Gradient Descent

CS 111
Winter 2023

When n gets big enough, even using sparse methods or approximations for the $n \times n$ Hessian matrix H. Using only the gradient vector ∇f (first derivatives), we can do

GRADIENT DESCENT:

$$x_{k+1} = x_k - s_k \nabla f(x_k)$$

gradient: downhill

stepsize (learning rate)

previous guess

next guess

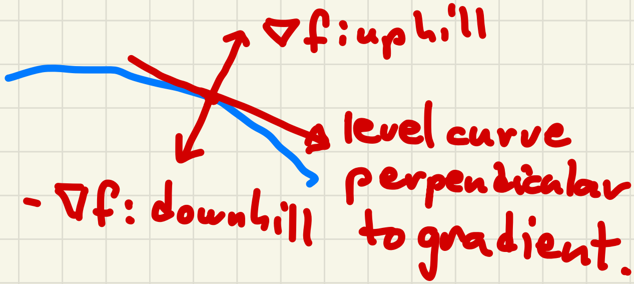
HOW TO CHOOSE STEP SIZE s_k?

1. 1-dimensional line search for lowest point
2. Use a fixed value chosen by experiment.
3. More sophisticated adjustments depending on k and direction.

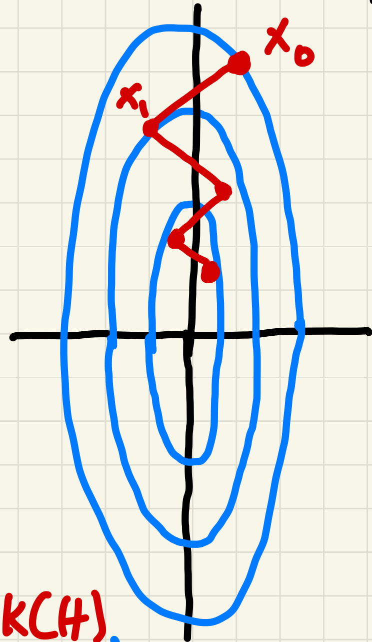
Probably the most common approach is #2, trial + error.

$$x_{k+1} = x_k - s_k \nabla f(x_k)$$

Recall: ∇f is perpendicular to the level curve.



Causes the path to zigzag:



Convergence is controlled by $\kappa(H)$, the condition # of the Hessian, even though you don't use H.

$$|f(x_{k+1}) - f(x^*)| \leq \left(1 - \frac{1}{\kappa(H)}\right) |f(x_k) - f(x^*)|$$

This is LINEAR convergence, not QUADRATIC (Newton).

For the Strang function, $H = \begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix}$, $\kappa(H) = \frac{1}{\alpha}$.

SEE JUPYTER EXAMPLES OF STRANG + ROSENBROCK.

GRADIENT DESCENT WITH MOMENTUM

Think of a heavy ball rolling downhill.
Instead of pure direction $d_k = -\nabla f(x_k)$, momentum mixes in some of d_{k-1}:

$x_0 = \text{guess}$; $d_{-1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

for $k = 0, 1, 2, \ldots$

$$d_k = -\nabla f(x_k) + \beta d_{k-1}$$

$$x_{k+1} = x_k + s d_k$$

β: momentum coefficient

s: step size

Roughly speaking, this replaces $\kappa(H)$ with $\sqrt{\kappa(H)}$.

If $\kappa(H) = 100$, $\left(1 - \frac{1}{\kappa(H)}\right) = .99$

but $\left(1 - \frac{1}{\sqrt{\kappa(H)}}\right) = .90$

This is huge! But it gives another parameter to tune.

See Jupyter examples with Strang + Rosenbrock.

NOTE: Much earlier in the course we saw an idea somewhat similar to momentum in conjugate gradient, which solves $Ax=b$ by $x^* = \operatorname{argmin}\left(\frac{1}{2}x^T A x - x^T b\right)$

Here $\nabla f(x) = Ax - b = -\text{residual}$, so r_k in our CG code is just $-\nabla f$.

... and the CG iterative step uses

$$d_k = -\nabla f(x_k) + \beta_k d_{k-1},$$

except that CG computes a new (optimal) β_k at each step.

NOT THE SAME AS MOMENTUM, BUT CURIOUSLY SIMILAR!

NESTEROV'S METHOD

Another way to accelerate gradient descent by using the previous direction d_{k-1}:

Less intuitive than momentum but sometimes more stable.

$$x_0 = \text{guess}\ ;\ d_{-1} = 0$$

for $k = 0, 1, 2, \ldots$

$$d_k = \beta d_{k-1} - \nabla f(x_k + \gamma d_{k-1})$$

$$x_{k+1} = x_k + s d_k$$

"lookahead gradient"

One additional parameter γ.