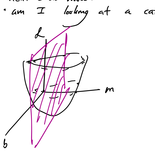
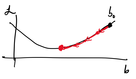
Gradient Descent

Monday, July 17, 2023 1:40 PM

- for $\mathcal{L}_{MSE}$ on a line, we had a nice, closed form solution
  can be represented by an algebraic expression we can solve
- next chess move?
- am I looking at a cat?



- can we find the minimum of the loss landscape using some numerical procedure?
  - okay to find approximate solution for b^* and m^*
  - assume we can measure the slope of the loss landscape at any point



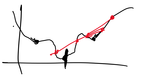
- start @ some initial guess b_0
  1. measure the slope at current location
  2. if slope ≈ 0 stop and set $b^* = b_0$
     else: take a step downhill
  3. repeat 1 and 2.

$$b_{new} = b_{old} - \delta \frac{\partial \mathcal{L}}{\partial b}\Big|_{b_{old}}$$

delta = learning rate

Q: what happens if δ is very small?
- learn slowly
- prone to get stuck in local minimum
- may be more accurate

Q: what if δ is too big?
- may get out of local minima
- make it unstable



Example:

$\mathcal{L}(b) = b^2 \qquad \frac{\partial \mathcal{L}}{\partial b} = 2b$

$b_0 = 10 \qquad \delta = 0.2 \qquad$ 4 steps

$\underset{b_{old}}{10} - (0.2)(2 \times 10) = \underset{b_{new}}{4} \qquad$ step 1

$4 - (0.2)(2 \times 4) = 1.6 \qquad$ step 2

$1.6 - (0.2)(2 \times 1.6) = 0.64 \qquad$ step 3

$0.64 - (0.2)(2 \times 0.64) = 0.256 \qquad$ step 4

$b \to 0$

check: $\frac{\partial \mathcal{L}}{\partial b}\Big|_{b=0} = 2b = 2(0) = 0$ ✓

Generalizing to $\mathcal{L}_{MSE}$

$$m_{new} = m_{old} - \delta \frac{\partial \mathcal{L}}{\partial m}\Big|_{m_{old}, b_{old}}$$
$$b_{new} = b_{old} - \delta \frac{\partial \mathcal{L}}{\partial b}\Big|_{m_{old}, b_{old}}$$

$$\left\} \begin{bmatrix} m_{new} \\ b_{new} \end{bmatrix} = \begin{bmatrix} m_{old} \\ b_{old} \end{bmatrix} - \delta \underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial m} \\ \frac{\partial \mathcal{L}}{\partial b} \end{bmatrix}}_{\text{gradient of the function } \mathcal{L}} \Leftarrow \right.$$

$\vec{\nabla} \mathcal{L}$

$\vec{\nabla} \mathcal{L}$ points in the direction of steepest ascent

$$\begin{bmatrix} m_{new} \\ b_{new} \end{bmatrix} = \begin{bmatrix} m_{old} \\ b_{old} \end{bmatrix} + \delta \vec{\nabla} \mathcal{L}$$

Example:

$f(x,y) = x^2 + 10y^2 \qquad \frac{\partial f}{\partial x} = 2x \qquad \frac{\partial f}{\partial y} = 20y$

$$\vec{\nabla} f\Big|_{x=3,\, y=-2} = \begin{bmatrix} 2x \\ 20y \end{bmatrix} = \begin{bmatrix} 6 \\ -40 \end{bmatrix}$$

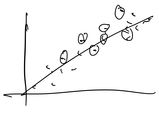
$F(m, b; x) \qquad f(x,y)$

Generalize to other models and objective functions

$$F(m, b; x) \to F(\underbrace{w_0, w_1 \ldots w_{m-1}}_{\text{"model parameters" "model weights"}}; x) \to F((w_i)_{i=0}^{m-1}; x)$$

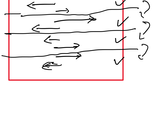
$$\mathcal{L}(w_0, w_1 \ldots w_{m-1}; (x_i, y_i)_{i=0}^{N-1}) \to \text{same for the loss of data point } x_i, y_i$$

data, true label



batching: N_b = size of a "batch" of data

$$\mathcal{L}_{batch} = \frac{1}{N_b} \sum_{i=0}^{N_b-1} \mathcal{L}(y_i^{(pred)}, y_i^{(true)})$$



$N_b \to N \qquad \mathcal{L}_{batch} \to \mathcal{L}_{all}$

small batches
- noisier
- helps get out of local minima
- faster to run

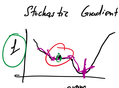
large batches
- more computation
- more precise
- more accurate to true loss
- less noise to get out of local minima

Hyperparameters

$F(m, b; x)$
parameters

- δ learning rate
- # of step sizes
- method for determining m_0, b_0

Stochastic Gradient Descent



- pick random batches ← not using all training data at once
- move
- repeat

"Full" GD
- use all training data

"Stochastic" GD
- use random batches

- use fixed batches
- fixed order

Supervised Learning

dataset → label $y^{(true)}$

dataset → data/batch $x \longrightarrow F((w_i); x) \to y^{(pred)} \to \mathcal{L}(y^{(pred)}, y^{(true)})$