Minimizing functions of several variables.

In real life, it's extremely rare that you have a meaningful function of one variable. So we now turn our attention to a general function.

We start with a brief review of Taylor's Theorem. Let $f: \mathbb{R}^n \to \mathbb{R}$.

Then

$$\nabla f = G = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \underline{\text{gradient vector}} \text{ of } f.$$

the <u>directional derivative</u> of f in the direction $\vec{v}$ is given by

$$D_{\vec{v}} f = \frac{d}{d\epsilon} f(\vec{x} + \epsilon \vec{v}) \Big|_{\epsilon=0} = G \cdot \vec{v}$$

The <u>Hessian</u> of f is the nxn matrix

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

We observe that the Hessian is symmetric (for functions with C^0 second partials). Geometrically,

$$\vec{v} \cdot H \vec{w} = D_{\vec{v}}(D_{\vec{w}} f)$$

"directional derivative in the $\vec{v}$ dir. of the directional derivative in the $\vec{w}$ direction of f"

while

$\vec{v} \cdot H \vec{v}$ = second derivative of f in the $\vec{v}$ direction.

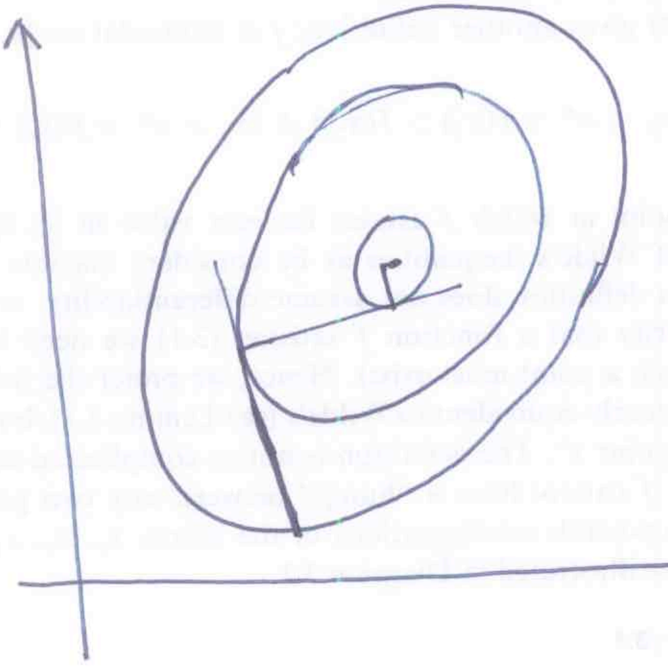
We care because

$$f(\vec{x}+\vec{h}) = f(\vec{x}) + G(\vec{x})^T \vec{h} + \frac{1}{2} \vec{h}^T H(x) \vec{h} + \ldots$$

which is Taylor's theorem in general.

We generally graph a function of two variables in terms of contour lines $f(\vec{x}) = c$. Note $G(\vec{x})$ is $\perp$ to the contour through $\vec{x}$.

Steepest descent method.



We can start by proposing that

we choose the direction

$-G(\vec{x})$ = step direction

since this direction has the smallest (most negative) directional derivative.

How far should we step? We should minimize the scalar function

$$f(\vec{x} - \lambda G(\vec{x})) \text{ over } \lambda > 0$$

to get the most from each step.

Several observations are important here.

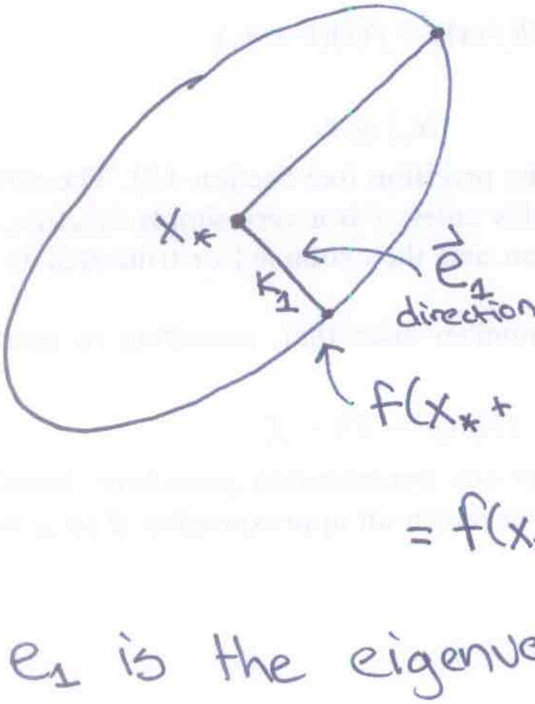
Convergence Theorem. (Luenberger, Linear and Nonlinear Programming, Chapter 7.)

If f is C^2, has a local min x_* and the Hessian $H(x_*)$ is positive definite then for x_k sufficiently close to x_*, if x_{k+1} minimizes f along the line through x_k in the $-G(x_k)$ direction,

$$f(x_{k+1}) - f(x_*) \leq \left(\frac{1-r}{1+r}\right)^2 \left(f(x_k) - f(x_*)\right)$$

where $r = \dfrac{\text{smallest eigenvalue of } H(x_k)}{\text{largest eigenvalue of } H(x_k)}$.

This theorem expresses the rate of convergence in terms of the geometry of the contours around x_*.



In general, these are ellipsoids, with axes given by eigenvectors of $H(\vec{x})$.

$$f(x_* + K\vec{e}_1) = f(x_*) + \frac{1}{2} K\vec{e}_1^{\,T} H K\vec{e}_1$$

$$= f(x_*) + \frac{1}{2} K^2 |\vec{e}_1|^2 \lambda_1$$

where e_1 is the eigenvector with eigenvalue λ_1.

Solving for the axes of the ellipse, we set $|\vec{e}_i| = 1$, and solve

$$\frac{1}{2} = \frac{1}{2} k^2 \lambda_1$$

or

$$k_1 = 1/\sqrt{\lambda_1}.$$

Therefore the ratio r of the largest and smallest eigenvalues

$$\begin{array}{l} \text{small} \rightarrow \\ \\ \text{large} \rightarrow \end{array} \frac{\lambda_i}{\lambda_j} = \left(\frac{1/\sqrt{\lambda_j}}{1/\sqrt{\lambda_i}} \right)^2 = \left(\frac{\overset{\cancel{\text{large}}\ \text{small}}{k_j}}{\underset{\cancel{\text{small}}\ \text{large}}{k_i}} \right)^2$$

is the square of the ratio of the ~~smallest~~ largest and smallest axes of the ellipse.

Conclusion. Performance is good for approximately circular contours ($r \approx 1$) and bad for long skinny ellipses ($r \approx 0$).

We next have

Convergence Theorem with inexact search. (Fletcher, Practical Methods of Optimization, Antoniou and Lu, page 110).

If $f(x_k)$ has a lower bound, $G(x)$ is uniformly continuous on $\{x : f(x) < f(x_0)\}$, the step directions d_k are not orthogonal to $-G(x_k)$, then if x_{k+1} obeys the Goldstein conditions on the line through x_k in direction d_k, the x_k converge to a point where $G(x) = 0$.

This often lets you save considerable time, since inexact search is a lot faster than something like Brent's method.

Mathematica demonstration of steepest descent with ~~and~~ Brent and with inexact line search.

Next up: conjugate directions!