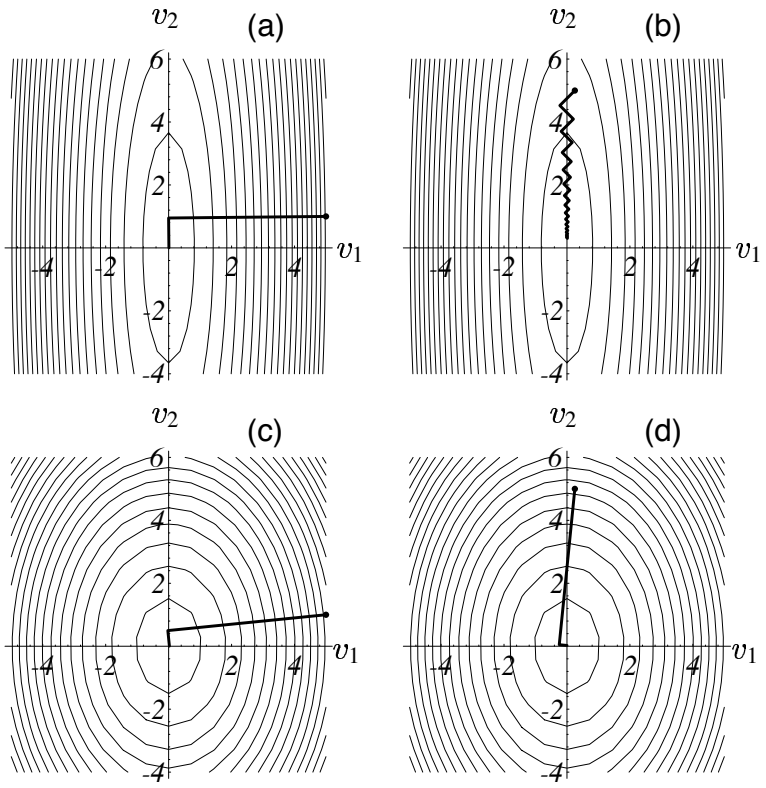
Convergence of gradient descent

The effectiveness of gradient descent depends critically on the "conditioning" of $\boldsymbol{H}$ and the starting point. Consider the two examples below:



(from Shewchuk, "... without the agonizing pain")

When the conditioning of $\boldsymbol{H}$ is poor, which here corresponds to the case where the ellipses denoting the level sets of our objective function are more eccentric or "squished", and we choose a bad starting point, convergence can take many iterations even in simple cases.

We can make this a bit more precise if we define mathematically what we really mean by the conditioning of $\boldsymbol{H}$. The **condition number** of a matrix $\boldsymbol{H}$, typically denoted $\kappa(\boldsymbol{H})$ is the ratio of the largest to smallest singular values of $\boldsymbol{H}$:

$$\kappa(\boldsymbol{H}) = \frac{\sigma_{\max}(\boldsymbol{H})}{\sigma_{\min}(\boldsymbol{H})}.$$

Note that by the $\sigma_{\max}(\boldsymbol{H})$ we mean the largest singular value and by $\sigma_{\min}(\boldsymbol{H})$ we mean the smallest *non-zero* singular value, i.e., σ_R where R is the rank of $\boldsymbol{H}$. For the case where $\boldsymbol{H}$ is a square matrix (as it is in our context), we can also equivalently write

$$\kappa(\boldsymbol{H}) = \frac{\lambda_{\max}(\boldsymbol{H})}{\lambda_{\min}(\boldsymbol{H})},$$

where $\lambda_{\max}(\boldsymbol{H})$ and $\lambda_{\min}(\boldsymbol{H})$ denote the largest and smallest eigenvalues of $\boldsymbol{H}$, respectively. The condition number is a natural way of quantifying just how sensitive we are going to be to noise, but it also plays a key role in determining how computationally challenging it will be to solve the least squares problem using iterative methods.

Specifically, below we will provide a bound that shows how $f(\boldsymbol{x}_k)$ approaches $f(\boldsymbol{x}^\star)$, where $\boldsymbol{x}^\star$ denotes the minimizer of f. Specifically, we will show that

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}^\star) \leq \left(1 - \frac{1}{\kappa(\boldsymbol{H})}\right)\left(f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\right). \tag{1}$$

Let's think a bit about what this says. Note that $1 - 1/\kappa(\boldsymbol{H})$ is always less than 1, so each iteration makes *some* progress. If $\kappa(\boldsymbol{H}) \leq 2$, then at each iteration we make *a lot* of progress – cutting the error in half with each iteration. However, if $\kappa(\boldsymbol{H})$ is very large, this constant becomes very close to 1, indicating only minor improvements.

Convergence analysis

Recall that we are trying to minimize

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b}.$$

Our convergence analysis will rely on one very useful property of $f(\boldsymbol{x})$, namely that we can write[1]

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\nabla f(\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{y}-\boldsymbol{x}). \tag{2}$$

We can easily verify this by just plugging in $\nabla f(\boldsymbol{x}) = \boldsymbol{H}\boldsymbol{x} - \boldsymbol{b}$ and simplifying. Specifically, note that we can equivalently write (2) as

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) = (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\nabla f(\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{y}-\boldsymbol{x}).$$

The right-hand side of this equation can be simplified as

$$\begin{aligned}
&(\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\nabla f(\boldsymbol{x}) + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{y}-\boldsymbol{x}) \\
&= (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{H}\boldsymbol{x}-\boldsymbol{b}) + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{y}-\boldsymbol{x}) \\
&= \boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{b} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b} + \frac{1}{2}\left(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} - 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{y}\right) \\
&= \frac{1}{2}\boldsymbol{y}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}^{\mathrm{T}}\boldsymbol{b} - \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b} \\
&= f(\boldsymbol{y}) - f(\boldsymbol{x}),
\end{aligned}$$

as desired.

[1] This is like taking a second-order Taylor approximation to f around the point $\boldsymbol{x}$, but since f is a quadratic function this is not an approximation but exact.

Equation (2) immediately tells us something about how much progress we make at each iteration. If we plug in $\boldsymbol{y} = \boldsymbol{x}_{k+1}$ and $\boldsymbol{x} = \boldsymbol{x}_k$ to (2), in which case $\boldsymbol{y} - \boldsymbol{x} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k = -\alpha_k \nabla f(\boldsymbol{x}_k)$, we obtain

$$f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k) - \alpha_k \|\nabla f(\boldsymbol{x})\|_2^2 + \frac{\alpha_k^2}{2} (\nabla f(\boldsymbol{x}_k))^{\mathrm{T}} \boldsymbol{H} \nabla f(\boldsymbol{x}_k). \quad (3)$$

Recall that we had set

$$\alpha_k = \frac{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{r}_k}{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{H} \boldsymbol{r}_k},$$

where $\boldsymbol{r}_k = -\nabla f(\boldsymbol{x}_k)$. Plugging this into (3) yields

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &= f(\boldsymbol{x}_k) - \frac{\|\boldsymbol{r}_k\|_2^4}{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{H} \boldsymbol{r}_k} + \frac{1}{2} \left(\frac{\|\boldsymbol{r}_k\|_2^2}{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{H} \boldsymbol{r}_k} \right)^2 \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{H} \boldsymbol{r}_k \\ &= f(\boldsymbol{x}_k) - \frac{1}{2} \frac{\|\boldsymbol{r}_k\|_2^4}{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{H} \boldsymbol{r}_k}. \end{aligned}$$

This tells us that we are guaranteed to make at least *some* progress at each iteration. Precisely how much depends on this rather strange looking function of $\boldsymbol{r}_k$, but we can actually get a much simpler expression by recalling that for any symmetric, positive semidefinite matrix $\boldsymbol{H}$ we have that

$$\lambda_{\min}(\boldsymbol{H}) \leq \frac{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{H} \boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}} \leq \lambda_{\max}(\boldsymbol{H}) \quad (4)$$

for all $\boldsymbol{x}$, where $\lambda_{\max}(\boldsymbol{H})$ and $\lambda_{\min}(\boldsymbol{H})$ denote the largest and smallest eigenvalues of $\boldsymbol{H}$, respectively. This is a fact that we essentially proved in the discussion of least squares in noise, although we did not explicitly state this at the time. Using the upper half of (4) we can get a simpler bound on how much progress we make at each iteration:

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_k) - \frac{1}{2\lambda_{\max}(\boldsymbol{H})} \|\boldsymbol{r}_k\|_2^2. \quad (5)$$

This bound is nice, but it would be even better if we could say something concrete on how large $\|\boldsymbol{r}_k\|_2^2$ will be. In particular, our intuition should be that if we are far from the solution, the gradient (or $\boldsymbol{r}_k$) should be large. There is a clever way to prove exactly this. First, note that (4) applied to (2) also yields

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + (\boldsymbol{y}-\boldsymbol{x})^{\mathrm{T}}\nabla f(\boldsymbol{x}) + \frac{\lambda_{\min}(\boldsymbol{H})}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2^2$$

We can obtain a simpler lower bound for $f(\boldsymbol{y})$ by determining the smallest value that the right-hand side of this could ever take over all possible choices of $\boldsymbol{y}$ To do this, we simply minimize this lower bound by taking the gradient with respect to $\boldsymbol{y}$ and setting it equal to zero:

$$\nabla f(\boldsymbol{x}) + \lambda_{\min}(\boldsymbol{H})(\boldsymbol{y}-\boldsymbol{x}) = 0,$$

From this we obtain that the lower bound will be minimized by

$$\boldsymbol{y}-\boldsymbol{x} = -\frac{1}{\lambda_{\min}(\boldsymbol{H})}\nabla f(\boldsymbol{x}).$$

Plugging this in yields

$$\begin{aligned} f(\boldsymbol{y}) &\geq f(\boldsymbol{x}) - \frac{1}{\lambda_{\min}(\boldsymbol{H})}\|\nabla f(\boldsymbol{x})\|_2^2 + \frac{1}{2\lambda_{\min}(\boldsymbol{H})}\|\nabla f(\boldsymbol{x})\|_2^2 \\ &= f(\boldsymbol{x}) - \frac{1}{2\lambda_{\min}(\boldsymbol{H})}\|\nabla f(\boldsymbol{x})\|_2^2. \end{aligned}$$

In particular, this applies when $\boldsymbol{y} = \boldsymbol{x}^\star$ (where $\boldsymbol{x}^\star$ denotes the minimizer of $f(\boldsymbol{x})$), which after some rearranging yields

$$\|\nabla f(\boldsymbol{x})\|_2^2 \geq 2\lambda_{\min}(\boldsymbol{H})\left(f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)\right). \tag{PL}$$

This is a famous and useful result, often referred to as the **Polyak-Łojasiewicz inequality**.

From our previous bound in (5) we have that

$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star) - \frac{1}{2\lambda_{\max}(\boldsymbol{H})}\|\boldsymbol{r}_k\|_2^2.$$

Combining this with the PL inequality we obtain

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}^\star) &\leq f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star) - \frac{\lambda_{\min}(\boldsymbol{H})}{\lambda_{\max}(\boldsymbol{H})}\left(f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\right) \\ &= \left(1 - \frac{\lambda_{\min}(\boldsymbol{H})}{\lambda_{\max}(\boldsymbol{H})}\right)\left(f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star)\right). \end{aligned}$$

That is, the gap between the current value of the objective function and the optimal value is cut down by a factor of $1 - 1/\kappa(\boldsymbol{H}) < 1$ at each iteration.