Gradient descent

Newton's method is useful, but it is costly:

$$x_{n+1} = x_n - \underbrace{\nabla f(x_n)^{-1} D f(x_n)}_{\text{expensive}} . \qquad [f : \mathbb{R}^N \to \mathbb{R}]$$

If we assume each $\partial f / \partial x_i$ and $\partial^2 f / \partial x_i \partial x_j$ can be evaluated in $O(1)$ operations, then to find the n^{th} Newton step:

$$p_n := -D^2 f(x_n)^{-1} D f(x_n)$$

requires us to form and solve the linear system:

$$+ D^2 f(x_n) p_n = \nabla f(x_n)$$

for each $n = 0, 1, \ldots$. How to do this?

1) Build $D^2 f(x_n)$ — $O(n^2)$ ops & $O(n^2)$ memory
2) Build $D f(x_n)$ — $O(n)$ ops & $O(n)$ memory
3) Gaussian elimination:
   a) compute LU decomposition (or — Cholesky factorization) of $-D^2 f(x_n)$:

   $$L_n U_n = -D^2 f(x_n)$$

   costs $O(n^3)$ ops & $O(n^2)$ memory
   b) forward solve w/ L_n — $O(n^2)$ ops
   c) backward solve w/ U_n — $O(n^2)$ ops

So: need $O(n^3)$ ops and $O(n^2)$ memory! Quite expensive!

Additional problem (which is true of most methods) — the iteration may not converge!

Note: if x_0 is sufficiently close to x^*, then Newton's method converges quadratically! Roughly, this means that the number of correct digits in the solution doubles at each iteration. For a tolerance of $\varepsilon > 0$, what does this mean? Again, roughly:

$$\underbrace{-\log_{10} \varepsilon}_{\simeq \text{ \# correct digits}} \cong 2^n$$

Solve for n:

$$n = \log_2 \log_{10} \frac{1}{\varepsilon} = O\left(\log \log \frac{1}{\varepsilon}\right).$$

So, n grows extremely slowly! For example, machine epsilon of a double-precision floating-point number is $\varepsilon_{mach} \cong 2.2 \times 10^{-16}$. Hence:

$$n \cong \log_2 \log_{10} \frac{1}{2.2 \cdot 10^{-16}} \simeq 4.$$

The issue here is whether the small number of iterations justifies the large cost per iteration — also, whether we can find a good start for ~~t~~ iteration! (That is, a good choice of x_0.)

Typically, for high-dimensional problems, the cost of Newton's method is not justified.

In general, how to we come up with an iterative scheme for solving a minimization problem:

$$x_{n+1} = x_n + p_n(x_n)?$$

A good starting point is to require that:

$$f(x_{n+1}) < f(x_n), \qquad (*)$$

so that we make progress. If we Taylor expand $f(x_{n+1})$:

$$f(x_{n+1}) = f(x_n + p_n) = f(x_n) + \nabla f(x_n)^T p_n + O(\|p_n\|_2^2)$$

we see that $(*)$ — to first order — is the same as requiring:

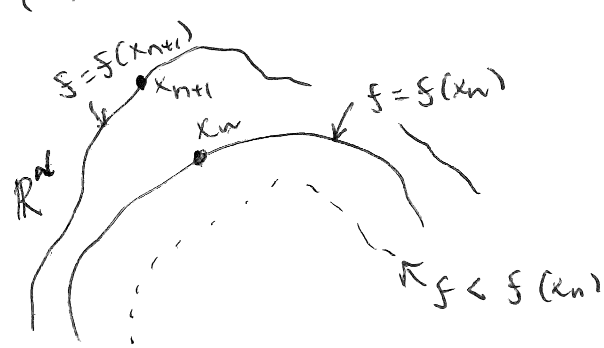
$$\boxed{\nabla f(x_n)^T p_n < 0.} \qquad (**)$$

We say that a choice of p_n which satisfies $(**)$ is a descent direction.

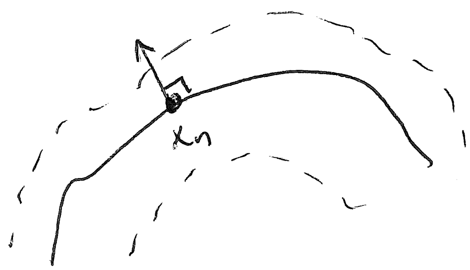
What does it mean for a vector to be a descent direction? Let's consider the $f(x_n)$-level set of f:

$$\{ x \in \mathbb{R}^N : f(x) = f(x_n) \}$$

Eg:



Recall that the gradient of f is orthogonal to the level set provided that f is C^1,



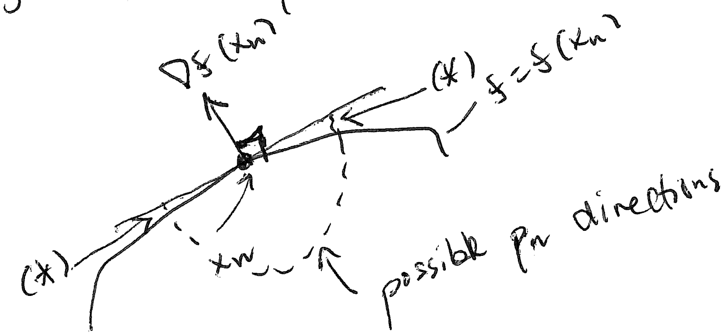
Why? Let $x : \mathbb{R} \to \mathbb{R}^n$ be a smooth parametric curve (with $x = x(t)$) s.t. $x(0) = \cancel{}\ x_n$. Then,

$$\frac{d}{dt} f(x(t))\Big|_{t=0} = \nabla f(x(t))^T \dot{x}(t)\Big|_{t=0} = \nabla f(x_n)^T \dot{x}(0).$$

($\dot{x}$: $\frac{d}{dt}$)

If we assume that $f(x(t)) \equiv f(x_n)$ for all $t \in \mathbb{R}$, then $\dot{x}(0)$ is a tangent vector of the $f(x_n)$-level set at x_n. So, we can see that $\nabla f(x_n)$ is orthogonal to all such tangent vectors, hence it is orthogonal to the level set.

So, the requirement for $\nabla f(x_n)^T p_n < 0$ to hold has the following geometric interpretation:



The $\nabla f(x_n)$ vector provides "support" for the half-space of directions which should lead to a reduction in f.

Note (from (*)) that this is not a guarentee!

A very simple choice for a descent direction is just

$$p_n = -\nabla f(x_n)$$

since:

$$\nabla f(x_n)^T p_n = -\nabla f(x_n)^T \nabla f(x_n)$$
$$= -\|\nabla f(x_n)\|_2^2 \leq 0.$$

Note: if $\nabla f(x_n) = 0$, then we have found a stationary point!

Let's examine gradient descent (or the steepest descent method) more closely:

$$x_{n+1} = x_n - \nabla f(x_n).$$

You should have seen in HW1 that you can use this iteration to reliably find a stationary point of a function, although it is dependent on the choice of x_0. Is this guaranteed to happen? Let's consider two examples where we try to solve:

$$\underset{x \in \mathbb{R}}{\text{minimize}} \quad x^2.$$

Very simple! Just have that $x^* = 0$, since $f'(x) = 2x$.

Example 1: Use the iteration $x_{n+1} = x_n - f(x_n)$.

Let $x_0 \in \mathbb{R}$ be arbitrary. Then; notice that:

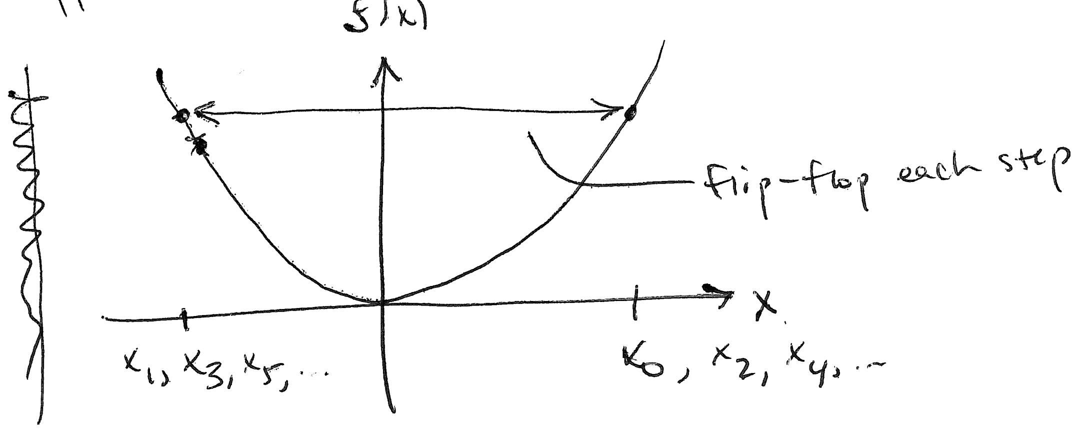
$$x_{n+1} = x_n - f'(x_n) = x_n - 2x_n = -x_n.$$

Hence:

$$x_n = (-1)^n x_0.$$

This sequence not only does not converge to $x^* = 0$, it does not converge at all!

What happened?



A simple fix is to introduce a step length $\alpha_n > 0$ s.t.:

$$x_{n+1} = x_n + \alpha_n p_n.$$

Does this necessarily fix the problem? Here it is clear that we need to choose $\alpha_n < 1$ for each $n \geq 0$.

In general, we should choose $\alpha > 0$ such that we minimize:

$$\underset{\alpha > 0}{\text{minimize}} \; f(x_n + \alpha p_n).$$

Example 2: Again, for $f(x) = x^2$, we get;

$$f(x_n + \alpha_n p_n) = (x_n + \alpha_n p_n)^2$$
$$= x_n^2 + 2\alpha_n x_n p_n + \alpha_n^2 p_n^2$$

Taking the derivative wrt α and setting the result equal to 0 gives:

$$0 = \frac{d}{d\alpha}\left\{ x_n^2 + 2\alpha_n x_n p_n + \alpha_n^2 p_n^2 \right\} = 2x_n p_n + 2\alpha_n p_n^2$$

Hence; since $p_n = -f(x_n) = -2x_n$:

$$0 = 2x_n \overset{-}{\cancel{+}} 2\alpha_n (2x_n) = 2x_n - 4\alpha_n x_n .$$

Hence: $\alpha_n = \frac{1}{2}$. And indeed:

$$x_{n+1} = x_n + \alpha_n p_n = x_n - \frac{1}{2} \cdot 2 \cdot x_n = 0 = x^* .$$

This means that we converge in one step, but to find the optimal step size, we essentially had to solve the original problem analytically. Since finding the optimal step length can be so expensive, we will usually find an approximate step size instead.

To drive this point home, consider using Newton's method to minimize the function:

$$g(\alpha_n) = f(x_n + \alpha_n p_n) .$$

We have:

$$\alpha_{n+1} = \alpha_n - \frac{g'(\alpha_n)}{g''(\alpha_n)}.$$

But:

$$g'(\alpha_n) = \nabla f(x_n + \alpha p_n)^T p_n$$

$$g''(\alpha_n) = p_n^T \nabla^2 f(x_n + \alpha p_n) p_n$$

So that:

$$\alpha_{n+1} = \alpha_n - \frac{\nabla f(x_n + \alpha p_n)^T p_n}{p_n^T \nabla^2 f(x_n + \alpha p_n) p_n}.$$

The cost of this iteration is already quite expensive. And note that the foregoing discussion about choosing a step size to encourage consistent reduction in the function value applies here, too!

We will discuss methods for so-called "inexact line search" next time.