8
Nonlinear Systems of Equations

We've seen multiple methods for solving linear systems of equations. In this chapter, we develop a method using linear algebra to solve nonlinear systems of equations. We begin with Newton's method for finding the roots of a single nonlinear equation. Then we will generalize the method to systems of equations using a matrix formalism.

8.1 Nonlinear Functions

A nonlinear function is, simply put, a function that fails the tests for linearity. You might have been surprised that the affine function $f(x) = ax + b$ was nonlinear. The functions $f(x) = \cos x$, $f(x) = x^2$, and $f(x) = \log x$ are all nonlinear with respect to the independent variable x.

By convention we usually write nonlinear functions in the form

$$f(x) = 0$$

This convention is not a limitation, as any nonlinear function that has a nonzero right hand side can be rewritten by moving the right hand terms to the left side. By always writing nonlinear functions in this way, we can solve the function by identifying values where the function equals zero, i.e. by finding the *roots* of the function. For example, the equation

$$(x-1)^3 = 8$$

has a unique solution when $x = 3$. We can rewrite this equation as the function

$$f(x) = (x-1)^2 - 8 = 0$$

Notice that this function is equal to zero (i.e. has a root) when $x = 3$. The solutions are the same.

Linear systems have exactly zero, one, or infinitely many solutions. By contrast, nonlinear systems can have any number of solutions. The function $f(x) = x^2 - 4$ has two roots: $x = 2$ and $x = -2$. Unlike linear

systems, there is no grand solvability theorem for nonlinear systems. Except in special cases (for example, polynomials), we cannot tell *a priori* how many unique solutions exist for a nonlinear equation. Even when we know a solution exists, we do not have a general procedure like Gaussian elimination for finding solutions to nonlinear equations. Instead, we often rely on numerical techniques to find some of the roots of nonlinear functions.

8.2 Newton's Method

Given a function $f(x)$, how do we find its roots? One powerful method builds on an observation regarding the tangent lines of $f(x)$ near its roots. Imagine we are at a point x_0 that is near a root. The tangent line of $f(x)$ at the point x_0 will itself have a root that is closer to the root of $f(x)$. Let's call this new point x_1.

If we draw another tangent line for f at x_1, we see that the root of the tangent line is again closer to the root of f. We can repeat this procedure again and again, each time moving closer to the root of f. Rather than solve the nonlinear f, we only need to solve a series of affine equations describing the tangent line at each iteration.

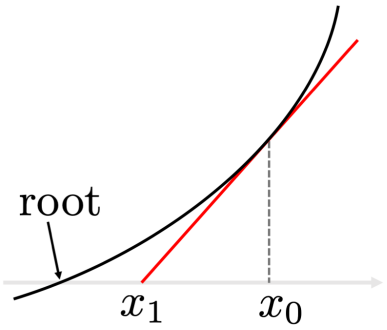

Figure 8.1: If a point x_0 is close to the root of a function (black), the tangent line (red) intersects the horizontal axis at a point x_1 that is closer to the root.

Let's formalize the above procedure. The starting point x_0; the values of f and its derivative f'; and the root x_1 of the tangent line are related by

$$f'(x_0) = \frac{f(x_0)}{x_0 - x_1}$$

You can interpret this formula as "the slope of the tangent line at x_0 ($f'(x_0)$) is equal to the height of the function at x_0 ($f(x_0)$) divided by the distance between x_0 and x_1." Rearranging, we can find the root of the tangent line based on values at our current point.

In other words, the slope of the tangent line $f'(x_0)$ is its rise $f(x_0)$ divided by its run $(x_0 - x_1)$.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Now we know the location of x_1, a point closer to the root of the original function f. We can apply the same procedure starting at x_1 to find a closer point x_2, and so on.

Newton published a very limited version of the method that bears his name. British mathematician Thomas Simpson was the first to apply the technique to general systems of nonlinear equations. He also noted connections between nonlinear systems and optimization.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$
$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$
$$\vdots$$
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

8.3 Convergence of Newton's Method

Let's find a root for the equation

$$f(x) = (x-4)^3 - 2x$$

By plotting the function, we see there is a root somewhere between $x = 6$ and $x = 6.5$. We can use Newton's Method to find a more precise estimate of the root. We first calculate the derivative

$$f'(x) = 3(x-4)^2 - 2$$

Let's choose our initial guess to be $x_0 = 6.0$. We're ready to calculate x_1.

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= x_0 - \frac{(x_0-4)^3 - 2x_0}{3(x_0-4)^2 - 2} \\ &= 6.0 - \frac{(6.0-4)^3 - 2(6.0)}{3(6.0-4)^2 - 2} \\ &= 6.4 \end{aligned}$$

We can check if we've found a root by evaluating $f(x_1)$. If x_1 is a root, $f(x_1)$ should equal zero.

$$f(x_1) = f(6.4) = 1.024 \neq 0$$

We haven't arrived at a root yet. Let's try another iteration of Newton's method to find a second guess (x_2) using x_1.

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\ &= x_1 - \frac{(x_1-4)^3 - 2x_1}{3(x_1-4)^2 - 2} \\ &= 6.4 - \frac{(6.4-4)^3 - 2(6.4)}{3(6.4-4)^2 - 2} \\ &= 6.332984293 \end{aligned}$$

The new value x_2 is closer to being a root: $f(6.332984293) = 0.03203498$. We can always move closer using more iterations as shown in the following table.

i	x_i	$f(x_i)$
0	6	-4
1	6.4	1.024
2	6.332984293	0.032034981
3	6.330748532	0.000034974
4	6.330746086	4.18421×10^{-11}

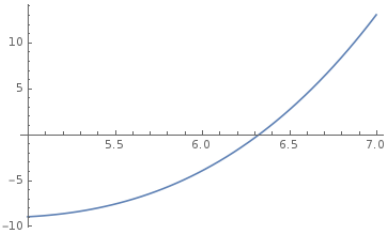

Figure 8.2: The function $f(x) = (x-4)^3 - 2x$ has a root between $x = 6$ and $x = 6.5$.

When studying numerical methods we will extend our answers far beyond the number of significant figures. As engineers we later "trim" our answers to an appropriate number of significant figures based on the uncertainty in the system.

Newton's method converges quadratically once the x_i are close to the actual root. "Close" is not well defined and varies with each function. If an initial guess is far from the true root, Newton's method can either 1.) converge slowly until it becomes close enough for quadratic converge to kick in, or 2.) not converge at all. If Newton's method is converging slowly or diverges, you should try a different initial guess.

The quadratic convergence stems from our use of a linear approximation for the function, leaving a residual bounded by the quadratic terms.

8.4 Multivariable Functions

Newton's method works well for nonlinear functions of a single variable. We use a variant of Newton's method to solve multivariable functions. Multivariable functions accept a vector of inputs and produce a vector of outputs. We write the names of multivariable functions using bold, non-italicized font – $\mathbf{f}(\mathbf{x})$ – to remind us that a multivariable functions return a vector of outputs.

Multivariable functions are also called *multivariate* or *vector-valued* functions.

We're already familiar with linear multivariable functions like $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x}$. This function accepts a vector of inputs ($\mathbf{x}$) and returns another vector of outputs ($\mathbf{A}\mathbf{x}$). We can also define nonlinear multivariable functions. An example with three inputs and three outputs is

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - x_3 \\ x_3^2 + 2x_2 \\ \cos x_1 \end{pmatrix}$$

If $\mathbf{x} = \begin{pmatrix} 0 \\ -1 \\ 2 \end{pmatrix}$, then

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} 0 - 2 \\ 2^2 + 2(-1) \\ \cos 0 \end{pmatrix} = \begin{pmatrix} -2 \\ 2 \\ 1 \end{pmatrix}$$

It's sometimes convenient to talk individually about the entries in the nonlinear function. We can write a multivariable function using the following notation

We use lowercase and italicized font (f_i) when referencing individual entries in a multivariable function since each entry produces only a single output.

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{pmatrix}$$

For the example above, $f_1 = x_1 - x_3$; $f_2 = x_3^2 + 2x_2$; and $f_3 = \cos x_1$.

8.5 The Jacobian Matrix

For functions of a single variable, Netwon's method uses the derivative to construct a linear approximation. The multivariable analog of

the derivative is matrix of partial derivatives called the *Jacobian*, which we write as $\mathbf{J}(\mathbf{x})$.

The Jacobian is named after German mathematician Carl Gustav Jacob Jacobi. I assume it is based on his last name, or possibly his second-to-last name.

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

The (i,j)th entry in the Jacobian is the partial derivative of the ith function with respect to the jth variable. If a multivariable function has n inputs and n outputs, its Jacobian is a square $n \times n$ matrix.

Let's compute the Jacobian for the function $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - x_3 \\ x_3^2 + 2x_2 \\ \cos x_1 \end{pmatrix}$.

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1}(x_1 - x_3) & \frac{\partial}{\partial x_2}(x_1 - x_3) & \frac{\partial}{\partial x_3}(x_1 - x_3) \\ \frac{\partial}{\partial x_1}\left(x_3^2 + 2x_2\right) & \frac{\partial}{\partial x_2}\left(x_3^2 + 2x_2\right) & \frac{\partial}{\partial x_3}\left(x_3^2 + 2x_2\right) \\ \frac{\partial}{\partial x_1}(\cos x_1) & \frac{\partial}{\partial x_2}(\cos x_1) & \frac{\partial}{\partial x_3}(\cos x_1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 2x_3 \\ -\sin x_1 & 0 & 0 \end{pmatrix}$$

8.6 Multivariable Newton's Method

For functions of a single variable, Newton's method iterates with the formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Using a multivariable linear approximation, we can define the multivariable analogue of Newton's method.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{J}^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)$$

As an example, let's find a root of the function

$$\mathbf{f} = \begin{pmatrix} x_1 x_2 - 2 \\ -x_1 + 3x_2 + 1 \end{pmatrix}$$

First we calculate the Jacobian matrix.

$$\mathbf{J}(\mathbf{x}) = \begin{pmatrix} x_2 & x_1 \\ -1 & 3 \end{pmatrix}$$

Using an initial guess of $\mathbf{x}_0 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$ we begin iterating.

$$\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \mathbf{J}^{-1}(\mathbf{x}_0)\mathbf{f}(\mathbf{x}_0) \\
&= \begin{pmatrix} -1 \\ -1 \end{pmatrix} - \begin{pmatrix} -1 & -1 \\ -1 & 3 \end{pmatrix}^{-1} \begin{pmatrix} (-1)(-1) - 2 \\ -(-1) + 3(-1) + 1 \end{pmatrix} \\
&= \begin{pmatrix} -2 \\ -1 \end{pmatrix}
\end{aligned}$$

Now we use $\mathbf{x}_1$ to find the next guess $\mathbf{x}_2$.

$$\begin{aligned}
\mathbf{x}_2 &= \mathbf{x}_1 - \mathbf{J}^{-1}(\mathbf{x}_1)\mathbf{f}(\mathbf{x}_1) \\
&= \begin{pmatrix} -2 \\ -1 \end{pmatrix} - \begin{pmatrix} -1 & -2 \\ -1 & 3 \end{pmatrix}^{-1} \begin{pmatrix} (-2)(-1) - 2 \\ -(-2) + 3(-1) + 1 \end{pmatrix} \\
&= \begin{pmatrix} -2 \\ -1 \end{pmatrix}
\end{aligned}$$

Our guess $\mathbf{x}_2$ is exactly equal to the previous guess $\mathbf{x}_1$. Since our guess didn't change we are probably at a root. We can check by evaluating $\mathbf{f}(\mathbf{x}_2)$.

$$\mathbf{f}(\mathbf{x}_2) = \begin{pmatrix} (-2)(-1) - 2 \\ -(-2) + 3(-1) + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Indeed, the vector $\begin{pmatrix} -2 \\ -1 \end{pmatrix}$ is a solution to our equation.

Nonlinear systems often have many solutions. Newton's method converges to the solution nearest the initial guess. If we chose the point $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ as our initial guess, Newton's method on the same function would converge to the root $\mathbf{x} = \begin{pmatrix} 3 \\ 2/3 \end{pmatrix}$ after four iterations.

"Nearest" in the topological sense, i.e. the solution that is down the gradient of the function at the initial guess.

8.7 Practical Considerations

Solving nonlinear equations is an art. Here are some tips.

- Nonlinear equations rarely have a single solution. Solvers try many (hundreds or thousands) of initial guesses to find several solutions. There is no general method for determining the total number of roots for a nonlinear system.
- Software packages like MATLAB's `fsolve` function can find roots with a variety of algorithms. Many techniques find points near roots and use Newton's method to finish the search.
- Software packages often allow users to provide both the function and the Jacobian. Knowing the Jacobian explicitly almost always

improves speed and numerical stability. If the user doesn't provide a Jacobian, the software will estimate the Jacobian at every iteration using finite differences.

- Single variable Newton's method requires the function be continuously differentiable. Multivariable functions require the Jacobian be invertible. So-called "gradient free" algorithms are available for functions with poorly behaving, computationally expensive, or discontinuous derivatives.

- The multivariable Newton's method involves inverting the Jacobian, which is computationally expensive. Instead, numerical solvers rearrange the iteration equation:

$$\begin{aligned}\mathbf{x}_{i+1} &= \mathbf{x}_i - \mathbf{J}^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)\\ \mathbf{J}(\mathbf{x}_i)\mathbf{x}_{i+1} &= \mathbf{J}(\mathbf{x}_i)\mathbf{x}_i - \mathbf{J}(\mathbf{x}_i)\mathbf{J}^{-1}(\mathbf{x}_i)\mathbf{f}(\mathbf{x}_i)\\ \mathbf{J}(\mathbf{x}_i)\left(\mathbf{x}_{i+1} - \mathbf{x}_i\right) &= -\mathbf{f}(\mathbf{x}_i)\end{aligned}$$

  In this form, the solver can use Gaussian elimination on the augmented matrix $[\mathbf{J}(\mathbf{x}_i) \;\; -\mathbf{f}(\mathbf{x}_i)]$ to solve for $\mathbf{x}_{i+1} - \mathbf{x}_i$; adding $\mathbf{x}_i$ gives the new estimate for $\mathbf{x}_{i+1}$.

- We've focused on multivariable funtions with an equal number of inputs and outputs. These functions have square Jacobian matrices. If the number of inputs and outputs differ, the pseudoinverse of the nonsquare Jacobian can be used to find roots. This technique is called Gauss-Newton Iteration.

9
Optimization, Convexity, and Hyperplanes

We formulated the least squares method and linear regression as optimization problems. Our goal was to minimize the sum of the squared errors by choosing parameters for the linear model. Optimization problems have enormous utility in data science, and most model fitting techniques are cast as optimizations. In this chapter, we will develop a general framework for describing and solving several classes of optimization problems. We begin by reviewing the fundamentals of optimization. Next, we discuss convexity, a property that greatly simplifies the search for optimal solutions. Finally we derive vector expressions for common geometric constructs and show how linear systems give rise to convex problems.

9.1 Optimization

Optimization is the process of minimizing or maximizing a function by selecting values for a set of variables or parameters (called *decision variables*). If we are free to choose any values for the decision variables, the optimization problem is *unconstrained*. If our solutions must obey a set of constraints, the problem is a *constrained optimization*. In constrained optimization, any set of values for the decision variables that satisfies the constraints is called a *feasible solution*. The goal of constrained optimization is to select the "best" feasible solution.

Optimization problems are formulated as either minimizations or maximizations. We don't need to discuss minimization and maximization separately, since minimizing $f(x)$ is equivalent to maximizing $-f(x)$. Any algorithm for minimizing can be used for maximizing by multiplying the objective by -1, and vice versa. For the rest of this chapter, we'll talk about minimizing functions. Keep in mind that everything we discuss can be applied to maximization problems by switching the sign of the objective.

During optimization we search for minima. A minimum can either be *locally* or *globally* minimal. A global minimum is has the smallest

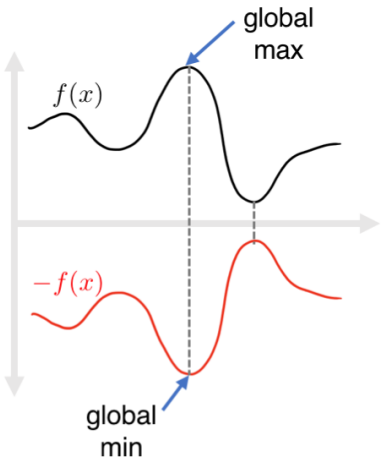

Figure 9.1: The maximum of a function $f(x)$ can be found by minimizing $-f(x)$.

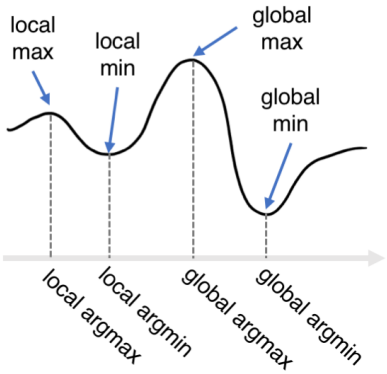

Figure 9.2: Minima and maxima of a function can be local or global.

objective value of any feasible solution. A local minimum has the smallest objective value for any of the feasible solutions in the surrounding area. The input to a function that yields the minimum is called the *argmin*, since it is the argument to the function that gives the minimum. Similarly, the *argmax* of a function is the input that gives the function's maximum. Consider the function $f(x) = 3 + (x-2)^2$. This function has a single minimum, f(2) = 3. The minimum is 3, while the argmin is $x = 2$, the value of the decision variable at which the minimum occurs. For optimization problems, the minimum (or maximum) is called the *optimal objective value*. The argmin (or argmax) is called the *optimal solution*.

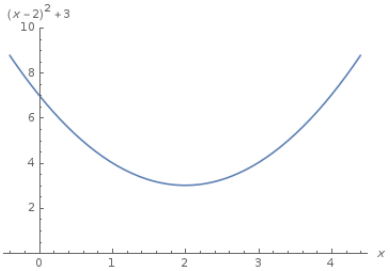

Figure 9.3: The function $f(x) = 3 + (x-2)^2$ has a minimum of $f = 3$ at argmin $x = 2$.

Unconstrained Optimization

You already know how to solve unconstrained optimization problems in a single variable: set the derivative to the function equal to zero and solve. This method of solution relies on the observation that both maxima and minima occur when the slope of a function is zero. However, it is important to remember that not all roots of the derivative are maxima or minima. Inflection points (where the derivative changes sign) also have derivatives equal to zero. You must always remember to test the root of the derivative to see if you've found a minimum, maximum, or inflection point. The easiest test involves the sign of the second derivative. If the second derivative at the point is positive, you've found a minimum. If it's negative, you've found a maximum. If the second derivative is zero, you've found an inflection point.

Any point where the derivative of a function equals zero is called an *extreme point* or *extremum*. Setting the derivative of a function equal to zero and solving for the extrema is called *extremizing* a function.

A similar approach works for optimizing multivariate functions. In this case one solves for points where the gradient is equal to zero, checking that you've not found an inflection point (called "saddle points" in higher dimensions).

Constrained Optimization

Constrained optimization problems cannot be solved by finding roots of the derivatives of the objective. Why? It is possible that the minima or maxima of the unconstrained problem lie outside the feasible region of the constrained problem. Consider our previous example of $f(x) = 3 + (x-2)^2$, which we know has an argmin at $x = 2$. Say we want to solve the constrained problem

$$\min f(x) = 3 + (x-2)^2 \quad \text{s.t.} \quad x \leq 1$$

The root of the derivative of f is still at $x = 2$, but values of x greater than one are not feasible. From the graph we can see that the minimum feasible value occurs at $x = 1$. The value of the derivative at $x = 1$ is -2, not zero.

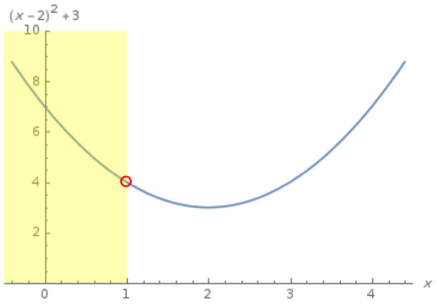

Figure 9.4: The yellow region is the feasible space ($x \leq 1$). The global argmin occurs at $x = 1$. The derivative of the function is not zero at this point.

In general, constrained optimization is a challenging field. Finding global optima for constrained problems is an unsolved area or research, one which is beyond the scope of this course. However, there are classes of problems that we can solve to optimality using the tools of linear algebra. These problems form the basis of many advanced techniques in data science.

9.2 *Convexity*

Many "solvable" optimization problems rely on a property called *convexity*. Both sets and functions can be convex.

Convex sets

A set of points is *convex* if given any two points in the set, the line segment connecting these points lies entirely in the set. You can move from any point in the set to any other point in the set without leaving the set. Circles, spheres, and regular polygons are examples of convex sets.

To formally define convexity, we construct the line segment between any two points in the set.

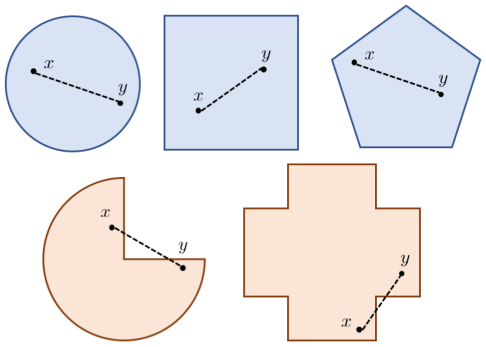

Figure 9.5: The blue shapes are convex. The red shapes are not convex.

Definition. *A set S is convex if and only if given any $\mathbf{x} \in S$ and $\mathbf{y} \in S$ the points $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ are also in S for all scalars $\lambda \in [0,1]$.*

The expression $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ is called a *convex combination* of $\mathbf{x}$ and $\mathbf{y}$. A convex combination of two points contains all points on the line segment between the two points. To see why, consider the 1-dimensional line segment between points 3 and 4.

$$\lambda(3) + (1-\lambda)(4) = 4 - \lambda, \quad \lambda \in [0,1]$$

When $\lambda = 0$, the value of the combination is 4. As λ moves from 0 to 1, the value of the combination moves from 4 to 3, covering all values in between.

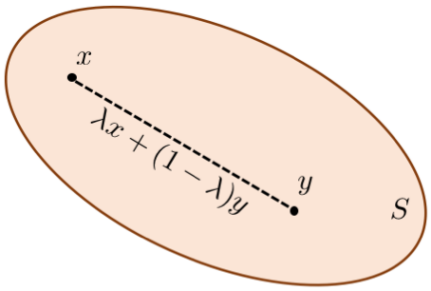

Figure 9.6: The segment connecting x and y can be defined as $\lambda x + (1-\lambda)y$ for $\lambda \in [0,1]$.

Convex combinations work in higher dimensions as well. The convex combination of the vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$ is

$$\lambda \begin{pmatrix}1\\0\end{pmatrix} + (1-\lambda)\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\lambda\\1-\lambda\end{pmatrix}$$

The combination goes from the first point $\begin{pmatrix}0\\1\end{pmatrix}$ when $\lambda = 0$ to the second point $\begin{pmatrix}1\\0\end{pmatrix}$ when $\lambda = 1$. Halfway in between, $\lambda = 1/2$ and the

combination is $\begin{pmatrix}1/2\\1/2\end{pmatrix}$, which is midway along the line connecting $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$. Sometimes it is helpful to think of a convex combination as a weighted sum of $\mathbf{x}$ and $\mathbf{y}$. The weighting (provided by λ) moves the combination linearly from $\mathbf{y}$ to $\mathbf{x}$ as λ goes from 0 to 1.

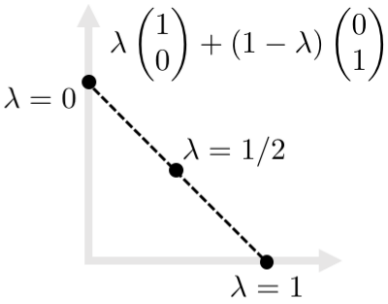

Figure 9.7: A convex combination in 2D.

Convex functions

There is a related definition for *convex functions*. This definition formalizes our visual idea of convexity (lines that curve upward) and concavity (lines that curve downward).

Definition 1. *A function f is convex if and only if*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}), \quad \lambda \in [0,1]$$

This definition looks complicated, but the intuition is simple. If we plot a convex (upward curving) function, any chord – a segment drawn between two points on the line – should lie above the line. We can define the chord between any two points on the line, say $f(\mathbf{x})$ and $f(\mathbf{y})$ as a convex combination of these points, i.e. $\lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y})$. This is the right hand side of the above definition. For convex functions, we expect this cord to be greater than or equal to the function itself over the same interval. The interval is the segment from $\mathbf{x}$ to $\mathbf{y}$, or the convex combination $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$. The values of the function over this interval are therefore $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y})$, which is the left hand side of the definition.

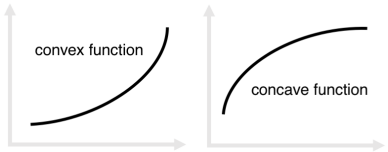

Figure 9.8: Convex functions curve upward. Concave functions curve downward.

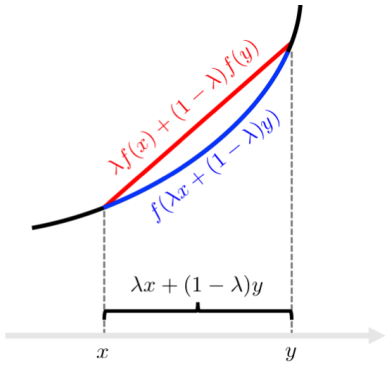

Figure 9.9: The chord connecting any two points of a convex function (red) lies above the function (blue).

Convexity in Optimization

Why do we care about convexity? In general, finding local optima during optimization is easy; just pick a feasible point and move downward (during minimization) until you arrive at a local minimum. The truly hard part of optimization is finding global optima. How can you be assured that your local optimum is a global optimum unless you try out all points in the feasible space?

Fortunately, convexity solves the local vs. global challenge for many important problems, as we see with the following theorem.

Theorem. *When minimizing a convex function over a convex set, all local minima are global minima.*

Convex functions defined over convex sets must have a special shape where no *strictly* local minima exist. There can be multiple local minima, but all of these local minima must have the same value (which is the global minimum).

All local minima are *less than or equal to* the global minimum. Strictly local minima must be *less than* the global minimum.

Let's prove that all local minima are global minima when minimizing a convex function over a convex set.

Proof. Suppose the convex function f has a local minimum at $\mathbf{x}'$ that is not the global minimum (which is at $\mathbf{x}^*$). By the convexity of f,

$$f(\lambda\mathbf{x}' + (1-\lambda)\mathbf{x}^*) \leq \lambda f(\mathbf{x}') + (1-\lambda)f(\mathbf{x}^*)$$

Since $\mathbf{x}'$ is at a local, but not global, minimum, we know that $f(\mathbf{x}') > f(\mathbf{x}^*)$. If we replace $f(\mathbf{x}^*)$ on the right hand side by the larger quantity $f(\mathbf{x}')$, the inequality ($\leq$) becomes a strict inequality ($<$). (Even if both sides were equal, adding a small amount to the right hand side would still make it larger.) We now have

$$f(\lambda\mathbf{x}' + (1-\lambda)\mathbf{x}^*) < \lambda f(\mathbf{x}') + (1-\lambda)f(\mathbf{x}')$$

which, by simplifying the right hand side, becomes

$$f(\lambda\mathbf{x}' + (1-\lambda)\mathbf{x}^*) < f(\mathbf{x}')$$

This statement says that the value of the function f on any point on the line segment from $\mathbf{x}'$ to $\mathbf{x}^*$ is less than the value of the function at $\mathbf{x}'$. If this is true, we can find a point arbitrarily close to $\mathbf{x}'$ that is below our supposed local minimum $f(\mathbf{x}')$. Clearly, $f(\mathbf{x}')$ cannot be a local minimum if we can find a lower point arbitrarily closer to it. Our conclusion contradicts our original supposition. No local minimum can exist that are not equal to the global minimum. □

For a simpler, yet less intuitive argument, let $\lambda = 1$. Then the inequality becomes $f(\mathbf{x}') < f(\mathbf{x}')$, which is nonsense.

The previous proof seemed to rely only on the convexity of the objective function, not on the convexity of the solution set. The role of convexity of the set is hidden. When we make an argument about a line drawn from the local to the global minimum, we assume that all the points on the line are feasible. Otherwise, it does not matter if they have a lower objective than the local minimum, since they would not be allowed. By assuming the solution set is convex, we are assured that any point on this line is also feasible.

Convexity of Linear Systems

This course focuses on linear functions and systems of linear equations. It would be enormously helpful if linear functions and the solution set of linear systems were convex. Then we can look for local optima during optimization and know that we've found global optima.

Let's first prove the convexity of linear functions. For a function to be convex, we require that a line segment connecting any two points in the line lie above or one the line. For linear functions, this is intuitively

true. The line segment connecting any two points is the line itself, so it always lies on the line. As a more formal argument, we describe a linear function as the product between a vector of coefficients $\mathbf{c}$ and $\mathbf{x}$, i.e. $f(\mathbf{x}) = \mathbf{c}^\mathsf{T}\mathbf{x}$. Let's start with the values of the function over the range spanned by arbitrary points $\mathbf{x}$ and $\mathbf{y}$. The segment of the domain corresponds to the convex combination $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$. The values of the function over this interval are

By convention, all vectors are column vectors, including $\mathbf{c}$; this requires a transposition to be conformable for multiplication by $\mathbf{x}$.

$$\begin{aligned} f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \mathbf{c}^\mathsf{T}(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \\ &= \mathbf{c}^\mathsf{T}\lambda\mathbf{x} + \mathbf{c}^\mathsf{T}(1-\lambda)\mathbf{y} \\ &= \lambda\mathbf{c}^\mathsf{T}\mathbf{x} + (1-\lambda)\mathbf{c}^\mathsf{T}\mathbf{y} \\ &= \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \end{aligned}$$

which satisfies the definition of convexity: $f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})$.

Now let's turn to a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. We want to show that the set of all solutions for this system (the *solution space*) is convex. Let's assume we have two points in the solution space, $\mathbf{x}$ and $\mathbf{y}$. Since $\mathbf{x}$ and $\mathbf{y}$ are solutions, we know that $\mathbf{A}\mathbf{x} = \mathbf{b}$ and $\mathbf{A}\mathbf{y} = \mathbf{b}$. If the solution set is convex, any point in the convex combination of $\mathbf{x}$ and $\mathbf{y}$ is also a solution.

Following the conventions of the optimization field, we call the right hand side of linear systems the column vector $\mathbf{b}$, not $\mathbf{y}$ as we have said previously.

$$\begin{aligned} \mathbf{A}(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) &= \mathbf{A}\lambda\mathbf{x} = \mathbf{A}(1-\lambda)\mathbf{y} \\ &= \lambda\mathbf{A}\mathbf{x} + (1-\lambda)\mathbf{A}\mathbf{y} \\ &= \lambda\mathbf{b} + (1-\lambda)\mathbf{b} \\ &= \mathbf{b} \end{aligned}$$

Since $\mathbf{A}(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) = \mathbf{b}$, we know that all points on the line between $\mathbf{x}$ and $\mathbf{y}$ are solutions, so the solution set is convex.

9.3 Geometry of Linear Equations

Why do linear systems have convex solution spaces? Before answering, we should understand the shape of individual equations (rows) in the systems $\mathbf{A}\mathbf{x} = \mathbf{b}$. The equation corresponding to the i^{th} row is

$$\mathbf{A}(i,:) \cdot \mathbf{x} = b_i$$

which we will simplify by using the notation

$$\mathbf{a} \cdot \mathbf{x} = b$$

where $\mathbf{a}$ and $\mathbf{x}$ are vectors and the value b is a scalar. In two dimensions, this expression defines a line

$$a_1 x_1 + a_2 x_2 = b$$

The above representation of a line is the *standard form*, which differs from the *slope-intercept* form you remember from algebra ($y = mx + b$). It seems intuitive why the slope-intercept form is a line; a change in x produces a corresponding change $m\Delta x$ in y, with an intercept b when $x = 0$. What is the analogous reasoning for why $\mathbf{a} \cdot \mathbf{x} = b$ is a line?

First, we note that the vector $\mathbf{a}$ always point perpendicular, or *normal* to the line. For the horizontal line $y = 3$, the vector $\mathbf{a} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ points vertically. For the vertical line $x = 3$, the vector $\mathbf{a} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ point horizontal. For the line $x_1 + x_2 = 1$, $\mathbf{a} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, which is still perpendicular to the original line.

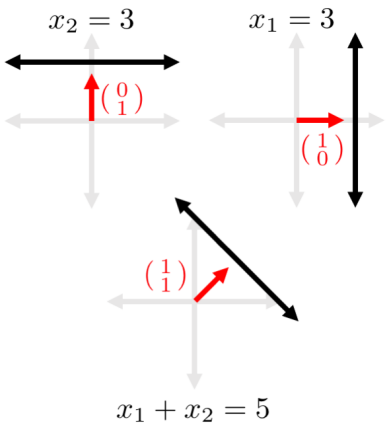

Figure 9.10: The vector $\mathbf{a}$ is always normal (perpendicular) to the line $\mathbf{a} \cdot \mathbf{x} = b$.

To help visualize the equation $\mathbf{a} \cdot \mathbf{x} = b$, we need to know the length of $\mathbf{a}$. The easiest solution is to normalize $\mathbf{a}$ by dividing both sizes of the equation by the norm of $\mathbf{a}$.

$$\frac{1}{\|\mathbf{a}\|}\mathbf{a} \cdot \mathbf{x} = b/\|\mathbf{a}\|$$

If we use our previous notation of $\hat{\mathbf{a}}$ for the normalized form of $\mathbf{a}$ and define scalar $d = b/\|\mathbf{a}\|$, we have

$$\hat{\mathbf{a}} \cdot \mathbf{x} = d$$

The equation $\hat{\mathbf{a}} \cdot \mathbf{x} = d$ is called the Hesse normal form of a line, plane, or hyperplane.

We know that $\hat{\mathbf{a}}$ is a unit vector normal to the line. What is the meaning of d? Let's compute the dot product $\hat{\mathbf{a}} \cdot \mathbf{x}$ using an arbitrary point $\mathbf{x}$ on the line.

$$d = \hat{\mathbf{a}} \cdot \mathbf{x} = \|\hat{\mathbf{a}}\| \, \|\mathbf{x}\| \cos\theta = \|\mathbf{x}\| \cos\theta$$

Thus, d is the projection of the magnitude of $\mathbf{x}$ onto the normal vector. For any point on the line, this projection is always the same length – the distance between the origin and the nearest point on the line. Conversely, a line is the set of all vectors whose projection against a vector $\hat{\mathbf{a}}$ is a constant distance (d) from the origin.

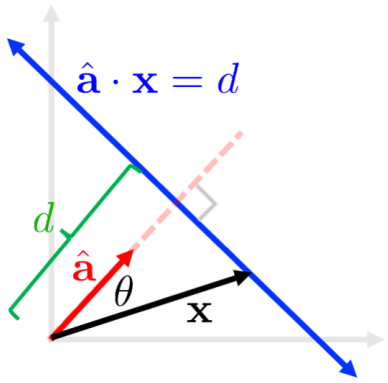

Figure 9.11: A line is the set of all points $\mathbf{x}$ whose projection onto $\hat{\mathbf{a}}$ is the distance d.

The same interpretation follows in higher dimensions. In 3D, the expression $\hat{\mathbf{a}} \cdot \mathbf{x} = d$ defines a plane with normal vector $\hat{\mathbf{a}}$ at a distance d from the origin. This definition fits with the algebraic definition of a plane that you may have seen previously: $a_1x_1 + a_2x_2 + a_3x_3 = d$. In higher dimensions (four or more), this construct is called a *hyperplane*.

Remember that when analyzing an expression of the form $\hat{\mathbf{a}} \cdot \mathbf{x} = d$, the constant on the right hand side (d) is only equal to the distance between the line and the origin if the vector $\hat{\mathbf{a}}$ is normalized. For example, the line

$$3x_1 + 4x_2 = 7$$

has coefficient vector $\mathbf{a} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, which is not normalized. To normalize $\mathbf{a}$, we divide both sides by $\|\mathbf{a}\| = \sqrt{3^2 + 4^2} = 5$, yielding

$$\frac{3}{5}x_1 + \frac{4}{5}x_2 = \frac{7}{5}$$

Now we can say that the distance between this line and the origin is 7/5.

9.4 *Geometry of Linear Systems*

The equation $\hat{\mathbf{a}} \cdot \mathbf{x} = d$ defines a hyperplane. It is also a single row in the linear system $\mathbf{Ax} = \mathbf{b}$. What does the entire system of equations look like? First, let's consider a set of three equations in two dimensions (so we can visualize them as lines). Solutions to $\mathbf{Ax} = \mathbf{b}$ are points of intersection of all three equations. If the lines are parallel, no solutions exist. If the lines all intersect at one point, there is a unique solution. If the lines are *colinear* (all fall upon the same line), infinitely many solutions exist. Note that these are the only options – zero, one, or infinitely many solutions, just as predicted by the grand solvability theorem. It is impossible to draw three straight lines that intersect in only two places.

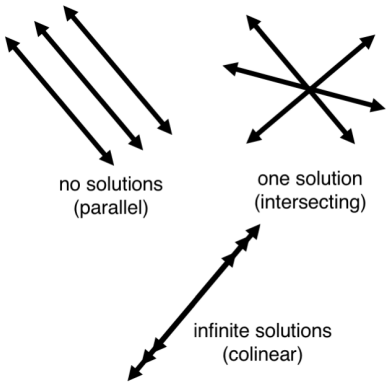

Figure 9.12: A system of linear equations can have zero, one, or infinitely many points of intersection.

If linear systems $\mathbf{Ax} = \mathbf{b}$ are a set of intersecting lines in 2D, what is do the inequalities $\mathbf{Ax} \leq \mathbf{b}$ represent? Each inequality states that the projection of $\mathbf{x}$ onto the normal vector $\mathbf{a}$ must be less than d. These points form a *half-plane* – all the points on one side of a hyperplane. The system $\mathbf{Ax} \leq \mathbf{b}$ has a solution space that is the overlap of multiple half-planes (one for each row in $\mathbf{A}$). As we proved earlier, this solution set is a convex set.

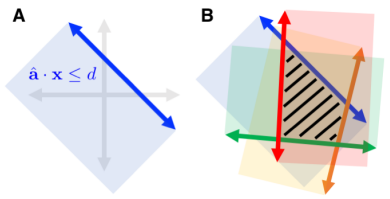

Figure 9.13: **A**. A single inequality defines a half-plane. **B**. Multiple half-planes intersect to form a convex solution set for the system $\mathbf{Ax} \leq \mathbf{b}$.