CSC 461: Machine Learning

Fall 2024

Gradient descent

Prof. Marco Alvarez, Computer Science
University of Rhode Island

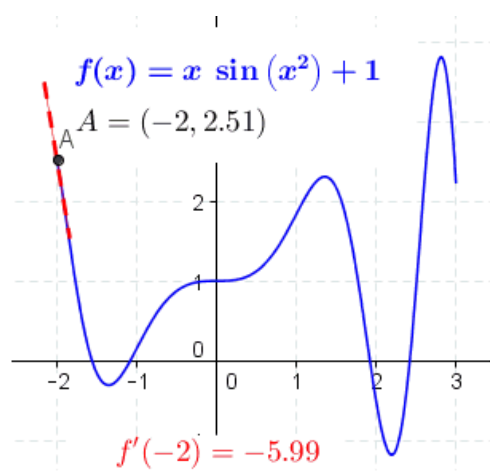
Derivatives and gradients

From calculus …

- For a function $f(x)$, its derivative $f'(x)$ represents:
  - rate of change at a point (how much a very small changes to the argument will change the value of the function)
  - steepness of the function at x
  - direction of increase/decrease
- Interpreting derivatives
  - positive slope
    - very small increase in x increases $f(x)$
  - negative slope
    - very small increase in x decreases $f(x)$
  - zero slope
    - local minimum, local maximum, or saddle point

The derivative of f with respect to x is $\frac{\partial f}{\partial x}$.
Both x and f can be a scalar, vector, or matrix.

Scalar function of a scalar argument



For a function $y = f(x)$ the derivative is given by $f'(x) = \alpha$, such that $\Delta y = \alpha \Delta x$

Scalar function of vector argument

- Input: a column vector $\mathbf{x}$
  - note that $\Delta\mathbf{x}$ is also a vector
- Output: a scalar
- Derivative
  - a row vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n]$
  - partial derivative α_i indicates how output y changes when x_i is changed

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

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

$$\Delta y = \alpha \Delta \mathbf{x} = \frac{\partial y}{\partial x_1}\Delta x_1 + \frac{\partial y}{\partial x_2}\Delta x_2 + \ldots + \frac{\partial y}{\partial x_n}\Delta x_n$$

Examples

- What is the derivative of:

$$f(\mathbf{x}) = x_1^3 + 2x_2 + 5x_3^4?$$

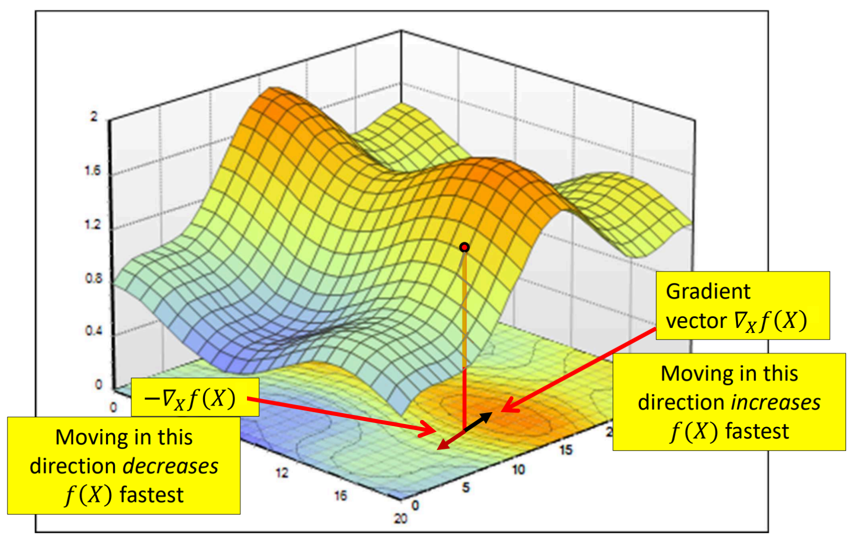
$$f(x, y, z) = x^3 + 2y + 5z^4?$$

$$f(x, y, z) = x^3y^2 + 2xy + 5yz^4?$$

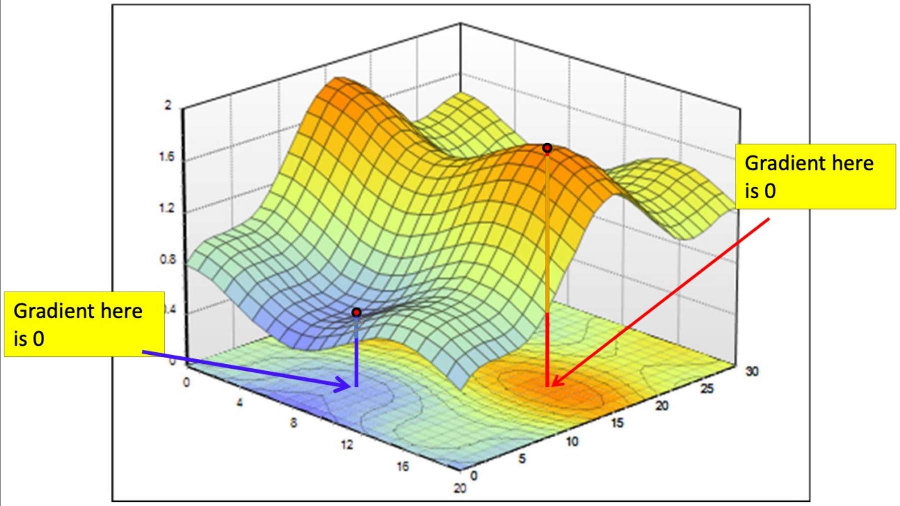
The gradient

- A **gradient** is the transpose of the **derivative**
  - a column vector of **partial derivatives**

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_d} \end{bmatrix}$$



From 11-785 Introduction to Deep Learning at CMU



From 11-785 Introduction to Deep Learning at CMU

The Hessian matrix

- Square matrix containing all second-order partial derivatives

$$\nabla^2_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_d \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_d \partial x_d} \end{bmatrix}$$

Gradient descent

Unconstrained optimization

- Given a multivariate function $f(\mathbf{x})$
  - calculate the derivative and solve for $\mathbf{x}$ where the derivative is zero
  - compute the Hessian at solution $\mathbf{x}^*$
    - if positive definite (positive eigenvalues) then it is a local minima
    - if negative definite (negative eigenvalues) then it is a local maxima
  - note this approach is not scalable
- Example
  - $f(x, y, z) = x^3 + 3x^2y - yz^3 + z^2$

From Linear Regression

- Minimize loss using a **closed-form** solution
  - setting the derivative of the loss to 0, then solving for **w**

$$\boxed{\arg\min_{\mathbf{w}} \frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2} \qquad \mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Issues?
  - what happens if we change the loss function?
  - what if data has high-dimensionality?

Alternative solution

- Iterative methods
  - apply an update rule iteratively until finding the solution (or approximating)

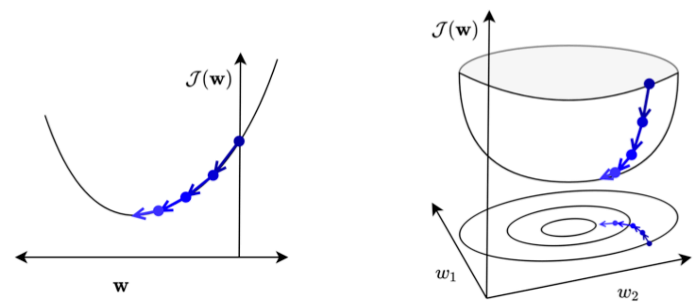


Figure from https://www.cs.toronto.edu/~rgrosse/courses/csc311_f21/lectures/lec03.pdf

Gradient descent

- Optimization technique used to find the value of x where $f(x)$ is **minimum**
  - randomly guess a starting point
  - walk iteratively (taking steps) in the **opposite direction** of the function's gradient
- Alternatively, to find the **maximum**, walk in the **direction** of the gradient (gradient ascent)
- Step size (a.k.a. **learning rate**) is critical
  - hyperparameter

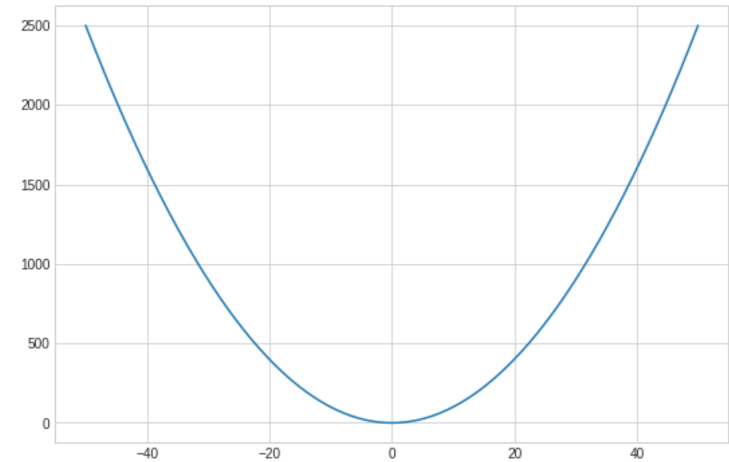
Gradient descent

Initialize **w** randomly

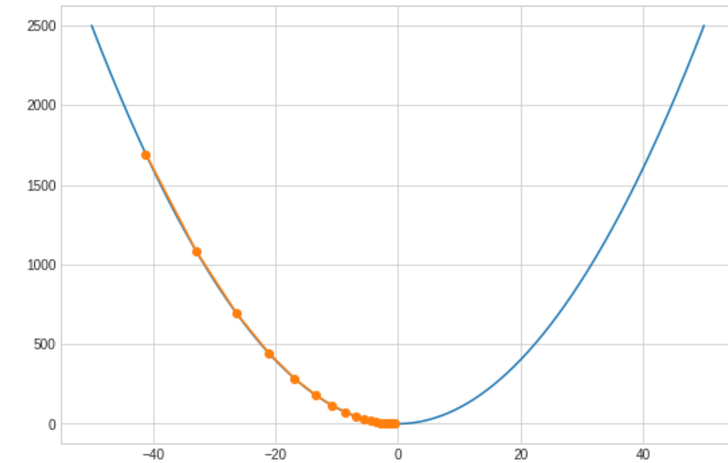
Repeat until convergence

$$w_{t+1} = w_t - \eta \cdot \nabla_w f(\mathbf{x})$$

Example: $f(x) = x^2$



Example: $f(x) = x^2$



Show me the code

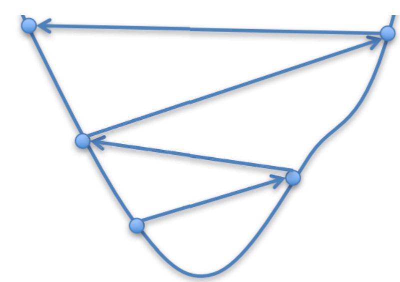
```python
# define a function and its derivative
f = lambda x: x ** 2
df = lambda x: 2 * x

# apply gradient descent
n_steps, l_rate = 10, 2
sol = np.random.randint(-50, 50)
for i in range(n_steps):
    sol = sol - (l_rate * df(sol))
    print(f'{sol:.4f}')
```

The learning rate (step size)

slow convergence

too small

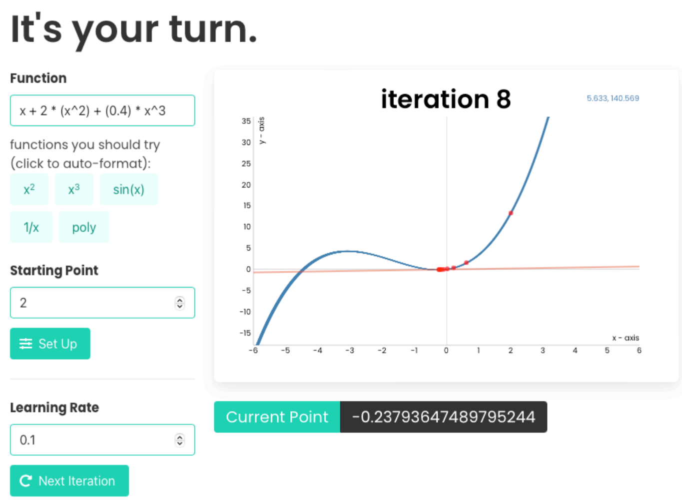
overshoots, may not converge



too large

https://www.cc.gatech.edu/~bboots3/CS4641-Fall2018/Lecture3/03_LinearRegression.pdf

Playground



https://uclaacm.github.io/gradient-descent-visualiser/#playground