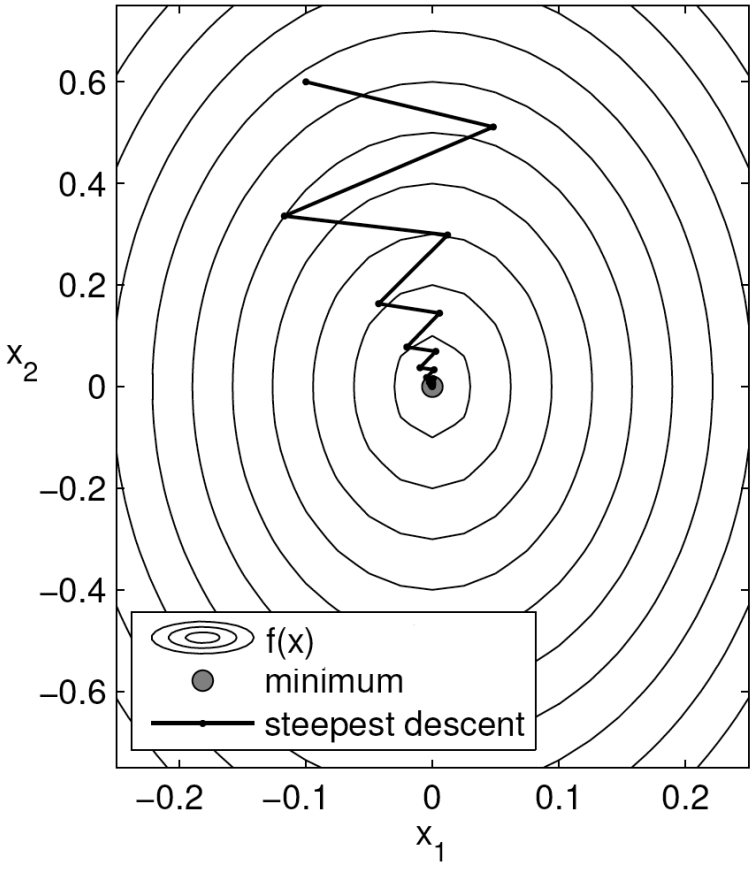
Steepest Descent

An intuitive approach, like going down a mountain. Search in the gradient direction (actually negative of gradient):

$$p_k = -g(x_k)$$

Unfortunately, this is usually a horrible approach. If we take an optimal step size, each iteration is orthogonal to the last, meaning we zip-zag towards our target.





We can prove this mathematically. If we choose the optimal step size in the line search:

$$\frac{df(x_{k+1})}{d\alpha} = 0$$

$$\frac{\partial f(x_{k+1})}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial \alpha} = 0$$

$$(\nabla f(x_{k+1}))^T p_k = 0$$

$$-g^T(x_{k+1}) g(x_k) = 0$$

Another disadvantage: this method does not naturally predict a step size, only a direction.

One option is to normalize (a unit length step):

$$p_k = \frac{-g(x_k)}{||g(x_k)||}$$

Another is to use $p_k = -g_k$, and assume the first-order change in f should be similar to that of the previous step:

$$\alpha_k g_k^T p_k = \alpha_{k-1} g_{k-1}^T p_{k-1}$$

$$\alpha_k = \alpha_{k-1} \frac{g_{k-1}^T p_{k-1}}{g_k^T p_k}$$

Jupyter example

Newton's Method

We can do better by using second-order information. Recall:

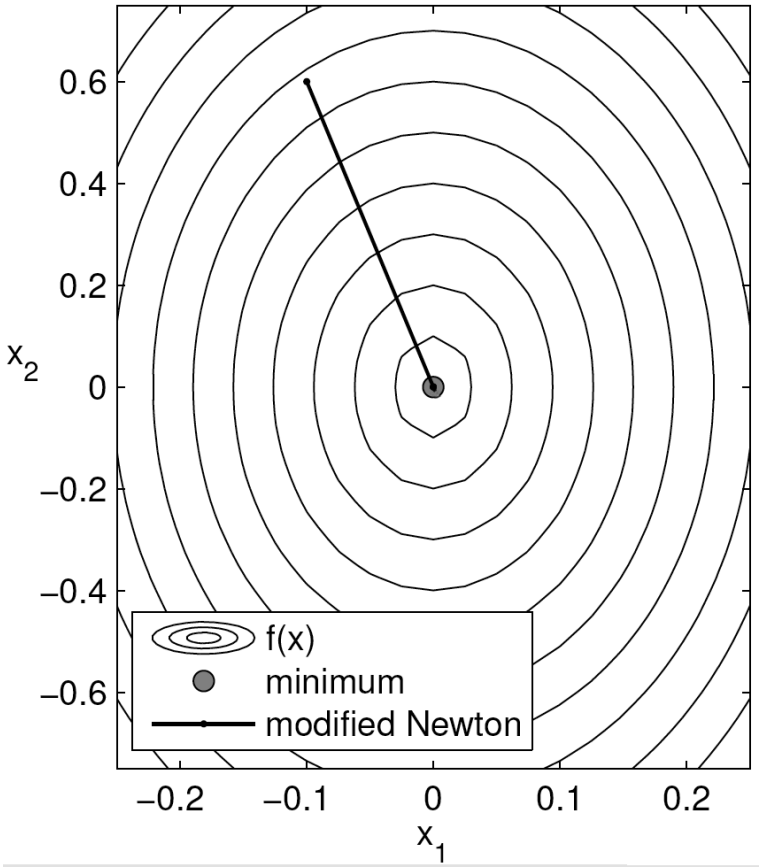
$$f(x_0 + s_k) \approx f_k + g_k^T s_k + \frac{1}{2} s_k^T H_k s_k$$

The Newton step yields the minimum of the function:

$$H_k s_k = -g_k$$

Compare to 1D approach

Advantage: if the function is quadratic, this will converge in one step.



Disadvantages:

1. The Newton step might yield a poor point (i.e., $f(x_k + s_k) > f(x_k)$)
2. The Hessian might not be positive definite (i.e., no minimum)
3. It requires the Hessian!

To deal with first problem: perform a line search along the Newton direction.

$$p_k = -H_k^{-1} g_k$$

and start with $\alpha_k = 1$

(note: one does not actually invert matrix)

To deal with the second problem: modify the Hessian.

$$B_k = H_k + \gamma I$$

The solve the third problem: we need a new method (discussed next time).