Topics:

- Unconstrained minimization
- Convex functions
  - local versus global optimum
- Gradient descent
- Ill-conditioning and Newton's Method

This lecture is based on Ch. 12 in ROB101 textbook by Jesse Grizzle.

A Brief Introduction to Optimization

There were a few times during the semester where we tried to find the "best" vector (or matrix) among a collection of many such vectors or matrices. For example, in least squares, we looked to find the vector x that minimized the expression $\|Ax-b\|^2$. In low-rank approximation, we looked to find the matrix $\hat{M}$ that has rank k (or less) that minimized the expression $\|\hat{M}-M\|_F^2$.

These were both specific instances of what is called a mathematical optimization problem. We will focus on unconstrained problems. You will learn a lot more about optimization problems in ESE 3040, and today is only meant to give you a small taste, and to show how essential linear algebra is in finding solutions.

Optimization is the process of finding one (or more) vectors $x \in \mathbb{R}^n$ that minimize a function $f: \mathbb{R}^n \to \mathbb{R}$. This is written as the optimization problem

$$\text{minimize } f(x) \qquad (P)$$

over $x \in \mathbb{R}^n$. In (P), the variable $x \in \mathbb{R}^n$ is called the decision variable, and the function $f: \mathbb{R}^n \to \mathbb{R}$ is called the cost function or objective function. Optimization problem (P) is called unconstrained because we are free to pick any $x \in \mathbb{R}^n$ we like to minimize $f(x)$. A constrained optimization problem has the additional requirement that x must satisfy some added conditions, e.g., live in the solution set of $Ax = b$. We will not consider such problems today, but you'll see many in ESE 3040.

The goal of optimization is to find a special decision variable x^* for which the cost function f is as small as possible, i.e., such that

$$f(x^*) \le f(x) \quad \text{for all } x \in \mathbb{R}^n. \qquad (*)$$

Such an x^* is called an optimal solution to problem (P), and is defined as the argmin of f:

$$x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x). \qquad (AM)$$

Equation (AM) simply says in math that x^* satisfies the definition (*) of an optimal point. [Note that if there are multiple optimal points, one instead writes

$$x^* \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} f(x)$$

to indicate x^* belongs to the set of optimal points. footnote]

Example: The least-squares problem

$$\text{minimize } \|A\underline{x} - \underline{b}\|^2$$

over $\underline{x} \in \mathbb{R}^n$ is an unconstrained optimization problem. The objective function is $f(\underline{x}) = \|A\underline{x} - \underline{b}\|^2$, and the optimal solution is

$$\underline{x}^* = (A^TA)^{-1}A^T\underline{b} = A^+\underline{b}$$

when $(A^TA)^{-1}$ exists. Otherwise, $\underline{x}^* \in \operatorname{argmin} \|A\underline{x} - \underline{b}\|^2 \iff A^TA\underline{x}^* = A^T\underline{b}$, i.e., if and only if $\underline{x}^*$ satisfies the normal equations $A^TA\underline{x} = A^T\underline{b}$.

Despite how simple and innocuous problem (P) looks, it can be used to encode very rich and very challenging problems. Even for $x \in \mathbb{R}$, we can get ourselves into trouble. Consider the following two functions that we wish to minimize.

Which do you think is "easier" to optimize. The left figure, with function $f_1(x)$, is "bowl-shaped", and is smallest at $x^* = 2$. What's "nice" about f_1 is that there's also an obvious algorithm for finding $x^* = 2$. If you imagine yourself as an ant standing on the function $f_1(x)$, all you need to do is "walk downhill" until I eventually find the bottom of the bowl.

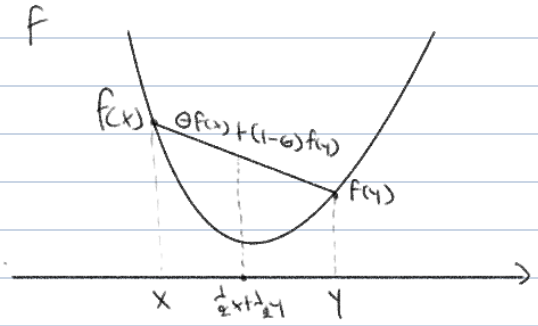
In contrast, the right figure with function $f_2(x)$, there are many hills and valleys. The optimal value $x^* = 3$ is the one for which $f_2(x)$ is smallest. But now if we again imagine ourselves as an ant standing on the function $f_2(x)$, our strategy of walking downhill will not always work! For example, if we were to start at $x = 1.5$, then walking downhill would bring us to the bottom of the first valley at $x = 1$. Now $x = 1$ is not an optimal point, since $f_2(3) < f_2(1)$, but if we look around at nearby points, then $f(1)$ is indeed the smallest. We call such a point $\tilde{x}$ that satisfies $f(\tilde{x}) \le f(x)$ for all x close to $\tilde{x}$, i.e., for $|x - \tilde{x}| \le \varepsilon$ for some $\varepsilon > 0$, a local minimum, and when we wish to emphasize that a point x^* satisfying (*) is indeed the best possible choice, we call it a global minimum.

As you may have guessed, we really like "bowl-shaped" functions for which our walking downhill strategy finds a global minimum. Such functions satisfy a geometric property called convexity. A convex function $f: \mathbb{R}^n \to \mathbb{R}$ is one which satisfies

The following property:

$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y) \quad \text{for all } \theta \in [0,1] \quad (CVX)$$
$$\text{and all } x, y \in \mathbb{R}^n$$

To understand what (CVX) is saying, it is best to draw what it means for a scalar function $f: \mathbb{R} \to \mathbb{R}$.



(CVX) says that if I pick any two points $f(x)$ and $f(y)$ on the graph, and draw a line segment between these two points, then this line lies above the graph. It turns out that this is exactly the right way to mathematically characterize "bowl-shaped" functions, even when $x \in \mathbb{R}^n$. The important feature of convex functions is that "walking downhill" will always bring us to a global minimum. We won't say much more about convex functions, but you'll see them again in ESE 3040, and there is a graduate level course, ESE 6050, which focuses entirely on convex optimization problems.

Example The affine function $f(x) = Ax - b$ is convex. To see this, we check that

$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y) \text{ for all } x, y \in \mathbb{R}^n \text{ and } \theta \in [0,1]$$

But $f(\theta x + (1-\theta)y) = A(\theta x + (1-\theta)y - b = \theta(Ax - b) + (1-\theta)(Ay - b) = \theta f(x) + (1-\theta) f(y)$.
Affine functions are on the "boundary" of being convex.

Example: The least squares objective $f(x) = \|Ax - b\|^2$ is convex. One can check this from the definition (CVX), but this is very tedious.

To gain some intuition, let's consider the scalar setting $f(x) = \|ax - b\|^2$. Expanding out $f(x)$ we see

$$f(x) = x^2 \|a\|^2 - 2abx + b^2,$$

which is an upward pointing quadratic since $\|a\|^2 > 0$ for any $a \neq 0$. This same intuition extends to $x \in \mathbb{R}^n$ setting. Expanding out $f(x) = \|Ax - b\|^2$, we get

$$f(x) = x^T A^T A x - 2 x^T A^T b + \|b\|^2.$$

This is a quadratic function, with quadratic term given by the quadratic form $x^T A^T A x$, defined by the positive semidefinite matrix $A^T A$. This means that $f(x)$ is an upward pointing bowl with ellipsoidal level sets (recall from Lecture 16), and hence is convex.

A more formal way of making this argument is to show that the Hessian $\nabla^2 f(x)$ of f is positive semi definite. Here $\nabla^2 f(x) = 2A^TA$, which is indeed positive semi definite. This is the matrix/vector equivalent of saying that $f(x) = ax^2 + bx + c$ is an upward pointing bowl if and only if $a \geq 0$.

If you don't remember what the Hessian $\nabla^2 f(x)$ of a function is, or how we computed $\nabla^2 f(x) = 2A^TA$ in the above, don't worry, as we'll review this in a bit.

Which Way is Down?

Let's assume that we are either in a "nice" setting where our objective function is convex (bowl shaped), or that we're happy to settle for a local minimum. How can we figure out which way is down so that we can tell our little ant friend which way they should walk. To get some intuition, we'll start with functions with $x \in \mathbb{R}^2$, and look at some cost function contour plots.

Let's start with two familiar examples:

- $f(x) = \|x\|^2 = x_1^2 + x_2^2$, which is nothing but the Euclidean norm squared of x, and
- $f(x) = \|x - b\|^2 = (x_1 - b_1)^2 + (x_2 - b_2)^2$, which is a very simple least squares objective with $A = I$.

We have plotted both of these functions, and their contour plots, below:

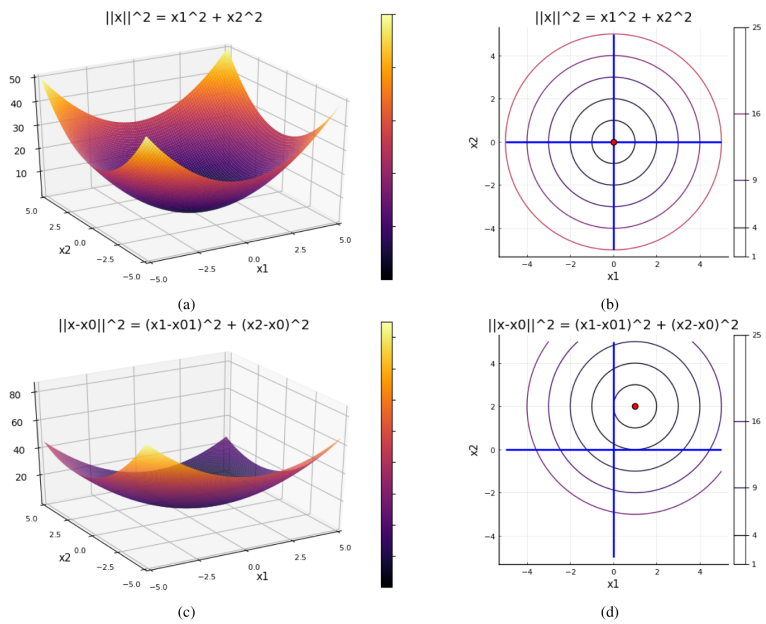


Figure 12.3: The graphs of two functions are shown in (a) and (c), with their corresponding contour plots shown in (b) and (d). Contour plots are lines where a function is constant. If you have ever used a topographical map when hiking, the lines on those maps indicate constant terrain height. In our case, the lines are constant $z = f(x_1, x_2)$ values as (x_1, x_2) vary. Because the cost function we use here is very simple, its lines of constant contour are circles. More "interesting" examples will follow.

The contour plots show the level sets of $f(x)$, which we've seen before. These are the sets $C_\alpha = \{x \in \mathbb{R}^2 \mid f(x) = \alpha\}$ for some constant α. Here, these end up being circles, since for example C_4 is the set of x_1, x_2 such that $x_1^2 + x_2^2 = 4$ or $(x_1 - b_1)^2 + (x_2 - b_2)^2 = 4$.

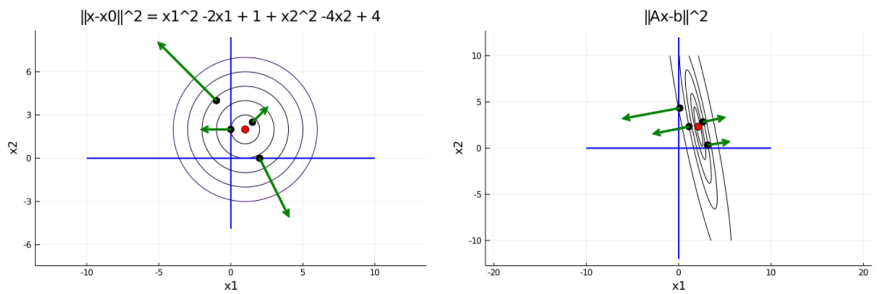
Can we use these contour plots to identify which way is downhill if our ant is currently sitting at point $\underline{x} \in \mathbb{R}^2$?

To answer this question, we'll need the gradient $\nabla f(\underline{x})$ of our function. Recall from Math 1410 that for a function $f: \mathbb{R}^n \to \mathbb{R}$, its gradient $\nabla f(\underline{x})$ is an n-vector of the partial derivatives of f:

$$\nabla f(\underline{x}) = \begin{bmatrix} \partial f/\partial x_1 (\underline{x}) \\ \vdots \\ \partial f/\partial x_n (\underline{x}) \end{bmatrix} \in \mathbb{R}^n.$$

For our functions on $\mathbb{R}^2$, this reduces to $\nabla f(\underline{x}) = \left(\frac{\partial f}{\partial x_1}(x_1, x_2), \frac{\partial f}{\partial x_2}(x_1, x_2) \right) \in \mathbb{R}^2$.

In the figure below, we show contour plots for $f_1(\underline{x}) = \|\underline{x} - \underline{b}\|^2 = (x_1 - 1)^2 + (x_2 - 2)^2$ and $f_2(\underline{x}) = \|A\underline{x} - \underline{b}\|^2$, along with the gradients $\nabla f(\underline{x})$ (green arrows) at various points.



(a) (b)

Figure 12.4: Contour Plots plus Gradients. (a) shows a very simple cost function, $||x - [1; 2]||^2$, where the contours of constant cost are circles. (b) shows a more typical situation, where the contours are squeezed in one direction. In both cases, the gradients, plotted in green arrows, are pointing in the direction of maximum increase of the function. Hence, the negative of the gradient is the direction of maximum decrease.

In both cases, the gradients are pointing in the direction of maximal increase: if we go in the opposite $-\nabla f(\underline{x})$ direction, we will move in the direction of maximal decrease!

Let's flex our calculus muscles a bit and compute the gradients of $f_1(\underline{x})$ and $f_2(\underline{x})$:

$$\nabla f_1(\underline{x}) = \begin{bmatrix} \partial/\partial x_1 \left((x_1 - 1)^2 + (x_2 - 2)^2 \right) \\ \partial/\partial x_2 \left((x_1 - 1)^2 + (x_2 - 2)^2 \right) \end{bmatrix} = 2 \begin{bmatrix} (x_1 - 1) \\ (x_2 - 2) \end{bmatrix}$$

and

$$\nabla f_2(\underline{x}) = \nabla (A\underline{x} - \underline{b})^T (A\underline{x} - \underline{b}) = 2 A^T (A\underline{x} - \underline{b})$$

[If you don't remember how to compute this gradient, you may wish to review Multilin notes, but don't worry, we won't ask you to calculate gradients on the homework/exams]

One thing you might notice in the plots above is that the red dots, which we placed on the function minimum, have no green arrows. This is because the gradient at these points is zero! In fact, this is true in general: a point $\underline{x}$ is a local minimum only if $\nabla f(\underline{x}) = 0$.

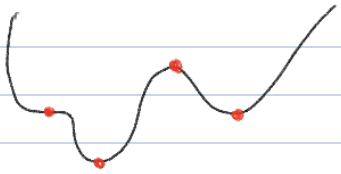
We won't prove this, but intuitively, this makes sense: if $\nabla f(\underline{x}) \neq \underline{0}$, then this means

there is a direction $-\nabla f(\underline{x})$ in which I could walk a little bit downhill to decrease $f(\underline{x})$, so $\nabla f(\underline{x})$ must be zero if we're at a local minima. Let's check that this holds for $\nabla f_1(\underline{x})$ and $\nabla f_2(\underline{x})$ above:

For $f_1(\underline{x})$, $\underline{x}^* = (1,2)$, and $\nabla f_1(\underline{x}^*) = 2\begin{bmatrix} x_1^* - 1 \\ x_2^* - 2 \end{bmatrix} = \underline{0}$, and so that checks out. And for $f_2(\underline{x})$, $\underline{x}^* = (A^TA)^{-1}A^T\underline{b}$, and

$$\nabla f_2(\underline{x}^*) = 2(A^TA\underline{x}^* - A^T\underline{b}) = 2(\underbrace{(A^TA)(A^TA)^{-1}}_{=I}A^T\underline{b} - A^T\underline{b}) = 2(A^T\underline{b} - A^T\underline{b}) = \underline{0}!$$

In general, while $\nabla f(\underline{x}^*) = \underline{0}$ is necessary for $\underline{x}^*$ to be a local minima, it is not sufficient, i.e., there may be certain points with $\nabla f(\underline{x}^*) = 0$ that are not local minima. For example, all red dots in the plot below have vanishing gradients, but only two are minima:



However, if our function f is convex (bowl shaped), we have the following theorem which tells us that walking downhill will always find a global minima:

Theorem: Let $f: \mathbb{R}^n \to \mathbb{R}$ be convex. Then $\underline{x}^*$ is globally optimal, i.e.,

$$f(\underline{x}^*) \leq f(\underline{x}) \quad \forall \underline{x} \in \mathbb{R}^n$$

if and only if $\nabla f(\underline{x}^*) = \underline{0}$.

Next, we'll use our new found insights to define gradient descent, a widely used iterative algorithm for finding local minima of optimization problem (P).

Gradient Descent

Our intuition so far is that we should try to "walk downhill" and that the negative gradient of the cost function $-\nabla f(\underline{x})$ tells us the steepest direction of descent at point $\underline{x}$. Can we turn this into an algorithm for minimizing (at least locally) a cost function $f(\underline{x})$?

This intuition is precisely the motivation behind the **gradient descent algorithm**, which starting with an initial guess $\underline{x}^{(0)}$, iteratively updates the current best guess $\underline{x}^{(k)}$ of $\underline{x}^*$ according to:

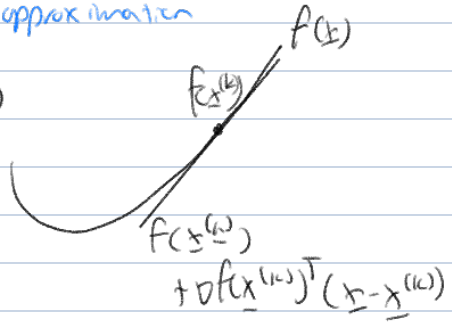
$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - s\nabla f(\underline{x}^{(k)}), \quad \text{for } k=0,1,2,\ldots \qquad (GD)$$

where $s>0$ is called the **step size**. The update rule (GD) moves you in the direction of a local minimum if $s>0$, but be careful, because if s is too large, you can overshoot (we'll see more about this later).

Because we know that $\nabla f(\underline{x}^*)=\underline{0}$ if $\underline{x}^*$ is a local minimum, we can use the norm of the gradient as a stopping criterion, i.e., if $\|\nabla f(\underline{x}^{(k)})\| \le \varepsilon$, for some small $\varepsilon>0$, we stop updating our iterates because $\underline{x}^{(k)}$ is "close enough" to $\underline{x}^*$ (typical choice of ε are 10^{-4} or 10^{-6}, depending on how precise of a solution is required).

Before looking at some examples of (GD) in action, let's try to get some intuition as to why it might work. Suppose we are currently at point $\underline{x}^{(k)}$; let's form a **linear approximation** of $f(\underline{x})$ near $\underline{x}^{(k)}$ using its **first order Taylor Series approximation**

$$f(\underline{x}) \approx \underbrace{f(\underline{x}^{(k)})}_{\text{base point}} + \underbrace{\nabla f(\underline{x}^{(k)})^T}_{\text{how } f \text{ changes at } \underline{x}^{(k)}} \underbrace{(\underline{x}-\underline{x}^{(k)})}_{\text{direction of change}} \qquad (TS)$$



As you can see from the figure, (TS) is a very good approximation of $f(\underline{x})$ when $\underline{x}$ is not too far from $\underline{x}^{(k)}$, but gets worse as we move further away.

Let's use (TS) to define a next point $\underline{x}^{(k+1)}$ so that $f(\underline{x}^{(k+1)}) < f(\underline{x}^{(k)})$. If we define $\Delta\underline{x}^{(k)} = \underline{x}^{(k+1)} - \underline{x}^{(k)}$, then evaluating (TS) at point $\underline{x}^{(k+1)}$ becomes

$$f(\underline{x}^{(k+1)}) - f(\underline{x}^{(k)}) \approx \nabla f(\underline{x}^{(k)})^T \Delta\underline{x}^{(k)} \left(= \langle \nabla f(\underline{x}^{(k)}), \Delta\underline{x}^{(k)} \rangle\right)$$

so that if we want $f(\underline{x}^{(k+1)}) < f(\underline{x}^{(k)})$, then we should find a nearby $\underline{x}^{(k+1)}$ such that

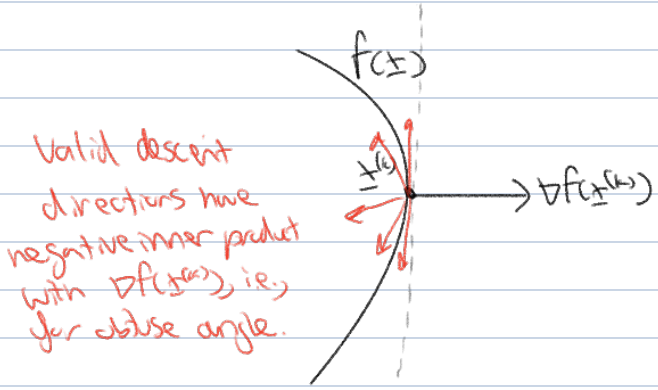
$$\nabla f(\underline{x}^{(k)})^T \Delta\underline{x}^{(k)} < 0. \qquad (*)$$

Now, assuming that $\nabla f(\underline{x}^{(k)}) \neq \underline{0}$ (so we're not at a local extremum), a clear choice for $\Delta\underline{x}^{(k)}$ is $-s\nabla f(\underline{x}^{(k)})$ for $s>0$ a step size chosen small

enough so that (TS) is a good approximation. In that case, we have

$$\nabla f(\underline{x}^{(k)})^T \Delta \underline{x}^{(k)} = -s \|\nabla f(\underline{x}^{(k)})\|^2 < 0.$$

In general though, any choice $\Delta \underline{x}^{(k)}$ such that (*) holds is a valid descent direction. Geometrically, this is illustrated in the picture below:



Example Let's use gradient descent to minimize $f(\underline{x}) = \frac{1}{2}\|\underline{x}\|^2$. This is a silly example, but one we can easily compute iterates by hand for. Here $\nabla f(\underline{x}) = \underline{x}$, so our descent direction is $-\nabla f(\underline{x}) = -\underline{x}$. Let's use $\underline{x} \in \mathbb{R}^2$ and use an initial guess of $\underline{x}^{(0)} = (1, 1)$. We'll use a step size of $s = \frac{1}{2}$. Then

$$\underline{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \underline{x}^{(1)} = \underline{x}^{(0)} - \frac{1}{2}\nabla f(\underline{x}^{(0)}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{2}\underline{x}^{(0)}$$

$$\underline{x}^{(2)} = \underline{x}^{(1)} - \frac{1}{2}\nabla f(\underline{x}^{(1)}) = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \left(\frac{1}{2}\right)^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \left(\frac{1}{2}\right)^2 \underline{x}^{(0)}$$

$$\cdots \quad \underline{x}^{(k)} = \left(\frac{1}{2}\right)^k \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$ So we see that $\underline{x}^{(k)} \to \underline{x}^* = \underline{0}$ exponentially quickly at rate $\left(\frac{1}{2}\right)^k$.

Example ONLINE NOTES PLEASE ADD A $\|A\underline{x} - \underline{b}\|^2$ EXAMPLE THAT'S MORE INTERESTING. aka TO DO MOSTLY COMPUTATIONAL. Show a good step size choice and one where it diverges.

Zig-Zags and What to do About Them

Let's consider a very simple optimization over $\underline{x} \in \mathbb{R}^2$ with cost function

$$f(x_1, x_2) = \frac{1}{2}(x^2 + by^2)$$

where we'll let $0 < b \leq 1$ vary. The optimal solution is obviously $(x^*, y^*) = (0, 0)$, but we'll use this to illustrate how gradient descent can get you into trouble sometimes.

Suppose we run gradient descent on f, and we further allow ourselves to pick the best possible step size s_k at each iteration, i.e., we choose step size

$$s_k = \underset{s>0}{\operatorname{argmin}}\ f(x^{(k)} - s\,\nabla f(x^{(k)})),$$

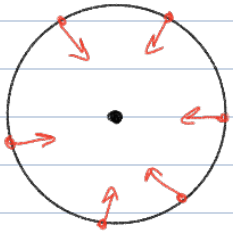
and then update $x^{(k+1)} = x^{(k)} - s_k \nabla f(x^{(k)})$.

This is called **exact line search** for selecting the step size, and is widely used in practice. If we use this choice of step size s_k, then it is possible to write an explicit formula for our iterates $(x^{(k)}, y^{(k)})$ as we progress down the bowl. Starting at $(x^{(0)}, y^{(0)}) = (b, 1)$, we set

$$x^{(k)} = b\left(\frac{b-1}{b+1}\right)^k, \quad y^{(k)} = \left(\frac{1-b}{1+b}\right)^k, \quad f(x^{(k)}, y^{(k)}) = \left(\frac{1-b}{1+b}\right)^{2k} f(x^{(0)}, y^{(0)}). \quad (*)$$

If $b = 1$, which corresponds to a function with level sets that are perfect circles, we succeed immediately in one step ($x^{(1)} = y^{(1)} = 0$). This is because the negative gradient always points directly to the optimal point $(0,0)$:

$b = 1$ case



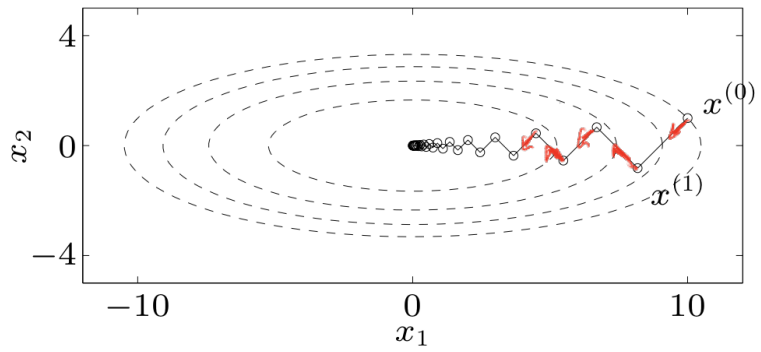
$-\nabla f(x)$ always points to optimal point $x^* = (0,0)$.

The real purpose of this example is seen when b is small. The crucial ratio in equation $(*)$ is

$$r = \frac{b-1}{b+1}.$$

If r is small, $(x^{(k)}, y^{(k)})$ converges to $(0,0)$ very quickly. However, if r is close to 1, then this convergence is very slow. For example, if $b = 1/10$, then $r = 9/11$; for $b = \frac{1}{100}$, $r = \frac{99}{101}$.

This means we need to take many more gradient steps to get close to $x^* = (0,0)$. The picture below highlights what's going wrong: for small b, the level sets become elongated ellipses, so that following gradients leads to us zigzagging our way to the origin instead of taking a straight path. It is this zigzagging that causes slow convergence.



So what's going wrong here? If we write $f(\underline{x})$ as a quadratic form, it is:

$$f(\underline{x}) = \begin{bmatrix} x \\ y \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \underline{x}^T K \underline{x}$$

Notice that the condition number of the matrix K is precisely $\frac{1}{b}$, which is large for small b. This means that one direction (in this case the x-axis) is penalized much more than the other (the y-axis). This leads to stretched out level sets, which leads to zigzags and slow convergence. How can we fix this?

Newton's Method (optional)

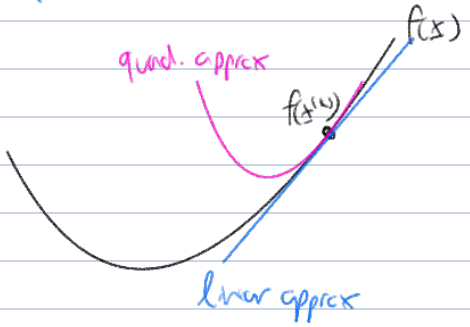
To derive the gradient descent method (GD), we used a first order approximation of $f(\underline{x})$ near $\underline{x}^{(k)}$ to figure out what direction we should move in. What the last example we saw showed was that when the gradient changes quickly (look at the direction of the gradients in the zigzag plot; they are all over the place!), things can go wrong. This suggests we should also account for how quickly the gradient changes: we need to compute the "gradient of the gradient", aka the Hessian of f, $\nabla^2 f(\underline{x})$. The Hessian of a function $f: \mathbb{R}^n \to \mathbb{R}$ is an $n \times n$ symmetric matrix with entries given by 2nd order partial derivatives of f:

$$[\nabla^2 f(\underline{x})]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(\underline{x}).$$

The Hessian tells us how quickly the gradient changes in the same way $f''(x)$ tells us how quickly $f'(x)$ changes for a scalar function $f(x)$. The Hessian lets us take a second order Taylor series approximation to our function $f(\underline{x})$ near our current guess $\underline{x}^{(k)}$:

$$f(\underline{x}) \approx f(\underline{x}^{(k)}) + \nabla f(\underline{x}^{(k)})^T (\underline{x} - \underline{x}^{(k)}) + (\underline{x} - \underline{x}^{(k)})^T \nabla^2 f(\underline{x}^{(k)}) (\underline{x} - \underline{x}^{(k)}), \quad (\star)$$

which provides a local quadratic approximation to $f(\underline{x})$ near $f(\underline{x}^{(k)})$:



As before, if we let $\Delta \underline{x}^{(k)} = \underline{x}^{(k+1)} - \underline{x}^{(k)}$, we can rewrite $(\star)$ as

$$f(\underline{x}^{(k+1)}) - f(\underline{x}^{(k)}) = (\Delta \underline{x}^{(k)})^T \nabla^2 f(\underline{x}^{(k)}) \Delta \underline{x}^{(k)} + \nabla f(\underline{x}^{(k)})^T \Delta \underline{x}^{(k)}. \quad (\star\star)$$

Since we want to make $f(\underline{x}^{(k+1)}) - f(\underline{x}^{(k)})$ as small as possible, it makes sense to pick $\Delta\underline{x}^{(k)}$ to minimize the RHS of (**), which is another minimization problem!

We'll focus on the setting where $\nabla^2 f(\underline{x}^{(k)})$ is positive definite: this corresponds to settings where our function is convex. In this case, the RHS of (**) is a positive definite quadratic function, which is minimized at

$$\Delta\underline{x}^{(k)} = -\nabla^2 f(\underline{x}^{(k)})^{-1} \nabla f(\underline{x}^{(k)}).$$

Using this descent direction instead of $-\nabla f(\underline{x}^{(k)})$ yields Newton's Method:

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \nabla^2 f(\underline{x}^{(k)})^{-1} \nabla f(\underline{x}^{(k)}).$$

The idea behind Newton's Method is to "unstretch" the stretched out directions, so that to our algorithm, the level sets of a function are locally circles. If we look at our bad example above, note that

$$\nabla^2 f(\underline{x}) = \begin{bmatrix} 1 & \\ & b \end{bmatrix} \Rightarrow \nabla^2 f(\underline{x})^{-1} = \begin{bmatrix} 1 & \\ & \frac{1}{b} \end{bmatrix}$$

so that for $\underline{x}^{(0)} = (b, 1)$, we have:

$$\underline{x}^{(1)} = \underline{x}^{(0)} - \nabla^2 f(\underline{x})^{-1} \nabla f(\underline{x})$$

$$= \begin{bmatrix} b \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & \\ & \frac{1}{b} \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

i.e., we converge in one step no matter what the choice of b is in $f(x,y) = \frac{1}{2}(x^2 + by^2)$!

The cost of this fast convergence though is that at each update, we need to solve a linear system of equations of the form

$$\nabla^2 f(\underline{x}^{(k)}) \Delta\underline{x}^{(k)} = \nabla f(\underline{x}^{(k)})$$

which may be expensive if $\underline{x}$ is a high-dimensional vector. It is for this reason that gradient descent based methods are the predominant methods used in machine learning, where often times the dimensionality of $\underline{x} \in \mathbb{R}^n$ can be on the order of millions or billions.