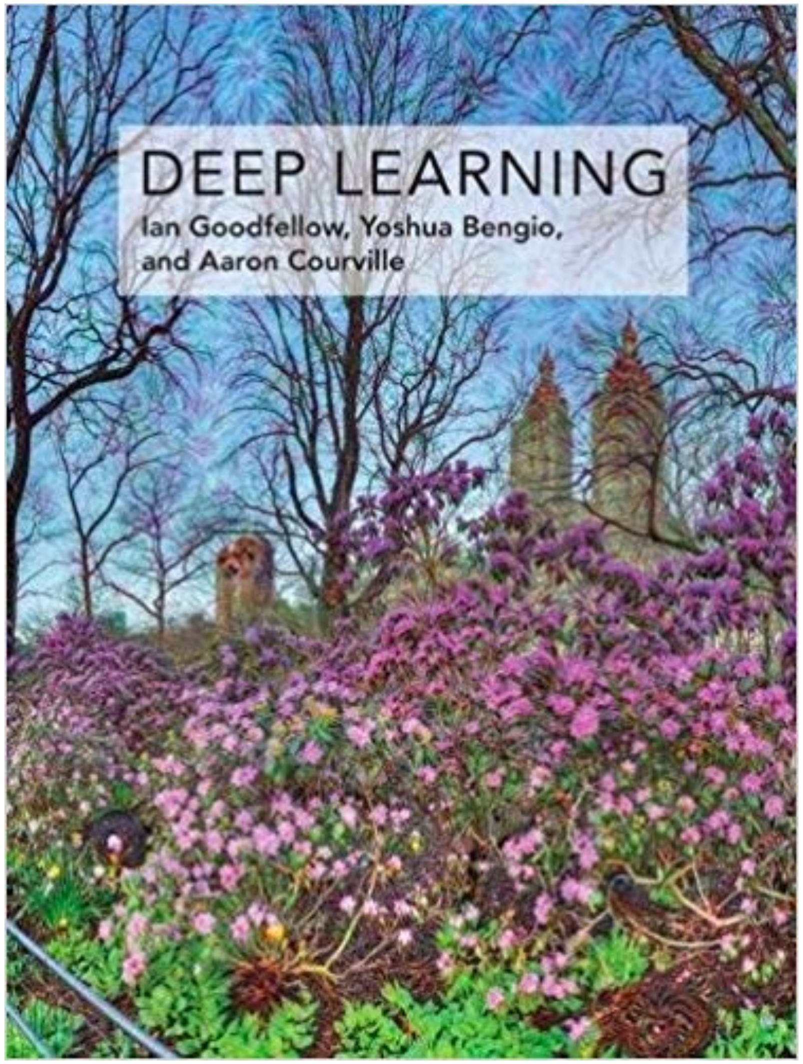
國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Deep Learning
深度學習
Fall 2023

Gradient-based Optimization
(Chapter 4.3-4.5,5.9)

Prof. Chia-Han Lee
李佳翰 教授



- Figure source: Textbook and Internet
- You are encouraged to buy the textbook.
- Please respect the copyright of the textbook. Do not distribute the materials to other people.

Gradient-based optimization

- Optimization refers to the task of either minimizing or maximizing some function $f(\boldsymbol{x})$ by altering $\boldsymbol{x}$. We usually phrase most optimization problems in terms of minimizing $f(\boldsymbol{x})$. Maximization may be accomplished via a minimization algorithm by minimizing $-f(\boldsymbol{x})$.

- The function we want to minimize or maximize is called the objective function, or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function.

Gradient-based optimization

- We often denote the value that minimizes or maximizes a function with a superscript $*$. For example, we might say $\boldsymbol{x}^* = \arg\min f(\boldsymbol{x})$.

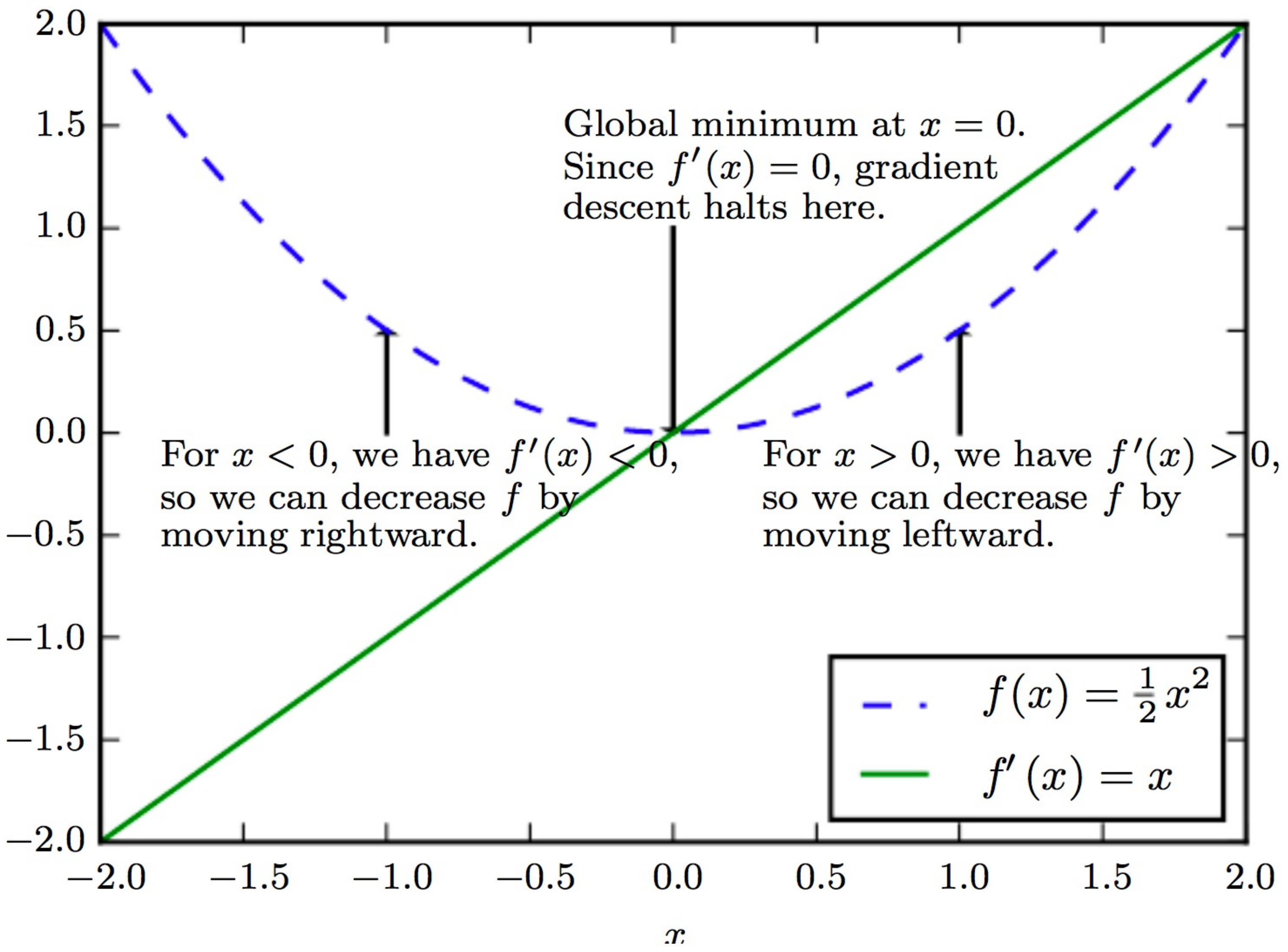
- Suppose we have a function $y = f(x)$, where both x and y are real numbers. The derivative of this function is denoted as $f'(x)$ or as $\frac{dy}{dx}$. The derivative $f'(x)$ gives the slope of $f(x)$ at the point x. It specifies how to scale a small change in the input to obtain the corresponding change in the output: $f(x + \epsilon) \approx f(x) + \epsilon f'(x)$.

Gradient-based optimization

- The derivative is therefore useful for minimizing a function because it tells us how to change x in order to make a small improvement in y.

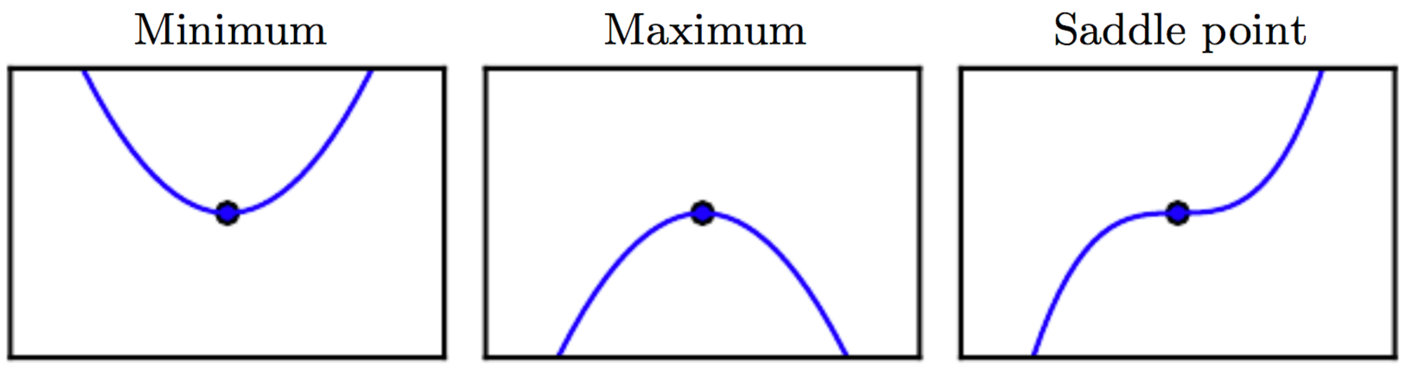
- For example, we know that $f(x - \epsilon\,\mathrm{sign}(f'(x)))$ is less than $f(x)$ for small enough ϵ. We can thus reduce $f(x)$ by moving x in small steps with the opposite sign of the derivative. This technique is called gradient descent.

Gradient-based optimization



Gradient-based optimization

- When $f'(x) = 0$, the derivative provides no information about which direction to move. Points where $f'(x) = 0$ are known as critical points, or stationary points.
- A local minimum is a point where $f(x)$ is lower than at all neighboring points, so it is no longer possible to decrease $f(x)$ by making infinitesimal steps.
- A local maximum is a point where $f(x)$ is higher than at all neighboring points, so it is not possible to increase $f(x)$ by making infinitesimal steps.
- Some critical points are neither maxima nor minima. These are known as saddle points.



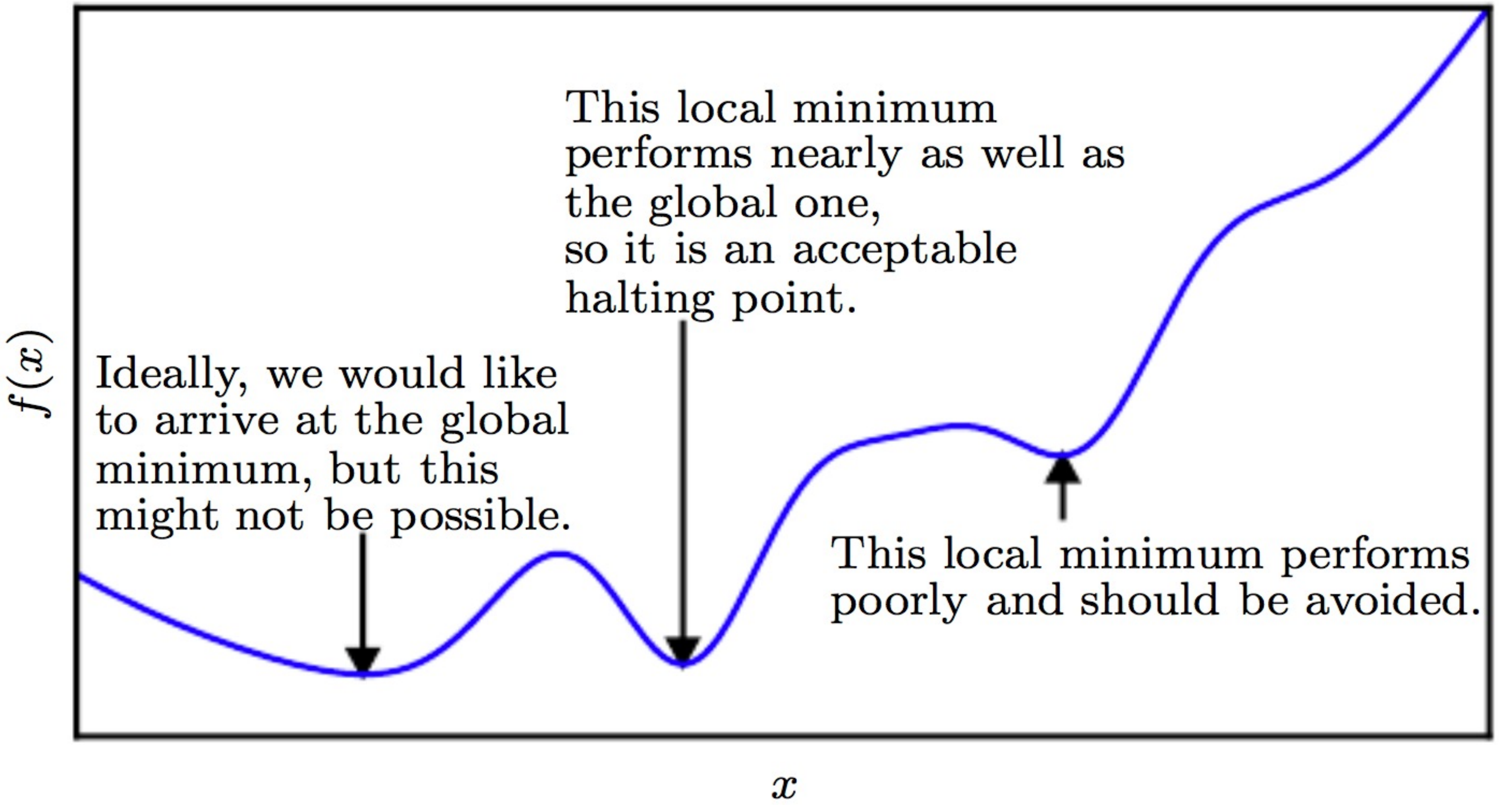
Gradient-based optimization

- A point that obtains the absolute lowest value of $f(x)$ is a global minimum. There can be only one global minimum or multiple global minima of the function. It is also possible for there to be local minima that are not globally optimal.

- In the context of deep learning, we optimize functions that may have many local minima that are not optimal and many saddle points surrounded by very flat regions. All of this makes optimization difficult, especially when the input to the function is multidimensional.

- We therefore usually settle for finding a value of f that is very low but not necessarily minimal in any formal sense.

Gradient-based optimization



Gradient-based optimization

- For functions with multiple inputs, we must make use of the concept of partial derivatives. 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 of f is the vector containing all 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.

- In multiple dimensions, critical points are points where every element of the gradient is equal to zero.

Gradient-based optimization

- The directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$. 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 $\boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ 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:

$$\min_{\boldsymbol{u}, \boldsymbol{u}^\top \boldsymbol{u}=1} \boldsymbol{u}^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.3}$$

$$= \min_{\boldsymbol{u}, \boldsymbol{u}^\top \boldsymbol{u}=1} ||\boldsymbol{u}||_2 ||\nabla_{\boldsymbol{x}} f(\boldsymbol{x})||_2 \cos \theta \tag{4.4}$$

where θ is the angle between $\boldsymbol{u}$ and the gradient.

Gradient-based optimization

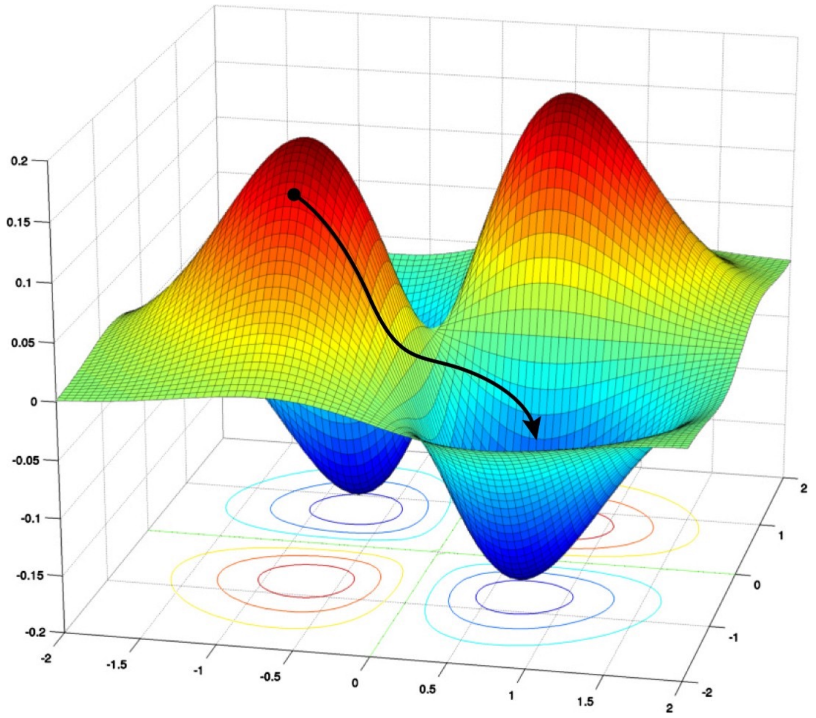
- Substituting in $||\boldsymbol{u}||_2 = 1$ and ignoring factors that do not depend on $\boldsymbol{u}$, this simplifies to $\min_{\boldsymbol{u}} \cos\theta$. This is minimized when u points in the opposite direction as the gradient. In other words, the gradient points directly uphill, and the negative 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.

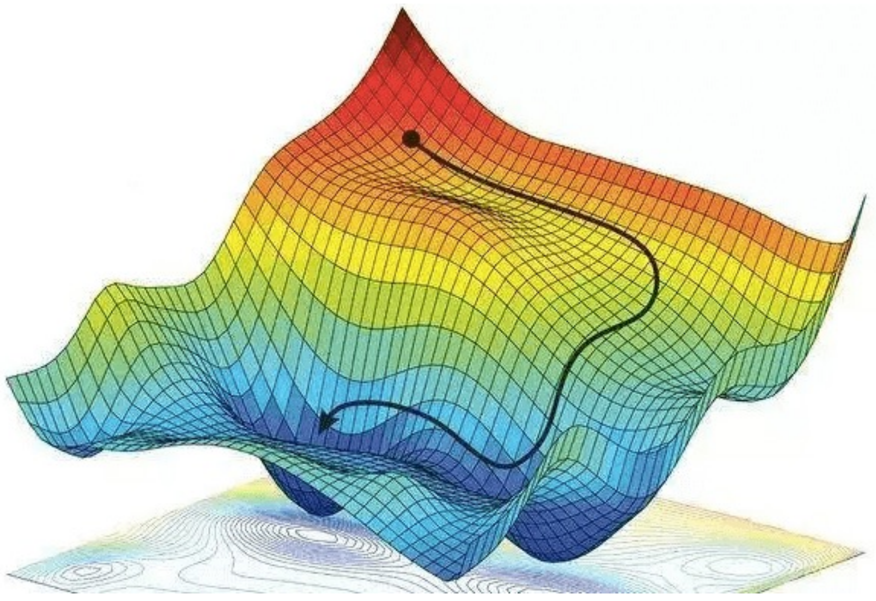
- Steepest descent proposes a new point

$$\boldsymbol{x}' = \boldsymbol{x} - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.5}$$

where ϵ is the learning rate, a positive scalar determining the size of the step.

Gradient-based optimization





- https://faun.pub/what-the-hell-is-gradient-descent-97a75cc5cc4e
- https://easyai.tech/en/ai-definition/gradient-descent/

Gradient-based optimization

- We can choose ϵ in several different ways. A popular approach is to set ϵ to a small constant. Sometimes, we can solve for the step size that makes the directional derivative vanish.

- Another approach is to evaluate $f(\boldsymbol{x} - \epsilon\nabla_{\boldsymbol{x}} f(\boldsymbol{x}))$ for several values of ϵ and choose the one that results in the smallest objective function value. This last strategy is called a line search.

Gradient-based optimization

- Steepest descent converges when every element of the gradient is zero (or, in practice, very close to zero).

- In some cases, we may be able to avoid running this iterative algorithm and just jump directly to the critical point by solving the equation $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$ for $\boldsymbol{x}$.

- 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) toward better configurations can be generalized to discrete spaces. Ascending an objective function of discrete parameters is called hill climbing.

Jacobian and Hessian matrices

- Sometimes we need to find all the partial derivatives of a function whose input and output are both vectors. The matrix containing all such partial derivatives is known as a Jacobian matrix. Specifically, if we have a function $\boldsymbol{f}: \mathbb{R}^m \to \mathbb{R}^n$, then the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{n \times m}$ of $\boldsymbol{f}$ is defined such that $J_{i,j} = \frac{\partial}{\partial x_j} f(\boldsymbol{x})_i$.

- We are also sometimes interested in a derivative of a derivative. This is known as a second derivative. For example, for a function $f: \mathbb{R}^n \to \mathbb{R}$, the derivative with respect to x_i of the derivative of f with respect to x_j is denoted as $\frac{\partial^2}{\partial x_i \partial x_j} f$.

Jacobian and Hessian matrices

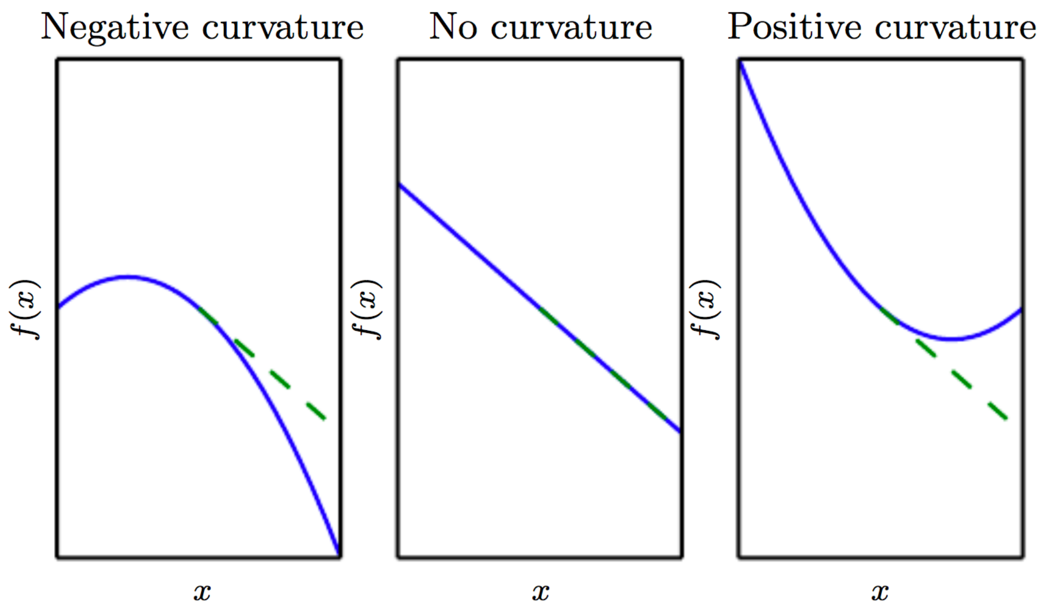
- In a single dimension, we can denote $\frac{d^2}{dx^2}f$ by $f''(x)$. The second derivative tells us how the first derivative will change as we vary the input. We can think of the second derivative as measuring curvature.

- 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.

Jacobian and Hessian matrices

- If the second derivative is negative, the function curves downward, so the cost function will actually decrease by more than ϵ.

- If the second derivative is positive, the function curves upward, so the cost function can decrease by less than ϵ.



Jacobian and Hessian matrices

- 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

$$\boldsymbol{H}(f)(\boldsymbol{x})_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}). \tag{4.6}$$

  Equivalently, the Hessian is the Jacobian of the gradient.
- Anywhere that the second partial derivatives are continuous, the differential operators are commutative; that is, their order can be swapped:

$$\frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x}) = \frac{\partial^2}{\partial x_j \partial x_i} f(\boldsymbol{x}). \tag{4.7}$$

- This implies that $\boldsymbol{H}_{i,j} = \boldsymbol{H}_{j,i}$, so the Hessian matrix is symmetric at such points. Most of the functions we encounter in the context of deep learning have a symmetric Hessian almost everywhere.

Jacobian and Hessian matrices

- Because the Hessian matrix is real and symmetric, we can decompose it into a set of real eigenvalues and an orthogonal basis of eigenvectors. The second derivative in a specific direction represented by a unit vector $\boldsymbol{d}$ is given by $\boldsymbol{d}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{d}$.

- When $\boldsymbol{d}$ is an eigenvector of $\boldsymbol{H}$, the second derivative in that direction is given by the corresponding eigenvalue. For other directions of $\boldsymbol{d}$, the directional second derivative is a weighted average of all the eigenvalues, with weights between 0 and 1, and eigenvectors that have a smaller angle with $\boldsymbol{d}$ receiving more weight.

- The maximum eigenvalue determines the maximum second derivative, and the minimum eigenvalue determines the minimum second derivative.

Jacobian and Hessian matrices

- The (directional) second derivative tells us how well we can expect a gradient descent step to perform. We can make a second-order Taylor series approximation to the function $f(\boldsymbol{x})$ around the current point $\boldsymbol{x}^{(0)}$:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)}), \tag{4.8}$$

where $\boldsymbol{g}$ is the gradient and $\boldsymbol{H}$ is the Hessian at $\boldsymbol{x}^{(0)}$.

- If we use a learning rate of ϵ, then the new point $\boldsymbol{x}$ will be given by $\boldsymbol{x}^{(0)} - \epsilon\boldsymbol{g}$. Substituting this into our approximation, we obtain

$$f(\boldsymbol{x}^{(0)} - \epsilon\boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon\boldsymbol{g}^\top\boldsymbol{g} + \frac{1}{2}\epsilon^2\boldsymbol{g}^\top\boldsymbol{H}\boldsymbol{g}. \tag{4.9}$$

Jacobian and Hessian matrices

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}. \tag{4.9}$$

- There are three terms here: the original value of the function, the expected improvement due to the slope of the function, and the correction we must apply to account for the curvature of the function.

- When the last term is too large, the gradient descent step can actually move uphill.

- When $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{g}$ is zero or negative, the Taylor series approximation predicts that increasing ϵ forever will decrease f forever.

Jacobian and Hessian matrices

- In practice, the Taylor series is unlikely to remain accurate for large ϵ, so one must resort to more heuristic choices of ϵ in this case. When $\boldsymbol{g}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{g}$ is positive, solving for the optimal step size that decreases the Taylor series approximation of the function the most yields

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H}\boldsymbol{g}}. \tag{4.10}$$

- In the worst case, when $\boldsymbol{g}$ aligns with the eigenvector of $\boldsymbol{H}$ corresponding to the maximal eigenvalue $\lambda_{\max}$, then this optimal step size is given by $\frac{1}{\lambda_{\max}}$. To the extent that the function we minimize can be approximated well by a quadratic function, the eigenvalues of the Hessian thus determine the scale of the learning rate.

Jacobian and Hessian matrices

- The second derivative can be used to determine whether a critical point is a local maximum, a local minimum, or a saddle point. Recall that on a critical point, $f'(x) = 0$.
- When the second derivative $f''(x) > 0$, the first derivative $f'(x)$ increases as we move to the right and decreases as we move to the left. This means $f'(x - \epsilon) < 0$ and $f'(x + \epsilon) > 0$ for small enough ϵ. In other words, as we move right, the slope begins to point uphill to the right, and as we move left, the slope begins to point uphill to the left. Thus, when $f'(x) = 0$ and $f''(x) > 0$ (known as the second derivative test), we can conclude that x is a local minimum.
- When $f'(x) = 0$ and $f''(x) < 0$, we can conclude that x is a local maximum.
- When $f''(x) = 0$, the test is inconclusive. In this case x may be a saddle point or a part of a flat region.

Jacobian and Hessian matrices

- In multiple dimensions, we need to examine all the second derivatives of the function. Using the eigendecomposition of the Hessian matrix, we can generalize the second derivative test to multiple dimensions.

- At a critical point, where $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$, we can examine the eigenvalues of the Hessian to determine whether the critical point is a local maximum, local minimum, or saddle point.

- When the Hessian is positive definite (all its eigenvalues are positive), the point is a local minimum.

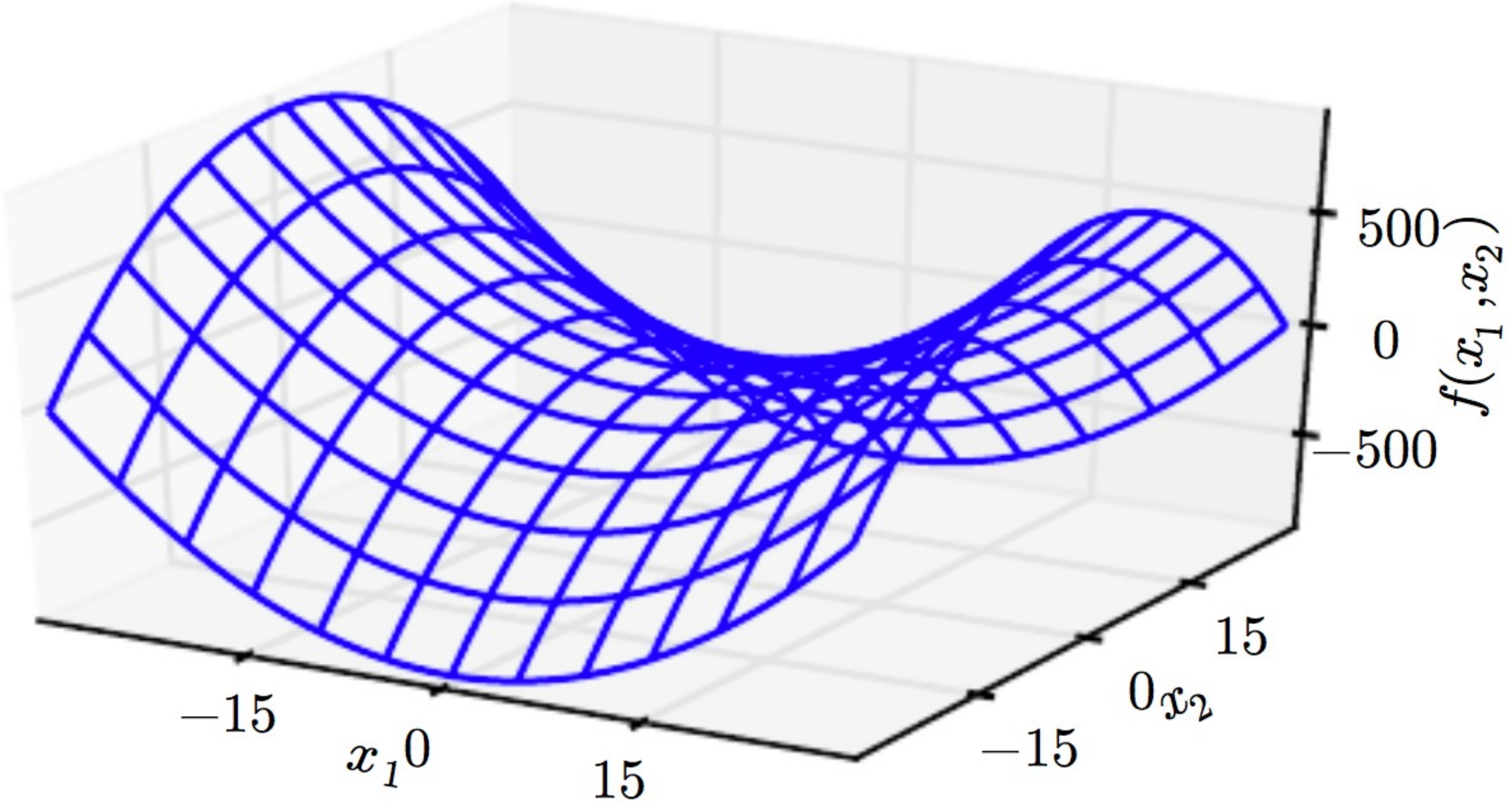
- When the Hessian is negative definite (all its eigenvalues are negative), the point is a local maximum.

Jacobian and Hessian matrices

- In multiple dimensions, it is actually possible to find positive evidence of saddle points in some cases. When at least one eigenvalue is positive and at least one eigenvalue is negative, we know that x is a local maximum on one cross section of f but a local minimum on another cross section.

- The multidimensional second derivative test is inconclusive whenever all the nonzero eigenvalues have the same sign but at least one eigenvalue is zero. This is because the univariate second derivative test is inconclusive in the cross section corresponding to the zero eigenvalue.

Jacobian and Hessian matrices



Jacobian and Hessian matrices

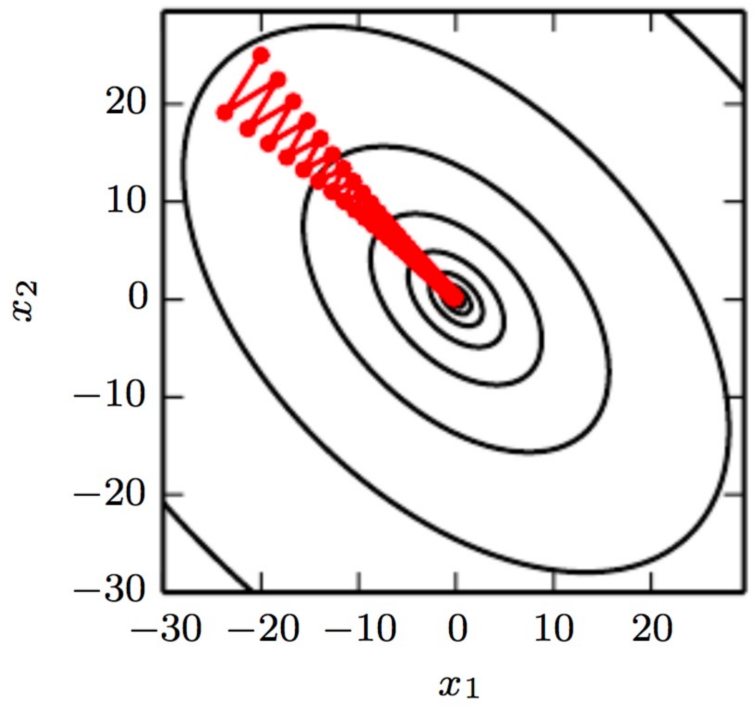
- The condition number is the ratio of the magnitude of the largest and smallest eigenvalue.

- In multiple dimensions, there is a different second derivative for each direction at a single point. The condition number of the Hessian at this point measures how much the second derivatives differ from each other.

- When the Hessian has a poor (large) condition number, gradient descent performs poorly. This is because in one direction, the derivative increases rapidly, while in another, it increases slowly. Gradient descent is unaware of this change in the derivative, so it does not know that it needs to explore preferentially in the direction where the derivative remains negative for longer.

Jacobian and Hessian matrices

- Poor condition number also makes choosing a good step size difficult. The step size must be small enough to avoid overshooting the minimum and going uphill in directions with strong positive curvature. This usually means that the step size is too small to make significant progress in other directions with less curvature.



Jacobian and Hessian matrices

- This issue can be resolved by using information from the Hessian matrix to guide the search. The simplest method for doing so is known as Newton's method. Newton's method is based on using a second-order Taylor series expansion to approximate $f(\boldsymbol{x})$ near some point $\boldsymbol{x}^{(0)}$:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x}-\boldsymbol{x}^{(0)})^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^{(0)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(0)})(\boldsymbol{x}-\boldsymbol{x}^{(0)}). \quad (4.11)$$

- If we then solve for the critical point of this function, we obtain

$$\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \boldsymbol{H}(f)(\boldsymbol{x}^{(0)})^{-1} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)}). \quad (4.12)$$

- When f is a positive definite quadratic function, Newton's method consists of applying equation 4.12 once to jump to the minimum of the function directly.

Jacobian and Hessian matrices

- When f is not truly quadratic but can be locally approximated as a positive definite quadratic, Newton's method consists of applying equation 4.12 multiple times.

- Iteratively updating the approximation and jumping to the minimum of the approximation can reach the critical point much faster than gradient descent would. This is a useful property near a local minimum, but it can be a harmful property near a saddle point.

- Newton's method is only appropriate when the nearby critical point is a minimum (all the eigenvalues of the Hessian are positive), whereas gradient descent is not attracted to saddle points unless the gradient points toward them.

Jacobian and Hessian matrices

- Optimization algorithms that use only the gradient, such as gradient descent, are called first-order optimization algorithms.

- Optimization algorithms that also use the Hessian matrix, such as Newton’s method, are called second-order optimization algorithms.

Jacobian and Hessian matrices

- Deep learning algorithms tend to lack guarantees because the family of functions used in deep learning is quite complicated.

- We sometimes gain some guarantees by restricting ourselves to functions that are either Lipschitz continuous or have Lipschitz continuous derivatives.

- A Lipschitz continuous function is a function f whose rate of change is bounded by a Lipschitz constant $\mathcal{L}$:

$$\forall \boldsymbol{x}, \forall \boldsymbol{y}, |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \mathcal{L}||\boldsymbol{x} - \boldsymbol{y}||_2. \qquad (4.13)$$

- Lipschitz continuity is a fairly weak constraint, and many optimization problems in deep learning can be made Lipschitz continuous with relatively minor modifications.

Constrained optimization

- Sometimes we may wish to find the maximal or minimal value of $f(\boldsymbol{x})$ for values of $\boldsymbol{x}$ in some set $\mathbb{S}$. This is known as constrained optimization. Points $\boldsymbol{x}$ that lie within the set $\mathbb{S}$ are called feasible points in constrained optimization terminology.

- We often wish to find a solution that is small in some sense. A common approach in such situations is to impose a norm constraint, such as $||\boldsymbol{x}|| \leq 1$.

Constrained optimization

- One simple approach to constrained optimization is simply to modify gradient descent taking the constraint into account.

- If we use a small constant step size ϵ, we can make gradient descent steps, then project the result back into $\mathbb{S}$.

- If we use a line search, we can search only over step sizes ϵ that yield new $\boldsymbol{x}$ points that are feasible, or we can project each point on the line back into the constraint region.

Constrained optimization

- A more sophisticated approach is to design a different, unconstrained optimization problem whose solution can be converted into a solution to the original, constrained optimization problem.

- For example, if we want to minimize $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^2$ with $\boldsymbol{x}$ constrained to have exactly unit L^2 norm, we can instead minimize $g(\theta) = f([\cos\theta, \sin\theta]^{\mathrm{T}})$ with respect to θ, then return $[\cos\theta, \sin\theta]$ as the solution to the original problem.

- This approach requires creativity; the transformation between optimization problems must be designed specifically for each case we encounter.

Constrained optimization

- 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 $\mathbb{S}$ in terms of equations and inequalities. We want a description of $\mathbb{S}$ in terms of m functions $g^{(i)}$ and n functions $h^{(j)}$ so that $\mathbb{S} = \{\boldsymbol{x} | \forall i, g^{(i)}(\boldsymbol{x}) = 0 \text{ and } \forall j, h^{(j)}(\boldsymbol{x}) \leq 0\}$.The equations involving $g^{(i)}$ are called the equality constraints, and the inequalities involving $h^{(j)}$ are called inequality constraints.

Constrained optimization

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

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}). \tag{4.14}$$

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

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) \tag{4.15}$$

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

$$\min_{\boldsymbol{x} \in \mathbb{S}} f(\boldsymbol{x}). \tag{4.16}$$

Constrained optimization

- This follows because any time the constraints are satisfied,

$$\max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}), \tag{4.17}$$

while any time a constraint is violated,

$$\max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \infty. \tag{4.18}$$

- These properties guarantee that no infeasible point can be optimal, and that the optimum within the feasible points is unchanged.

Constrained optimization

- To perform constrained maximization, we can construct the generalized Lagrange function of $-f(x)$, which leads to this optimization problem:

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}\geq 0} -f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}). \tag{4.19}$$

- We may also convert this to a problem with maximization in the outer loop:

$$\max_{\boldsymbol{x}} \min_{\boldsymbol{\lambda}} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}\geq 0} f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) - \sum_j \alpha_j h^{(j)}(\boldsymbol{x}). \tag{4.20}$$

- The sign of the term for the equality constraints does not matter; we may define it with addition or subtraction as we wish, because the optimization is free to choose any sign for each λ_i.

Constrained optimization

- An inequality constraint $h^{(i)}(\boldsymbol{x})$ is active if $h^{(i)}(\boldsymbol{x}^*) = 0$. Because an inactive $h^{(i)}$ has negative value, then the solution to $\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}\geq 0} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha})$ will have $\alpha_i = 0$. We can thus observe that at the solution, $\boldsymbol{\alpha} \odot \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}$. $\odot$ denotes element-wise product, or Hadamard product.

- In other words, for all i, we know that at least one of the constraints $\alpha_i \geq 0$ or $h^{(i)}(\boldsymbol{x}) \leq 0$ must be active at the solution.

- We can say that either the solution is on the boundary imposed by the inequality and we must use its KKT multiplier to influence the solution to $\boldsymbol{x}$, or the inequality has no influence on the solution and we represent this by zeroing out its KKT multiplier.

Constrained optimization

- A simple set of properties, called Karush-Kuhn-Tucker (KKT) conditions, describe the optimal points of constrained optimization problems. 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": $\boldsymbol{\alpha} \odot \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$.

Example: linear least squares

- Suppose we want to find the value of $\boldsymbol{x}$ that minimizes

$$f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2. \tag{4.21}$$

- Specialized linear algebra algorithms can solve this problem efficiently; however, we can also explore how to solve it using gradient-based optimization as a simple example of how these techniques work.

- First, we need to obtain the gradient:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top\boldsymbol{b}. \tag{4.22}$$

  We can then follow this gradient downhill, taking small steps.

Example: linear least squares

- One can also solve this problem using Newton's method. In this case, because the true function is quadratic, the quadratic approximation employed by Newton's method is exact, and the algorithm converges to the global minimum in a single step.

- Now suppose we wish to minimize the same function, but subject to the constraint $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} \leq 1$. To do so, we introduce the Lagrangian

$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda \left(\boldsymbol{x}^{\top}\boldsymbol{x} - 1 \right). \tag{4.23}$$

We can now solve the problem

$$\min_{\boldsymbol{x}} \max_{\lambda, \lambda \geq 0} L(\boldsymbol{x}, \lambda). \tag{4.24}$$

Example: linear least squares

- The smallest-norm solution to the unconstrained least-squares problem may be found using the Moore-Penrose pseudoinverse: $\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{b}$. If this point is feasible, then it is the solution to the constrained problem. Otherwise, we must find a solution where the constraint is active.

- By differentiating the Lagrangian with respect to $\boldsymbol{x}$, we obtain the equation

$$\boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}^\top \boldsymbol{b} + 2\lambda \boldsymbol{x} = 0. \tag{4.25}$$

This tells us that the solution will take the form

$$\boldsymbol{x} = (\boldsymbol{A}^\top \boldsymbol{A} + 2\lambda \boldsymbol{I})^{-1} \boldsymbol{A}^\top \boldsymbol{b}. \tag{4.26}$$

Example: linear least squares

- The magnitude of λ must be chosen such that the result obeys the constraint. We can find this value by performing gradient ascent on λ. To do so, observe

$$\frac{\partial}{\partial \lambda} L(\boldsymbol{x}, \lambda) = \boldsymbol{x}^\top \boldsymbol{x} - 1. \tag{4.27}$$

  When the norm of $\boldsymbol{x}$ exceeds 1, this derivative is positive, so to follow the derivative uphill and increase the Lagrangian with respect to λ, we increase λ.

- Because the coefficient on the $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$ penalty has increased, solving the linear equation for $\boldsymbol{x}$ will now yield a solution with a smaller norm. The process of solving the linear equation and adjusting λ continues until $\boldsymbol{x}$ has the correct norm and the derivative on λ is 0.

Stochastic gradient descent

- Stochastic gradient descent (SGD) is an extension of the gradient descent algorithm.

- A recurring problem in machine learning is that large training sets are necessary for good generalization, but large training sets are also more computationally expensive.

Stochastic gradient descent

- The cost function often decomposes as a sum over training examples of some per-example loss function. For example, the negative conditional log-likelihood of the training data can be written as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\text{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}), \tag{5.96}$$

L is the per-example loss $L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y|\boldsymbol{x}; \boldsymbol{\theta})$.

- For additive cost functions, gradient descent computes

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}). \tag{5.97}$$

- The computational cost of this operation is $O(m)$. As the training set size grows to billions of examples, the time to take a single gradient step becomes prohibitively long.

Stochastic gradient descent

- The insight of SGD is that the gradient is an expectation, which may be approximately estimated using a small set of samples. Specifically, on each step of the algorithm, we can sample a minibatch of examples $\mathbb{B} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m')}\}$ drawn uniformly from the training set.

- The minibatch size m' is typically chosen to be a relatively small number of examples, ranging from one to a few hundred.

- Crucially, m' is usually held fixed as the training set size m grows. We may fit a training set with billions of examples using updates computed on only a hundred examples.

Stochastic gradient descent

- The estimate of the gradient is formed as

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \tag{5.98}$$

  using examples from the minibatch $\mathbb{B}$.

- The stochastic gradient descent algorithm then follows the estimated gradient downhill:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}, \tag{5.99}$$

  where ϵ is the learning rate.

Stochastic gradient descent

- For a fixed model size, the cost per SGD update does not depend on the training set size m.

- The number of updates required to reach convergence usually increases with training set size. However, as m approaches infinity, the model will eventually converge to its best possible test error before SGD has sampled every example in the training set. Increasing m further will not extend the amount of training time needed to reach the model's best possible test error.

- From this point of view, one can argue that the asymptotic cost of training a model with SGD is $O(1)$ as a function of m.

Stochastic gradient descent

- The optimization algorithm may not be guaranteed to arrive at even a local minimum in a reasonable amount of time, but it often finds a very low value of the cost function quickly enough to be useful.

- Starting in 2006, deep learning was initially interesting because it was able to generalize to new examples better than competing algorithms when trained on medium-sized datasets with tens of thousands of examples.

- Soon after, deep learning garnered additional interest in industry because it provided a scalable way of training nonlinear models on large datasets.