Proof: Consider the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, (\sqrt{n}-1)\sqrt{n}$. These are the columns for which $S(\sqrt{n}, \sqrt{n})$ has value 1. The element of the matrix Z in the row $j\sqrt{n}$ of column $k\sqrt{n}$, $0 \leq k < \sqrt{n}$ is $z^{nkj} = 1$. Thus, the product of these rows of Z times the vector $S(\sqrt{n}, \sqrt{n})$ equals $\sqrt{n}$ and the $1/\sqrt{n}$ normalization yields $f_{j\sqrt{n}} = 1$.

For rows whose index is not of the form $j\sqrt{n}$, the row b, $b \neq j\sqrt{n}$, $j \in \{0, \sqrt{n}, \ldots, \sqrt{n}-1\}$, the elements in row b in the columns $0, \sqrt{n}, 2\sqrt{n}, \ldots, (\sqrt{n}-1)\sqrt{n}$ are $1, z^b, z^{2b}, \ldots, z^{(\sqrt{n}-1)b}$ and thus $f_b = \frac{1}{\sqrt{n}}\left(1 + z^b + z^{2b} \cdots + z^{(\sqrt{n}-1)b}\right) = \frac{1}{\sqrt{n}} \frac{z^{\sqrt{n}b}-1}{z^b-1} = 0$ since $z^{b\sqrt{n}} = 1$ and $z^b \neq 1$. ∎

ms better suited to perhaps a homework question

10.5 Gradient

The gradient of a function $f(\mathbf{x})$ of d variables, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, at a point $\mathbf{x_0}$ is denoted $\bigtriangledown f(\mathbf{x_0})$. It is a d-dimensional vector with components $\frac{\partial f(\mathbf{x_0})}{\partial x_1}, \frac{\partial f(\mathbf{x_0})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x_0})}{\partial x_d}$, where $\frac{\partial f}{\partial x_i}$ are partial derivatives. Without explicitly stating, we assume that the derivatives referred to exist. The rate of increase of the function f as we move from $\mathbf{x_0}$ in a direction $\mathbf{u}$ is $\bigtriangledown f(\mathbf{x_0}) \cdot \mathbf{u}$. So the direction of steepest descent is $-\bigtriangledown f(\mathbf{x_0})$; this is a natural direction to move to minimize f. But by how much should we move? A large move may overshoot the minimum. See Figure 10.6. A simple fix is to minimize f on the line from $\mathbf{x_0}$ in the direction of steepest descent by solving a one dimensional minimization problem. This gives us the next iterate $\mathbf{x_1}$ and we repeat. We do not discuss the issue of step-size any further. Instead, we focus on infinitesimal gradient descent, where, the algorithm makes infinitesimal moves in the $-\bigtriangledown f(\mathbf{x_0})$ direction. Whenever $\bigtriangledown \mathbf{f}$ is not the zero vector, we strictly decrease the function in the direction $-\bigtriangledown \mathbf{f}$, so the current point is not a minimum of the function f. Conversely, a point $\mathbf{x}$ where $\bigtriangledown \mathbf{f} = \mathbf{0}$ is called a *first-order local optimum* of f. A first-order local optimum may be a local minimum, local maximum, or a saddle point. We ignore saddle points since numerical error is likely to prevent gradient descent from stoping at a saddle point. In general, local minima do not have to be global minima, see Figure 10.6, and gradient descent may converge to a local minimum that is not a global minimum. When the function f is convex, this is not the case.

A function f of a single variable x is said to be convex if for any two points a and b, the line joining $f(a)$ and $f(b)$ is above the curve $f(\cdot)$. A function of many variables is convex if on any line segment in its domain, it acts as a convex function of one variable on the line segment.

Definition 10.1 *A function f over a convex domain is a convex function if for any two points $\mathbf{x}$ and $\mathbf{y}$ in the domain, and any λ in $[0,1]$ we have*

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}).$$

The function is concave if the inequality is satisfied with $\geq$ instead of $\leq$.

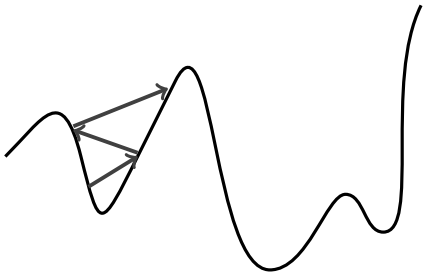

Figure 10.6: Gradient descent overshooting minimum

Theorem 10.8 *Suppose f is a convex, differentiable function defined on a closed bounded convex domain. Then any first-order local minimum is also a global minimum. Infinitesimal gradient descent always reaches the global minimum.*

Proof: We will prove that if $\mathbf{x}$ is a local minimum, then it must be a global minimum. If not, consider a global minimum point $\mathbf{y} \neq \mathbf{x}$. On the line joining $\mathbf{x}$ and $\mathbf{y}$, the function must not go above the line joining $f(\mathbf{x})$ and $f(\mathbf{y})$. This means for an infinitesimal $\varepsilon > 0$, moving distance ε from $\mathbf{x}$ towards $\mathbf{y}$, the function must decrease, so $\bigtriangledown \mathbf{f}(\mathbf{x})$ is not $\mathbf{0}$, contradicting the assumption that $\mathbf{x}$ is a local minimum. ∎

The second derivatives $\frac{\partial^2}{\partial x_i \partial x_j}$ form a matrix, called the Hessian, denoted $H(f(\mathbf{x}))$. The Hessian of f at $\mathbf{x}$ is a symmetric $d \times d$ matrix with ij^{th} entry $\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x})$. The second derivative of f at $\mathbf{x}$ in the direction $\mathbf{u}$ is the rate of change of the first derivative in the direction $\mathbf{u}$ from $\mathbf{x}$. It is easy to see that it equals

$$\mathbf{u}^T \, H(f(\mathbf{x}))\mathbf{u}.$$

To see this, note that the second derivative of f along the unit vector $\mathbf{u}$ is

$$\sum_j u_j \, \frac{\partial}{\partial x_j} \left(\bigtriangledown f(\mathbf{x}) \cdot \mathbf{u} \right) = \sum_j u_j \sum_i \frac{\partial}{\partial x_j} \left(u_i \frac{\partial f(\mathbf{x})}{\partial x_i} \right)$$
$$= \sum_{j,i} u_j u_i \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i}.$$

Theorem 10.9 *Suppose f is a function from a closed convex domain D in $\mathbf{R}^d$ to the reals and the Hessian of f exists everywhere in D. Then f is convex (concave) on D if and only if the Hessian of f is positive (negative) semi-definite everywhere on D.*

Gradient descent requires the gradient to exist. But, even if the gradient is not always defined, one can minimize a convex function over a convex domain efficiently, i.e., in polynomial time. Technically, one can only find an approximate minimum and the time

depends on the error parameter as well as the presentation of the convex set. We do not go into these details. But, in principle we can minimize a convex function over a convex domain. We can also maximize a concave function over a concave domain. However, in general, we do not have efficient procedures to maximize a convex function over a convex domain. It is easy to see that at a first-order local minimum of a possibly non-convex function, the gradient vanishes. But second-order local decrease of the function may be possible. The steepest second-order decrease is in the direction of $\pm\mathbf{v}$, where, $\mathbf{v}$ is the eigenvector of the Hessian corresponding to the largest absolute valued eigenvalue.

10.6 Linear Programming

Linear programming is an optimization problem that has been carefully studied and is immensely useful. We consider linear programming problem in the following form where A is an $m \times n$ matrix, $m \leq n$, of rank m, $\mathbf{c}$ is $1 \times n$, $\mathbf{b}$ is $m \times 1$, and $\mathbf{x}$ is $n \times 1$:

$$\max \ \mathbf{c} \cdot \mathbf{x} \quad \text{subject to} \quad A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geq 0.$$

Inequality constraints can be converted to this form by adding slack variables. Also, we can do Gaussian elimination on A and if it does not have rank m, we either find that the system of equations has no solution, whence we may stop or we can find and discard redundant equations. After this preprocessing, we may assume that A 's rows are independent.

The simplex algorithm is a classical method to solve linear programming problems. It is a vast subject and is well discussed in many texts. Here, we will discuss the ellipsoid algorithm which is in a sense based more on continuous mathematics and is closer to the spirit of this book.

10.6.1 The Ellipsoid Algorithm

The first polynomial time algorithm for linear programming[45] was developed by Khachiyan based on work of Iudin, Nemirovsky and Shor and is called the ellipsoid algorithm. The algorithm is best stated for the seemingly simpler problem of determining whether there is a solution to $A\mathbf{x} \leq \mathbf{b}$ and if so finding one. The ellipsoid algorithm starts with a large ball in d-space which is guaranteed to contain the polyhedron $A\mathbf{x} \leq \mathbf{b}$. Even though we do not yet know if the polyhedron is empty or non-empty, such a ball can be found. The algorithm checks if the center of the ball is in the polyhedron, if it is, we have achieved our objective. If not, we know from the Separating Hyperplane Theorem of convex geometry that there is a hyperplane called the separating hyperplane through the center of the ball

[45] Although there are examples where the simplex algorithm requires exponential time, it was shown by Shanghua Teng and Dan Spielman that the expected running time of the simplex algorithm on an instance produced by taking an arbitrary instance and then adding small Gaussian perturbations to it is polynomial.