Numerical Optimization Review

Akhil Vasvani

March 2019

1 Questions

Exercise 1. What is underflow and overflow?

Proof. One major difficulty in performing continuous math on a digital computer is the need to represent infinitely many real numbers with a finite number of bit patterns. So this means that for almost all real numbers, there will be some approximation error when representing the number in the computer (rounding error). Rounding error compounds across many operations and can cause algorithms to fail because they are not designed to minimize the accumulation of rounding error.

One form of rounding error that is particularly devastating is **underflow**. Underflow occurs when numbers near zero are rounded to zero. Many functions behave qualitatively differently when their argument is zero rather than a small positive number. For example, we usually want to avoid division by zero (some software environments will raise exceptions when this occurs, others will return a result with a placeholder not-a-number value) or taking the logarithm of zero (this is usually treated as $-\infty$, which then becomes not-a-number if it is used for many further arithmetic operations).

Another highly damaging form of numerical error is **overflow**. Overflow occurs when numbers with large magnitude are approximated as ∞ or $-\infty$. Further arithmetic will usually change these infinite values into not-a-number values. □

Exercise 2. How to tackle the problem of underflow or overflow for softmax function or log softmax function?

Proof. One example of a function that must be stabilized against underflow and overflow is the softmax function. The softmax function is often used to predict the probabilities associated with a multinoulli distribution. The softmax function is defined to be:

$$\operatorname{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}. \tag{1}$$

For the softmax function, when all of the x_i are equal to some constant c— if c is very negative, then $\exp(c)$ will underflow. This means the denominator of the softmax will become 0, so the final result is undefined. When c is very large and positive, $\exp(c)$ will overflow, again resulting in the expression as a whole being undefined.

Both of these difficulties can be resolved by instead evaluating softmax($\boldsymbol{z}$) where $\boldsymbol{z} = \boldsymbol{x} - \max_i x_i$. Simple algebra shows that the value of the softmax function is not changed analytically by adding or subtracting a scalar from the input vector. Subtracting $\max_i x_i$ results in the largest argument to exp being 0, which rules out the possibility of overflow. Likewise, at least one term in the denominator has a value of 1, which rules out the possibility of underflow in the denominator leading to a division by zero.

However, underflow in the numerator can still cause the expression as a whole to evaluate to zero. This means that if we implement log softmax(x) by first running the softmax subroutine then passing the result to the log function, we could erroneously obtain $-\infty$. Instead, we must implement a separate function that calculates log softmax in a numerically stable way. The log softmax function can be stabilized using the same trick as we used to stabilize the softmax function.

Example. Softmax Trick:

$$\pi_i = \frac{\exp(x_i - b)}{\sum_{j=1}^{n} \exp(x_i - b)}, \qquad b = \max_{i=1}^{n} x_i \tag{2}$$

Log Softmax Trick:

$$\log \pi_i = x_i - \Big(b + \log \sum_{j=1}^{n} \exp(x_j - b)\Big), \qquad b = \max_{i=1}^{n} x_i \tag{3}$$

□

Exercise 3. What is the condition number?

Proof. Consider the function $f(x) = \boldsymbol{A}^{-1}\mathbf{x}$. When $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ has an eigenvalue decomposition, its **condition number** is:

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right| \tag{4}$$

This is the ratio of the magnitude of the largest and smallest eigenvalue. When this number is large, matrix inversion is particularly sensitive to error in the input. This sensitivity is an intrinsic property of the matrix itself, not the result of rounding error during matrix inversion. □

Exercise 4. What is poor conditioning?

Proof. Conditioning refers to how rapidly a function changes with respect to small changes in its inputs. Functions that change rapidly when their inputs are perturbed slightly can be problematic for scientific computation because rounding errors in the inputs can result in large changes in the output.

When the condition number is very high, then optimization performs very poorly which is **poor conditioning**. Poorly conditioned matrices amplify preexisting errors when we multiply by the true matrix inverse. □

Exercise 5. What are grad, div, and curl?

Proof. The partial derivative $\frac{\partial}{\partial x_i} f(\boldsymbol{x})$ measures how f changes as only the variable x_i increases at point $\boldsymbol{x}$. The **gradient** (grad) generalizes the notion of derivative to the case where the derivative is with respect to a vector: the gradient of f is the vector containing all of the partial derivatives, denoted $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. Element i of the gradient is the partial derivative of f with respect to x_i.

Divergence of a vector indicates how much the vector spreads out from the certain point. Imagine water coming from a faucet. A point or region with positive divergence is often referred to as a "source", while a point or region with negative divergence is a "sink". It is denoted as $\nabla \cdot f$.

Curl of a vector indicates that how much the vector curls or twists around a point. Imagine rotating water in a bucket. The water "curls" in a clockwise or counterclockwise direction. One can measure the curl by putting a piece of dust in the liquid and seeing if it spins around its own axis. It is denoted by $\nabla \times f$.

Example. Say $f(x, y, z) = xy^2 - yz$

$\boldsymbol{\nabla}$ (The gradient) $= \frac{\partial}{\partial x}\hat{\mathbf{i}} + \frac{\partial}{\partial y}\hat{\mathbf{j}} + \frac{\partial}{\partial z}\hat{\mathbf{k}}$, so

$$\boldsymbol{\nabla}_x = \frac{\partial f}{\partial x} = y^2$$

$$\boldsymbol{\nabla}_y = \frac{\partial f}{\partial y} = 2xy - z$$

$$\boldsymbol{\nabla}_z = \frac{\partial f}{\partial y} = -y$$

,

$$\Rightarrow \boldsymbol{\nabla} f = y^2\hat{\mathbf{i}} + (2xy - z)\hat{\mathbf{j}} - y\hat{\mathbf{k}}.$$

Say $F(x, y) = \frac{x}{y}\hat{\mathbf{i}} + (2x - 3y)\hat{\mathbf{j}}$

$\boldsymbol{\nabla} \cdot F(x, y)$ (the divergence) $= \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} = \frac{1}{y} - 3 = \frac{1-3y}{y}$.

Let's say $F(x, y, z) = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}$

$\boldsymbol{\nabla} \times F(x, y, z)$ (the curl)

$$= \begin{vmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ x & y & z \end{vmatrix} = \hat{\mathbf{i}}(\frac{\partial}{\partial y}(z) - \frac{\partial}{\partial z}(y)) - \hat{\mathbf{j}}(\frac{\partial}{\partial x}(z) - \frac{\partial}{\partial z}x) + \hat{\mathbf{k}}(\frac{\partial}{\partial x}y - \frac{\partial}{\partial y}x) = 0.$$

□

Exercise 6. What are critical or stationary points in multi-dimensions?

Proof. In multi-dimensions, **critical points** or **stationary points** are points where every element of the gradient is equal to 0.

Note, critical points are either minima or maxima, which can be local or global. The stationary point or saddle point is neither a maxima or minima.

□

Exercise 7. Why should you do gradient descent when you want to minimize a function?

Proof. The derivative is useful for minimizing a function because it tells us how to change x in order to make a small improvement in $f(x)$. For example, we know that $f(x - \epsilon \text{sign}(f'(x)))$ is less than $f(x)$ for small enough ϵ. We can thus reduce $f(x)$ by moving x in small steps with opposite sign of the derivative. This technique is called **gradient descent**.

In terms of vector calculus, the answer still remains the same (minimizing a function because it tells us how to change the input in order to make a small improvement in the output). In order to understand how we arrive at the same conclusion, let's lay out some definitions:

The directional derivative in direction u (a unit vector) is the slope of the function f in direction u. In other words, the directional derivative is the derivative of the function $f(\boldsymbol{x} + \alpha \boldsymbol{u})$ with respect to α, evaluated at $\alpha = 0$. Using the chain rule, we can see that $\frac{\partial}{\partial \alpha} f(\boldsymbol{x} + \alpha \boldsymbol{u})$ evaluates to $u^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ when $\alpha = 0$.

To minimize f, we would like to find the direction in which f decreases the fastest. We can do this using the directional derivative:

$$\min_{\boldsymbol{u}, \boldsymbol{u}^\top \boldsymbol{u} = 1} \boldsymbol{u}^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$= \min_{\boldsymbol{u}, \boldsymbol{u}^\top \boldsymbol{u} = 1} ||\boldsymbol{u}||_2 ||\nabla_{\boldsymbol{x}} f(\boldsymbol{x})||_2 \cos(\theta)$$

where θ is the angle between u and the gradient. Substituting in $||u||_2 = 1$ and ignoring factors that do not depend on u, this simplifies to $\min u \cos \theta$. This is minimized when u points in the opposite direction as the gradient. In other words, the gradient points directly uphill, and the negative gradient points directly downhill. We can decrease f by moving in the direction of the negative gradient. This is known as the method of steepest descent or gradient descent.

See Figure 1.

□

Exercise 8. What is line search?

Proof. The **line search** is a strategy which evaluates $f(\boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}))$ for several values of ϵ and chooses the one that results in the smallest object function value.

Note, ϵ is the learning rate, a positive scalar determining the size of each step. □

Exercise 9. What is hill climbing?

Proof. Although gradient descent is limited to optimization in continuous spaces, the general concept of repeatedly making a small move (that is approximately the best small move) towards better configurations can be generalized to discrete spaces. Ascending an objective function of discrete parameters is called **hill climbing**. □

Exercise 10. What is a Jacobian matrix?

Proof. The matrix containing all such partial derivatives is known as a **Jacobian matrix**. Specifically, if we have a function $f\colon \mathbb{R}^m \to \mathbb{R}^n$, then the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{n \times m}$ of $\boldsymbol{f}$ is defined such that $J_{i,j} = \frac{\partial}{\partial x_j} f(\boldsymbol{x})_i$.

Example. $f(x,y,z) = (xy + 2yz, 2xy^2z) \to \mathbf{f} = (f_1, f_2)$

$$\Rightarrow f_1(x,y,z) = xy + 2yz,$$
$$\Rightarrow f_2(x,y,z) = 2xy^2z$$
$$\text{Jacobian Matrix: } \boldsymbol{D}f(c) = \begin{bmatrix} \boldsymbol{D_1}f_1(c) & \boldsymbol{D_2}f_1(c) & \dots & \boldsymbol{D_n}f_1(c) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \boldsymbol{D_1}f_m(c) & \boldsymbol{D_2}f_m(c) & \dots & \boldsymbol{D_n}f_m(c) \end{bmatrix}$$

$$\nabla f_1(x,y,z) = \Big[\boldsymbol{D_1}f_1(x,y,z),\ \boldsymbol{D_2}f_1(x,y,z),\ \boldsymbol{D_3}f_1(x,y,z)\Big] = \Big[\frac{\partial f_1}{\partial x},\ \frac{\partial f_1}{\partial y},\ \frac{\partial f_1}{\partial z}\Big]$$
$$= [y,\ x + 2z,\ 2y]$$

$$\nabla f_2(x,y,z) = \Big[\frac{\partial f_2}{\partial x},\ \frac{\partial f_2}{\partial y},\ \frac{\partial f_2}{\partial z}\Big] = [2y^2z,\ 4xyz,\ 2xy^2]$$

$$\Rightarrow \boldsymbol{D}f(x,y,z) = \begin{bmatrix} y & x+2z & 2y \\ 2y^2z & 4xyz & 2xy^2 \end{bmatrix},$$

defined $\forall x, y, z$

□

Exercise 11. What is curvature?

Proof. The second derivative measures **curvature**—how the first derivative will change as we vary the input. This is important because it tells us whether a gradient step will cause as much of an improvement as we would expect based on the gradient alone.

Suppose we have a quadratic function (many functions that arise in practice are not quadratic but can be approximated well as quadratic, at least locally). If such a function has a second derivative of zero, then there is no curvature. It is a perfectly flat line, and its value can be predicted using only the gradient. If the gradient is 1, then we can make a step of size ϵ along the negative gradient, and the cost function will decrease by ϵ. If the second derivative is negative, the function curves downward, so the cost function will actually decrease by more than ϵ. Finally, if the second derivative is positive, the function curves upward, so the cost function can decrease by less than ϵ.

Example. See Figure 2.

□

Exercise 12. What is a Hessian matrix?

Proof. When our function has multiple input dimensions, there are many second derivatives. These derivatives can be collected together into a matrix called the **Hessian matrix**. The Hessian matrix $\boldsymbol{H}(f)(\boldsymbol{x}$ is defined such that.

$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} \tag{5}$$

Equivalently, the Hessian is the Jacobian of the gradient. □

Exercise 13. What is constrained optimization? What are some of the techniques involved?

Proof. Sometimes we wish not only to maximize or minimize a function $f(\boldsymbol{x})$ over all possible values of $\boldsymbol{x}$. Instead we may wish to find the maximal or minimal value of $f(\boldsymbol{x})$ for values of $\boldsymbol{x}$ in some set $\mathbb{Q}$. This is known as **constrained optimization**. Points $\boldsymbol{x}$ that lie within the set $\mathbb{Q}$ are called **feasible** points in constrained optimization terminology.

The **Karush–Kuhn–Tucker** (KKT) approach provides a very general solution to constrained optimization. With the KKT approach, we introduce a new function called the **generalized Lagrangian** or **generalized Lagrange function**. To define the Lagrangian, we first need to describe $\mathbb{Q}$ in terms of equations and inequalities. We want a description of $\mathbb{Q}$ in terms of m functions $g^{(i)}$ and n functions $h^{(j)}$ so that $\mathbb{Q} = \{\boldsymbol{x} \,|\, \forall i, g^{(i)}(\boldsymbol{x}) = 0$ and $\forall j, h^{(j)}(\boldsymbol{x}) \leq 0\}$. The equations involving $g^{(i)}$ are called the **equality constraints** and the inequalities involving $h^{(j)}$ are called **inequality constraints**.

We introduce new variables λ_i and α_j for each constraint, these are called the KKT multipliers. The generalized Lagrangian is then defined as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}). \tag{6}$$

We can now solve a constrained minimization problem using unconstrained optimization of the generalized Lagrangian. Observe that, so long as at least one feasible point exists and $f(\boldsymbol{x})$ is not permitted to have value ∞, then

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq \mathbf{0}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}). \tag{7}$$

has the same optimal objective function value and set of optimal points $\boldsymbol{x}$ as

$$\min_{\boldsymbol{x} \in \mathbb{Q}} f(\boldsymbol{x}). \tag{8}$$

This follows because any time the constraints are satisfied,

$$\max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq \mathbf{0}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}), \tag{9}$$

while any time a constraint is violated,

$$\max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq \mathbf{0}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \infty. \tag{10}$$

A simple set of properties describe the optimal points of constrained optimization problems. These properties are called the Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951). They are necessary conditions, but not always sufficient conditions, for a point to be optimal. The conditions are:

- The gradient of the generalized Lagrangian is zero.
- All constraints on both $\boldsymbol{x}$ and the KKT multipliers are satisfied.
- The inequality constraints exhibit "complementary slackness": $\boldsymbol{\alpha} \odot \boldsymbol{h}(\boldsymbol{x}) = 0$.

Example. Suppose you are running a factory, producing some sort of widget that requires steel as a raw material. Your costs are predominantly human labor, which is \$20 per hour for your workers, and the steel itself, which runs for \$170 per ton. Suppose your revenue R is loosely modeled by the following equation:

$$R(h,s) = 200h^{\frac{2}{3}}s^{\frac{1}{3}},$$

where h represents hours of labor and s represents tons of steel. If your budget is b what is the maximum possible revenue M?

Let's start by writing the Lagrangian $\mathcal{L}(h,s,\lambda)$ based on the function $R(h,s)$ and the constraint $20h + 170s = b$:

$$\mathcal{L}(h,s,\lambda,b) = 200h^{\frac{2}{3}}s^{\frac{1}{3}} - \lambda(20h + 170s - b).$$

Then we find the critical points of $\mathcal{L}$, meaning the solutions to

$$\nabla\mathcal{L}(h,s,\lambda,b) = 0$$

$$\Rightarrow \nabla\mathcal{L}(h,s,\lambda,b) = \left(\frac{400}{3}\left(\frac{s}{h}\right)^{\frac{1}{3}} - 20\lambda, \frac{200}{3}\left(\frac{h}{s}\right)^{\frac{2}{3}} - 170\lambda, -20h - 170s + b, \lambda\right) = 0$$

$$\rightarrow \nabla_\lambda\mathcal{L} = -20h - 170s + b = 0$$

$$\rightarrow \nabla_h\mathcal{L} = \frac{400}{3}\left(\frac{s}{h}\right)^{\frac{1}{3}} - 20\lambda = 0$$

Let's call $u = \left(\frac{h}{s}\right)^{\frac{1}{3}}$, so

$$\nabla_h\mathcal{L} = \frac{400}{3}(u)^{-1} - 20\lambda = 0 \rightarrow \frac{40}{3}u^{-1} = 2\lambda \rightarrow \lambda = \frac{20}{3}u^{-1}$$

$$\rightarrow \nabla_s\mathcal{L} = \frac{200}{3}\left(\frac{h}{s}\right)^{\frac{2}{3}} - 170\lambda = 0 \rightarrow \frac{200}{3}u^2 - 170\lambda = 0 \rightarrow \frac{20}{3}u^2 - 17\left(\frac{20}{3}u^{-1}\right) = 0$$

$$\Rightarrow u^2 - 17u^{-1} = 0 \rightarrow u^3 = 17 \rightarrow \left(\left(\frac{h}{s}\right)^{\frac{1}{3}}\right)^3 = 17 \rightarrow \frac{h}{s} = 17 \rightarrow s = \frac{h}{17}.$$

$$\rightarrow \nabla_b\mathcal{L} = \lambda = 0$$

(This doesn't tell us anything except to minimize the constraint!)

$$\nabla_\lambda\mathcal{L} = -20h - 170s + b = 0 \rightarrow -20h - 170\left(\frac{h}{17}\right) + b = 0 \rightarrow b = 30h \rightarrow h = \frac{b}{30}.$$

$$\nabla_\lambda\mathcal{L} = -20h - 170s + b = 0 \rightarrow -20(17s) - 170s + b = 0 \rightarrow b = 510s \rightarrow s = \frac{b}{510}.$$

$$\Rightarrow \lambda = \frac{20}{3}\left(\frac{\frac{b}{30}}{\frac{b}{510}}\right)^{-\frac{1}{3}} = \frac{20}{3}\left(\frac{b}{30}\cdot\frac{510}{b}\right)^{-\frac{1}{3}} = \frac{20}{3}\left(17\right)^{-\frac{1}{3}} \approx 2.593.$$

Therefore, we found that \$2.59, it would mean each additional dollar you spend over your budget would yield another \$2.59, in revenue. Conversely, decreasing your budget by a dollar will cost you that much in lost revenue.

□

Descent.png

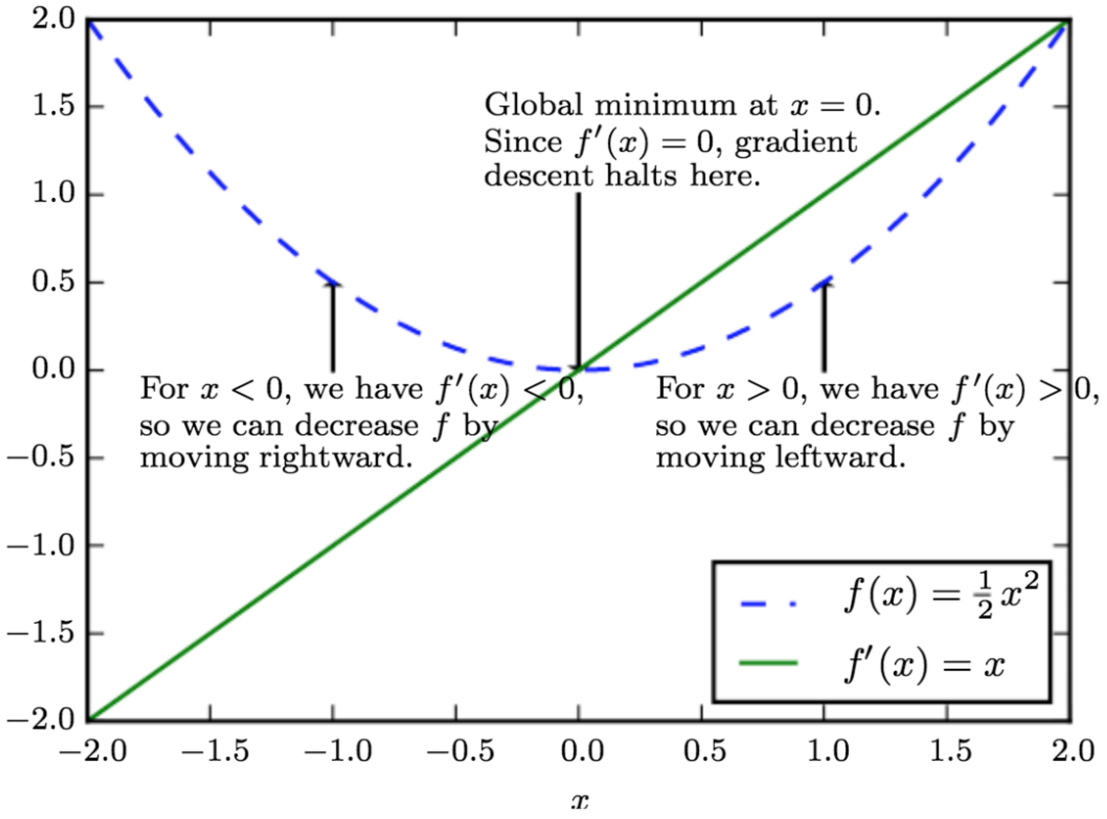

Figure 1: An illustration of how Gradient descent algorithm uses the derivatives of a function can be used to follow the function downhill to a minimum.

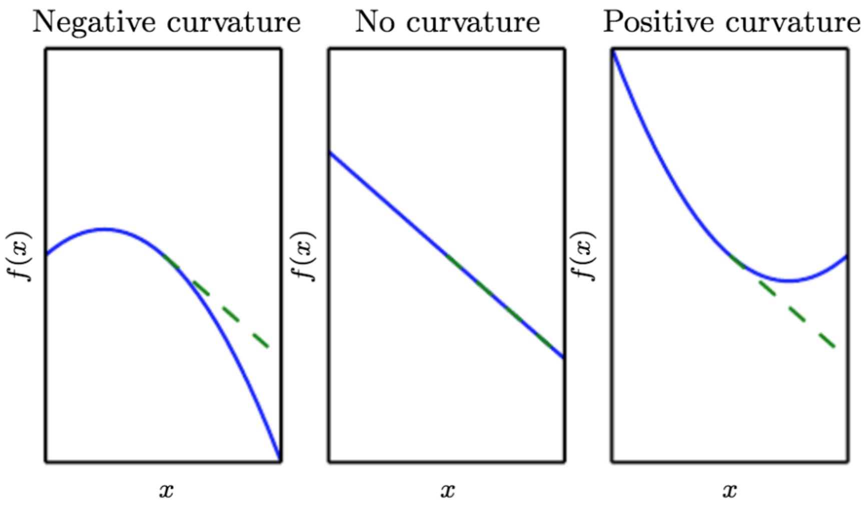

Figure 2: The second derivative determines the curvature of a function. Here we show quadratic functions with various curvature. The dashed line indicates the value of the cost function we would expect based on the gradient information alone as we make a gradient step downhill. In the case of negative curvature, the cost function actually decreases faster than the gradient predicts. In the case of no curvature, the gradient predicts the decrease correctly. In the case of positive curvature, the function decreases slower than expected and eventually begins to increase, so steps that are too large can actually increase the function inadvertently.