2nd Order Gradient Methods: The Role of the Hessian

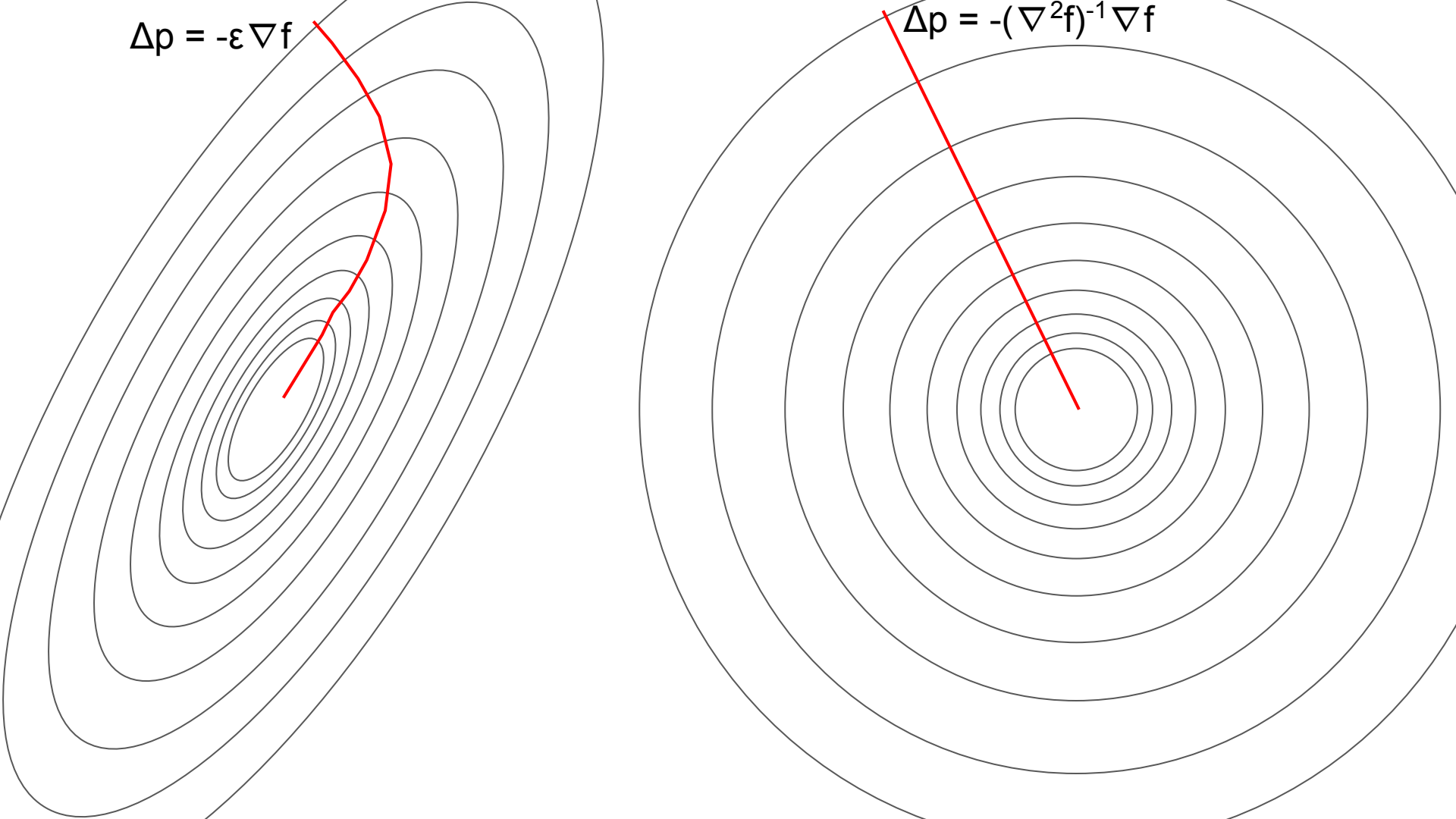
Suppose the cost surface is locally quadratic?

$\text{Cost} = x^T Ax/2 + Bx + C$

Minimized or maximized when the derivative is zero.

$\nabla \text{Cost} = Ax + B$
$\nabla \text{Cost} = 0$ when $x = -A^{-1}B$

Taylor series:
$\text{Cost} = x^T(\nabla^2 f)x/2 + (\nabla f)x + f(x_0)$ where x is the position relative to x_0


$\Delta p = -\varepsilon \nabla f$
$\Delta p = -(\nabla^2 f)^{-1} \nabla f$

When to use which?

In theory a first order gradient step always works, so it is a good backup.

First order gradient descent is slow in valleys (the derivative you care about is small relative to derivatives you don't care about), and is attracted to valleys. The first few steps of first order gradient descent are great, and then things get sloooow.

Approximate Hessians are bad when they have small eigenvalues (H^{-1} is large in those directions).

Saddle points (some eigenvalues are negative) need special handling.

Levenberg Marquardt

Update: $\Delta p = -(\lambda I + \nabla^2 f)^{-1} \nabla f$

λ can be set large enough to make the matrix that is inverted positive definite.

λ large makes this update approximate $\Delta p = -\varepsilon \nabla f$ (a first order gradient update) with $\varepsilon = 1/\lambda$.

λ small makes this update approximate $\Delta p = -(\nabla^2 f)^{-1} \nabla f$ (a second order update)

An adaptive step size algorithm (Levenberg Marquart):

Good step: decrease λ ($\lambda = 0.9*\lambda$, cautiously move to 2nd order method)

Bad step: increase λ ($\lambda = 2*\lambda$, quickly back off to 1st order method).

Trust Regions