Lecture 4 - Gradient Descent

Contents

5 Gradient Descent

5.1 Recap

- Training a neural network revolves around the concept of *Empirical Risk Minimization (ERM)*. In this approach, we aim to minimize the prediction error across a given set of data samples.
- We are provided with a training dataset composed of input-output pairs, where each training instance consists of an input X and a corresponding desired output d. The training data serves

as the foundation for adjusting the neural network's parameters to improve its predictions.

- The variable d represents the target or desired output for each instance. It is the ground truth value that we want the network's predictions to approximate as closely as possible.
- Our primary goal is to minimize the discrepancy between the output of the neural network (i.e., the predicted function) and the desired output function. This can be visualized as minimizing the *area under the curve* of the error function. However, since we do not have access to the actual underlying function, we instead rely on sample data points to approximate this error.
- The *loss function* represents the shaded area under the error curve, measuring the average difference (distance) between the predicted output and the actual target output for each sample in the training set.
- By minimizing the loss function, we effectively reduce the error between the predicted and desired values. This process aims to decrease the overall shaded area (the error), bringing the model's predictions closer to the ground truth values.
- Once the input data points (X values) are identified, the only elements we can modify to further reduce the loss are the parameters of the neural network (i.e., the weights and biases).
- Adjusting these parameters does not change the input data points; instead, it shifts the predicted function up or down. This modification allows the function to better align with the target output, thereby reducing the error.
- Therefore, the process of training a neural network involves systematically modifying the model's parameters in such a way that the loss function is minimized, ensuring that the predictions improve over time.
- Ultimately, this problem can be framed as a *function minimization* task, where the goal is to find the set of parameters that results in the smallest possible error between the predicted and desired outputs.

5.2 Derivatives

Definition

The derivative is the multiplicative factor that multiplied with the tiny increment in x to give the tiny increment in y.

- For any $y = f(x)$, expressed as a multiplier α to a tuny increment Δx to obtain the increments Δy to the ouput.

$$\Delta y = \alpha \cdot \Delta x \tag{1}$$

Where α is the derivative.

- $\dfrac{dy}{dx}$ can only be written if both x and y are scalar.

5.3 Multivariate Scalar Function: Scalar Function of Vector Argument

- **x as a Vector**: A multi-variate function, $y = f(x)$, where $\mathbf{x}$ is now a column vector. Δx (increment in $\mathbf{x}$) is also a column vector of the same shape and size as $\mathbf{x}$.
- **Partial Derivatives**: Each partial derivative α_i tells you how much the output changes when only x_i (a particular variable) is changed.
- Partial derivatives indicate change in y due to incremental changes in each component of x individually.

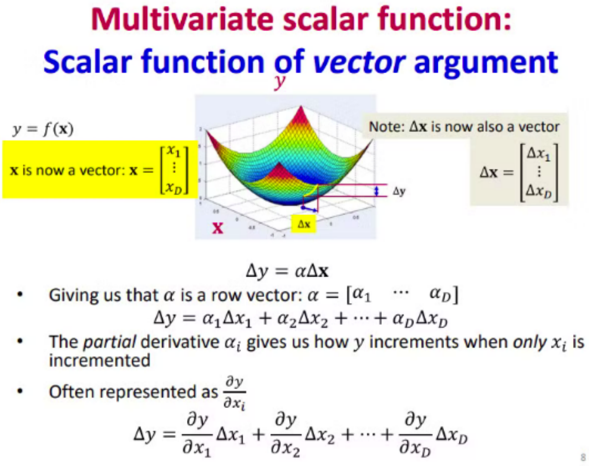
5.4 Gradient of a Scalar Function of a Vector

Definition

- **Row Vector Representation**: α is rewritten as $\nabla_x y$, which is a row vector containing partial derivatives of y with respect to each component of $\mathbf{x}$.
- **Gradient Definition**: The **gradient** is the transpose of the vector of partial derivatives, making it a column vector of the same dimensionality as $\mathbf{x}$.
- It points in the direction of steepest ascent or descent of the function value.
- **Direction of Change**: The gradient tells us the direction in which the output changes the fastest.
- Walking in the opposite direction of the gradient yields the steepest descent.
- **Formula**:

$$df(x) = \nabla_x f(x) \cdot dx$$

  This represents the change in the function value for small steps in $\mathbf{x}$.



Multivariate scalar function:

Scalar function of *vector* argument

$y = f(\mathbf{x})$

$\mathbf{x}$ is now a vector: $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$

Note: $\Delta\mathbf{x}$ is now also a vector

$\Delta\mathbf{x} = \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_D \end{bmatrix}$

$$\Delta y = \alpha \Delta\mathbf{x}$$

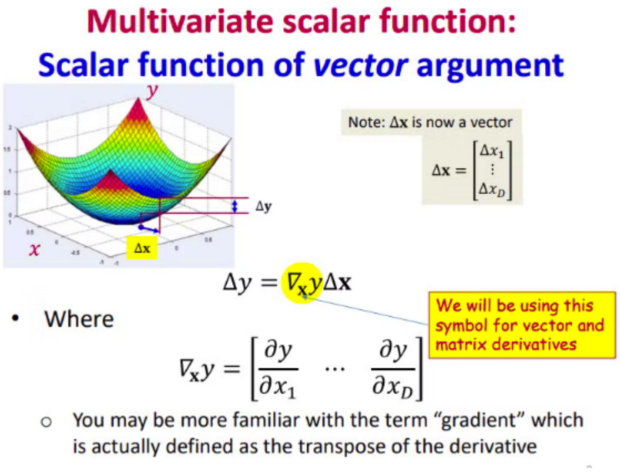
- Giving us that α is a row vector: $\alpha = [\alpha_1 \quad \cdots \quad \alpha_D]$

$$\Delta y = \alpha_1 \Delta x_1 + \alpha_2 \Delta x_2 + \cdots + \alpha_D \Delta x_D$$

- The *partial* derivative α_i gives us how y increments when *only* x_i is incremented
- Often represented as $\frac{\partial y}{\partial x_i}$

$$\Delta y = \frac{\partial y}{\partial x_1}\Delta x_1 + \frac{\partial y}{\partial x_2}\Delta x_2 + \cdots + \frac{\partial y}{\partial x_D}\Delta x_D$$

8



Multivariate scalar function:

Scalar function of *vector* argument

Note: $\Delta\mathbf{x}$ is now a vector

$\Delta\mathbf{x} = \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_D \end{bmatrix}$

$$\Delta y = \nabla_{\mathbf{x}} y \Delta\mathbf{x}$$

We will be using this symbol for vector and matrix derivatives

- Where

$$\nabla_{\mathbf{x}} y = \left[\frac{\partial y}{\partial x_1} \quad \cdots \quad \frac{\partial y}{\partial x_D}\right]$$

  ○ You may be more familiar with the term “gradient” which is actually defined as the transpose of the derivative

5.5 Properties of the Gradient

- **Gradient's Role in Function Increase**: The gradient $\nabla_x f(x)$ is the vector of partial derivatives and dictates the direction in which the function increases the fastest.
- The function increases most rapidly when Δx (the change in $\mathbf{x}$) is aligned with the gradient.
- **Zero Gradient**: $\nabla_x y = 0$ means the function value does not change no matter which direction a small step is taken.

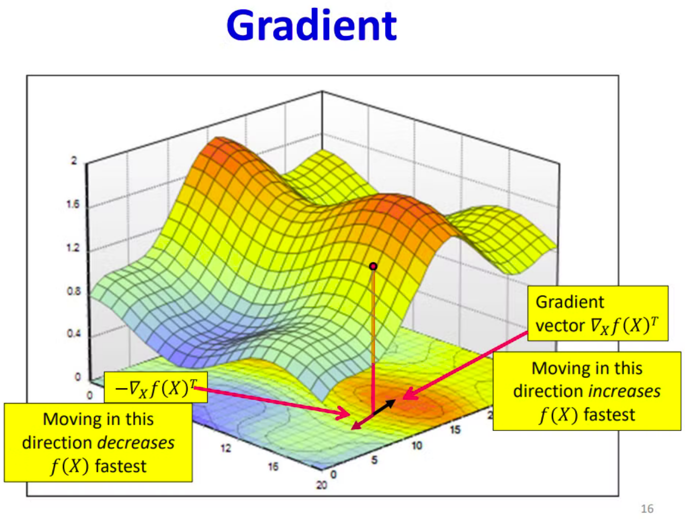
5.6 Inner Product and Vector Behavior

The below equation is a vector inner product.

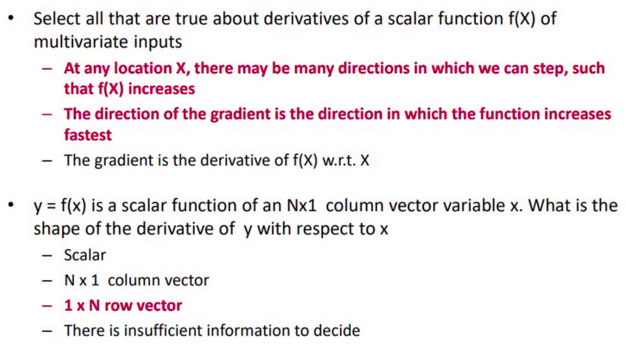
$$df(X) = \nabla_x f(X) dX \tag{2}$$

5.6.1 Properties of Inner Products

- **Inner Product of Vectors**: The inner product $u^T v = ||u||||v|| \cos\theta$ quantifies both the length of vectors and the angle between them.
- **Maximum Length**: When the angle $\theta = 0$, the vectors are aligned, and the inner product is maximized.
- **Gradient and Inner Product**: The derivative $df(x)$ is the inner product between the gradient $\nabla_x f(x)$ and dx. The change in the function value is largest when dx is in the same direction as the gradient.

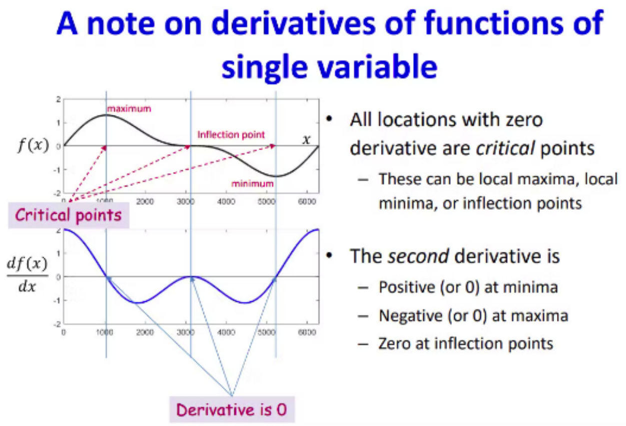


The Gradient vector $\nabla_X f(X)^T$ is perpendicular to the level curve. Slicing the function, then we'll get an equal height contour. Those edges are all at the same height. When Looking at the shadows on the ground it would be a curve, that's the level set the gradient is always going to be orthogonal to the level set.



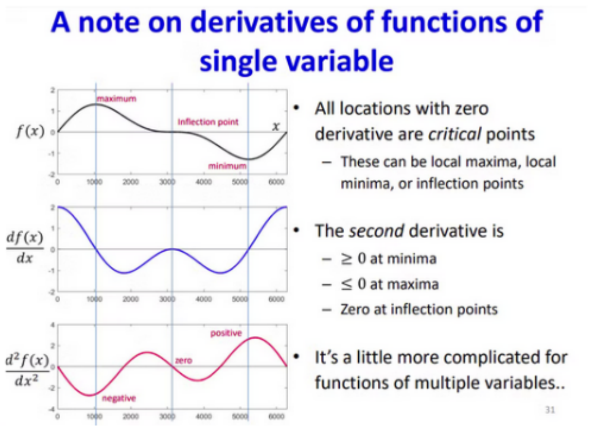
- Select all that are true about derivatives of a scalar function f(X) of multivariate inputs
  - **At any location X, there may be many directions in which we can step, such that f(X) increases**
  - **The direction of the gradient is the direction in which the function increases fastest**
  - The gradient is the derivative of f(X) w.r.t. X
- y = f(x) is a scalar function of an Nx1 column vector variable x. What is the shape of the derivative of y with respect to x
  - Scalar
  - N x 1 column vector
  - **1 x N row vector**
  - There is insufficient information to decide

5.7 The Problem of Optimization



A note on derivatives of functions of single variable

- All locations with zero derivative are *critical* points
  - These can be local maxima, local minima, or inflection points
- The *second* derivative is
  - Positive (or 0) at minima
  - Negative (or 0) at maxima
  - Zero at inflection points



A note on derivatives of functions of single variable

- All locations with zero derivative are *critical* points
  - These can be local maxima, local minima, or inflection points
- The *second* derivative is
  - ≥ 0 at minima
  - ≤ 0 at maxima
  - Zero at inflection points
- It's a little more complicated for functions of multiple variables..

Things get crazy when we have functions of multiple variables. The optimal is still likely to be a turning point where the gradient is zero.

5.7.1 Optimum as a Turning Point:

For a function $f(X)$ of multiple variables X, finding the minimum still involves identifying points where the gradient is zero. The gradient ${}_X f(X)$ is a vector of partial derivatives, and setting this to zero gives the candidate points for optimization:

$$\nabla_X f(X) = 0$$

The gradient gives us the direction and magnitude of the steepest ascent or descent. However, equating the gradient to zero alone is not sufficient to determine whether the point is a minimum, maximum, or saddle point. To determine the nature of the stationary point, we must analyze the second derivative.

5.7.2 Role of the Hessian Matrix

Second Derivative and Curvature: To confirm whether a turning point is a minimum (or maximum), we must examine the second derivative of the function. In the case of multi-variable functions, the second derivative is captured by the **Hessian matrix**, H, which is a square matrix of second-order partial derivatives:

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

The Hessian captures how the second derivative changes with respect to each variable, revealing important information about the curvature of the function at that point.

Directional Curvature: For functions of two variables, the curvature of the function can vary depending on the direction we take. If we imagine slicing through the function along different directions, the second derivative (curvature) may differ for each slice. This directional dependence of curvature is what makes optimization in multiple dimensions more complex than in a single dimension.

5.7.3 Interpretation of the Hessian for Optimization

Eigenvectors and Eigenvalues of the Hessian: The eigenvectors of the Hessian matrix indicate the principal directions of curvature, while the eigenvalues correspond to the curvature along these directions. Specifically:

$$H\mathbf{v} = \lambda\mathbf{v}$$

where $\mathbf{v}$ is an eigenvector and λ is the corresponding eigenvalue. The eigenvalues tell us whether the function curves upward or downward in each principal direction:

- If all the eigenvalues are positive ($\lambda_i > 0$ for all i), the function has a **local minimum**.
- If all the eigenvalues are negative ($\lambda_i < 0$ for all i), the function has a **local maximum**.
- If some eigenvalues are positive and others are negative, the function has a **saddle point**.

The Hessian matrix encapsulates this directional curvature. Its eigenvectors represent the principal directions of curvature, and the eigenvalues tell us how the function behaves along these directions.

Boat analogy: Imagine the curve is like a boat. **The second derivative (curvature) would vary depending on the direction of the slice.** This is why the Hessian, a matrix of second derivatives, is crucial—it captures these variations in different directions.

Strict Minimum: For a strict local minimum, every single eigenvalue of the Hessian must be positive, indicating that the function curves upwards in every direction. In other words, at a strict minimum, the curvature along all principal directions is positive, meaning that the function is "bowl-shaped" at that point:

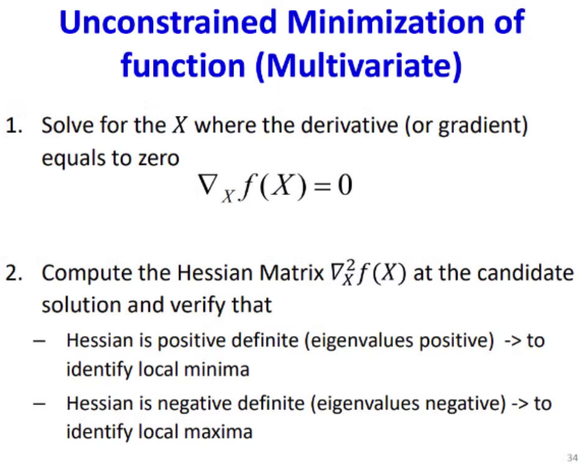
$$\lambda_i > 0 \quad \text{for all } i$$

This ensures that the function reaches its lowest value in the neighborhood of that point, and any small perturbation in the input will result in an increase in the function's value.

5.8 When Closed-Form Solutions Fail

In many real-world scenarios, especially with more complex functions, it is not possible to solve $\nabla_X f(X) = 0$ explicitly due to the intractability of the function. In such cases, closed-form solutions (i.e., direct analytical solutions) are unavailable. Instead, we resort to iterative methods like gradient descent, Newton's method, or others:

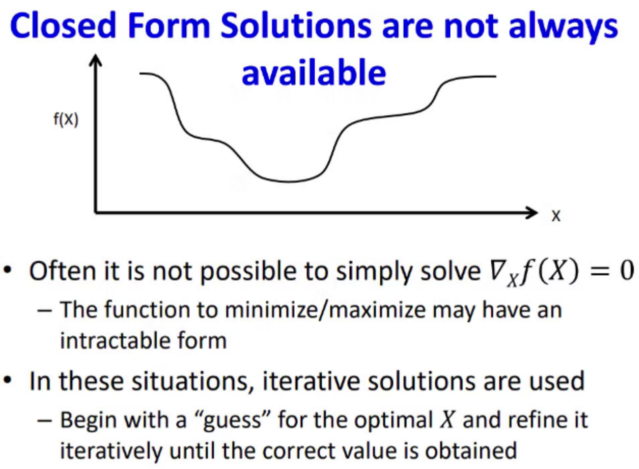
- Start with an initial guess for X.
- Refine this guess iteratively using a rule (such as moving in the opposite direction of the gradient) until convergence is achieved.



Unconstrained Minimization of function (Multivariate)

1. Solve for the X where the derivative (or gradient) equals to zero
$$\nabla_X f(X) = 0$$
2. Compute the Hessian Matrix $\nabla_X^2 f(X)$ at the candidate solution and verify that
   - Hessian is positive definite (eigenvalues positive) -> to identify local minima
   - Hessian is negative definite (eigenvalues negative) -> to identify local maxima

34



Closed Form Solutions are not always available

- Often it is not possible to simply solve $\nabla_X f(X) = 0$
  - The function to minimize/maximize may have an intractable form
- In these situations, iterative solutions are used
  - Begin with a "guess" for the optimal X and refine it iteratively until the correct value is obtained

These iterative approaches are especially useful when the function has complex structures, non-convexities, or multiple minima and maxima.

5.8.1 Iterative Methods

Instead of finding the solution in one step, we use **iterative methods**. These methods approach the solution gradually, refining a guess over time. Here's how it works:

1. Initial Guess: Start with a guess for X. This can be a random guess or based on domain knowledge. For example, if you're optimizing a function related to a physical system, you might have a rough idea of where the optimal solution lies.

2. Update the Guess: Use an iterative rule to update the guess. One common rule is **gradient descent**, where you adjust your guess in the direction of steepest descent (the direction opposite to the gradient of the function). Mathematically, this is expressed as:

$$X_{\text{new}} = X_{\text{old}} - \eta \nabla_X f(X_{\text{old}})$$

where η is the *learning rate* (a small step size).

- If the gradient is large, the function is steep, and the update will be significant.
- If the gradient is small, the function is relatively flat, and the update will be minor.

3. Repeat Until Convergence: Continue refining the guess until the updates become very small, indicating that you've reached a point close to the optimum. This is when the function value no longer changes significantly between iterations. In practice, we often stop when:

$$\|\nabla_X f(X)\| \text{ is sufficiently small}$$

or the improvement in function value $f(X)$ becomes minimal.

5.8.2 When to Use Iterative Methods

Iterative methods like gradient descent are crucial when:

- **Closed-form solutions don't exist**: For example, minimizing neural network loss functions in machine learning.
- **Function is too complex**: Functions with many variables and complex relationships (non-linear, non-convex).
- **Speed and efficiency**: Iterative methods can approximate solutions quickly, making them practical for large datasets or models.

Iterative methods allow us to approximate solutions even for very complex problems where finding the exact answer would be too computationally expensive or impossible. **Gradient Descent is an example of Iterative Method**.

This iterative approach is particularly useful in situations where the function:

- Has multiple variables, making analytical solutions complicated.
- Contains non-linearities that make exact derivatives difficult to solve.
- Has a complex shape with multiple local minima or maxima.

5.9 Gradient Descent

In **gradient descent**, we aim to minimize a function by moving in the direction opposite to the gradient.

5.9.1 For Uni-variate Functions

We often use **gradient descent** to find the minimum of a univariate function. This involves starting at a random point, checking if it is a minimum, and then iteratively adjusting the point until we believe we have found the minimum.

Gradient Descent Uni-variate Functions (Imp.)

- Start at some random point x_0 and check whether this point is a minimum or maximum by evaluating the derivative $f'(x_0)$.
- If $f'(x_0) \neq 0$, we move to a new point x_1 according to the sign of the derivative.

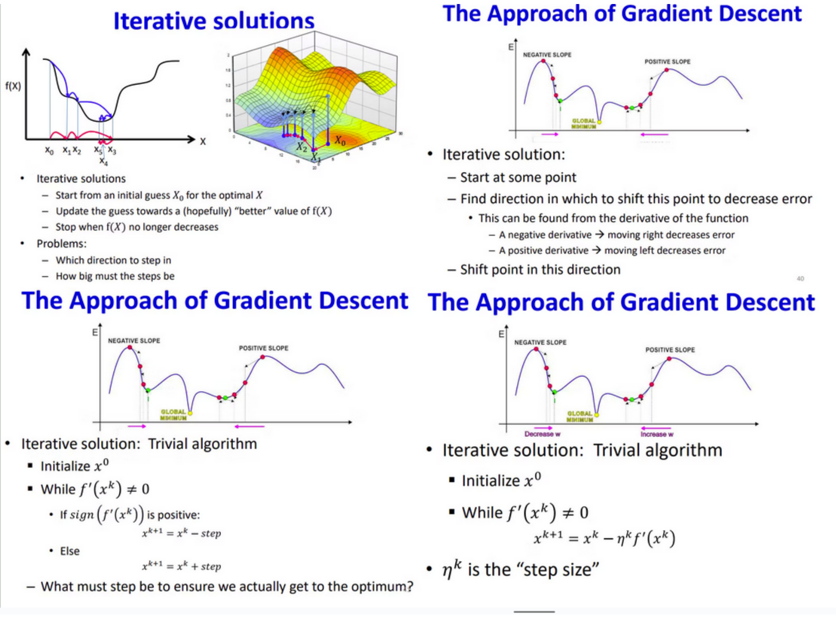
$$x_{k+1} = x_k - \eta \cdot f'(x_k) \tag{3}$$

  where η is the step size (learning rate).

  - If $f'(x_0) > 0$, the function is increasing, meaning we might be at or near a local maximum. We step in the opposite direction (i.e., take a step back) to decrease the function.
  - If $f'(x_0) < 0$, the function is decreasing, so we take a step forward to approach the minimum.

- We continue this process, updating x at each step until $f'(x) = 0$, which could indicate a local minimum, maximum, or saddle point.
- To confirm if we have reached a local minimum or maximum, we examine the second derivative $f''(x)$:
  - If $f''(x) > 0$, the function is concave upwards, confirming a local minimum.
  - If $f''(x) < 0$, the function is concave downwards, confirming a local maximum.

In other words, we are always taking a step **against the derivative** to find the minimum.



Iterative solutions

- Iterative solutions
  - Start from an initial guess X_0 for the optimal X
  - Update the guess towards a (hopefully) "better" value of f(X)
  - Stop when f(X) no longer decreases
- Problems:
  - Which direction to step in
  - How big must the steps be

The Approach of Gradient Descent

- Iterative solution:
  - Start at some point
  - Find direction in which to shift this point to decrease error
    - This can be found from the derivative of the function
      - A negative derivative → moving right decreases error
      - A positive derivative → moving left decreases error
  - Shift point in this direction

The Approach of Gradient Descent

- Iterative solution: Trivial algorithm
  - Initialize x^0
  - While $f'(x^k) \neq 0$
    - If $sign\left(f'(x^k)\right)$ is positive:

      $x^{k+1} = x^k - step$
    - Else

      $x^{k+1} = x^k + step$
  - What must step be to ensure we actually get to the optimum?

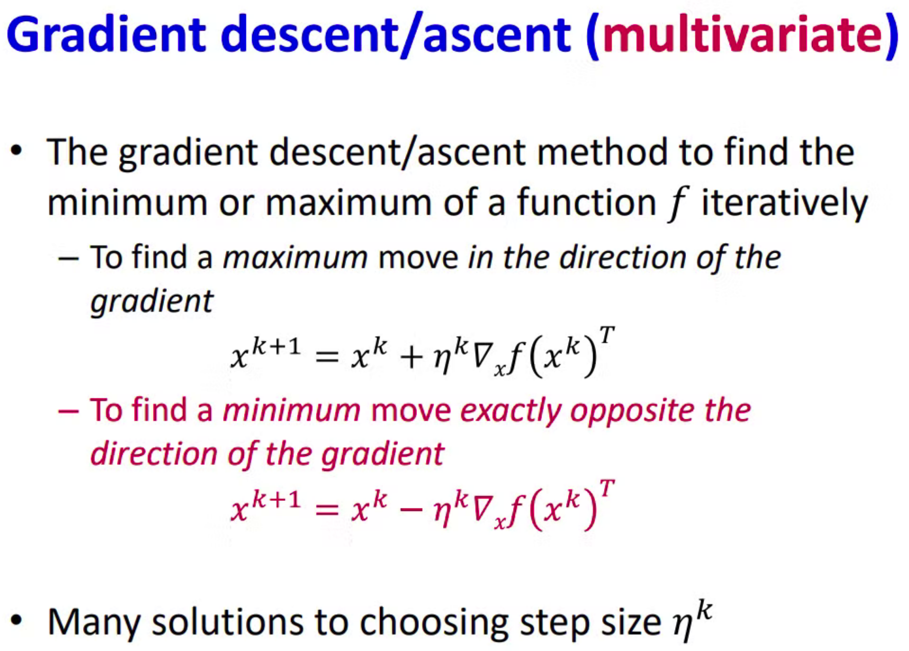
The Approach of Gradient Descent

- Iterative solution: Trivial algorithm
  - Initialize x^0
  - While $f'(x^k) \neq 0$

    $x^{k+1} = x^k - \eta^k f'(x^k)$
- η^k is the "step size"

5.9.2 For Bi-variate Functions

In a multivariate function we know that moving in the direction of the gradient is where the function increases the fastest, and moving opposite of the gradient is where the function decreases the fastest. **To find a minimum we always move opposite to the direction of the gradient**.



Gradient descent/ascent (multivariate)

- The gradient descent/ascent method to find the minimum or maximum of a function f iteratively
  - To find a *maximum* move *in the direction of the gradient*

$$x^{k+1} = x^k + \eta^k \nabla_x f\left(x^k\right)^T$$

  - To find a *minimum* move *exactly opposite the direction of the gradient*

$$x^{k+1} = x^k - \eta^k \nabla_x f\left(x^k\right)^T$$

- Many solutions to choosing step size η^k

I can stop when the gradient is 0 or until the function does not decrease a lot.

5.10 Gradient Descent Convergence Criteria

The gradient descent algorithm converges when one of the following criteria is satisfied:

$$|f(x^{(k+1)}) - f(x^{(k)})| < \epsilon_1$$

This checks if the change in function values between consecutive iterations is small.

$$\|\nabla_x f(x^{(k)})\| < \epsilon_2$$

This checks if the gradient (i.e., the slope) is small, indicating proximity to a local minimum.

5.11 Gradient Descent Algorithm

5.12 Key Observations

- This method aims to find a **local minimum**.
- Far from the minimum, the slope is steep, so larger steps are needed. Closer to the minimum, the slope is smaller, so smaller steps are needed.

Gradient Descent Algorithm

We initialize the algorithm as follows:

$$x^0, \quad k = 0$$

Then iteratively apply the following steps:

1. Update the next point:
$$x^{k+1} = x^k - \eta \nabla_x f(x^k)$$
where η is the step size (learning rate) and $\nabla_x f(x^k)$ is the gradient of the function at x^k.

2. Increment the iteration counter:
$$k = k + 1$$

3. Continue iterating until the difference in function values is smaller than a pre-defined threshold:
$$|f(x^{k+1}) - f(x^k)| > \epsilon$$

- The derivative is large far from the minimum and small near the minimum. This guides the size of the step.
- If the step size η is constant, the algorithm works fine for simple functions. However, it may not work well for more complex functions with irregular shapes or many local minima.

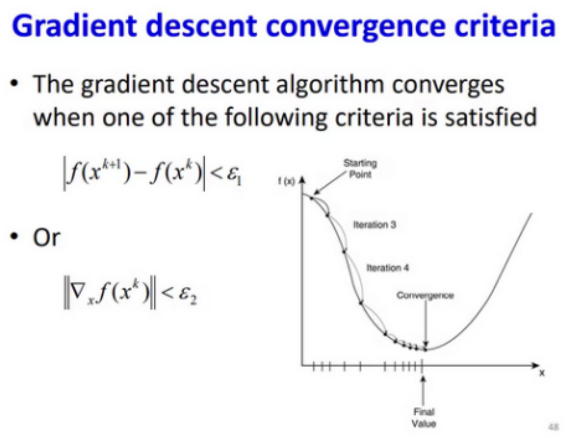
5.13 Additional Details

5.13.1 Choosing the Step Size

- **Far from the minimum**: Since the slope is steep, you need to take **big steps**.
- **Close to the minimum**: Since the slope is shallow, you need to take **small steps**.

5.13.2 Special Considerations for Complex Functions

For functions with **complex structures**, non-convexities, or multiple local minima and maxima, a constant step size η may lead to suboptimal results. More advanced techniques, such as **adaptive step sizes** or **momentum-based methods**, may be required.



Gradient descent convergence criteria

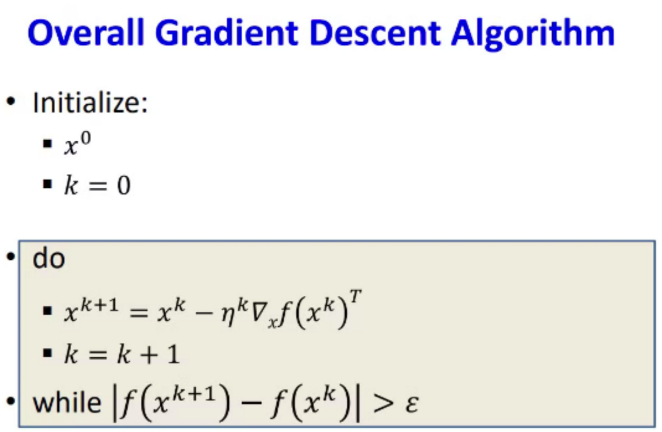
- The gradient descent algorithm converges when one of the following criteria is satisfied

$$\left|f(x^{k+1}) - f(x^k)\right| < \varepsilon_1$$

- Or

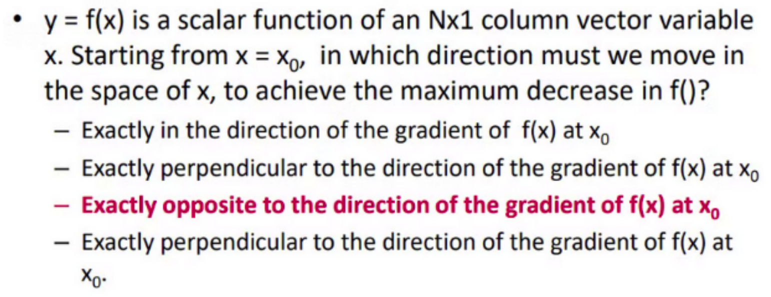
$$\left\|\nabla_x f(x^k)\right\| < \varepsilon_2$$

48



Overall Gradient Descent Algorithm

```
• Initialize:
    ▪ $x^0$
    ▪ $k = 0$
• do
    ▪ $x^{k+1} = x^k - \eta^k \nabla_x f(x^k)^T$
    ▪ $k = k + 1$
• while $\left|f(x^{k+1}) - f(x^k)\right| > \varepsilon$
```



- y = f(x) is a scalar function of an Nx1 column vector variable x. Starting from x = x_0, in which direction must we move in the space of x, to achieve the maximum decrease in f()?
  - Exactly in the direction of the gradient of f(x) at x_0
  - Exactly perpendicular to the direction of the gradient of f(x) at x_0
  - **Exactly opposite to the direction of the gradient of f(x) at x_0**
  - Exactly perpendicular to the direction of the gradient of f(x) at x_0.