Lecture 16 - SGD and GA

Stochastic Gradient decent: (SGD):

- In simulated annealing if your temperature is zero you can only go up, never go down.
- You still have to explore the neighborhood to find out which direction is up.
- If we know more about the landscape we can directly go up/down.

Steepest decent: use the derivative to take steps into the right direction.

$\Rightarrow \underset{\Theta}{\operatorname{argmin}} \; J(\Theta)$

$\Rightarrow$ take the derivative $\nabla J(\Theta)$

$\Rightarrow \Theta^{(i+1)} = \Theta^{i} - \alpha \nabla J(\Theta)$

↳ uses specified stepsize

- In many cases $J(\Theta)$ has the form

$$J(\Theta) = \sum_{j=1}^{N} J_j(\Theta)$$

$\Rightarrow$ $$\Theta^{i+1} = \Theta^{i} - \alpha \sum_{i=1}^{N} \nabla J_j(\Theta)$$

each data point adds a derivative to this sum.

stochastic gradient decent: update after only one element of the sum at a time

$$\Theta^{i+1} = \Theta^{i} - \alpha \nabla J_j(\Theta)$$

$\hookrightarrow$ we take $\nabla J_j(\Theta)$ as an approximation for $\nabla J(\Theta)$

$\Rightarrow$ Θ^{i+1} might go into the "wrong" direction

$\Rightarrow$ In the long run we should still be going down.

SGD:

- initialize Θ
- while not done:
  - randomly reshuffle all data
  - For $j = 1 \dots M$

$$\Theta^{new} = \Theta^{old} - \alpha \nabla J_j(\Theta)$$

$\Rightarrow$ Faster trick: pick random j, do update, pick next random j.

$\Rightarrow$ Can also do subset of data points (batch SGD)

Example: Linear Regression

$$J(\Theta) = \frac{1}{2} \sum_{i=1}^{N} \left(f_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

with $f_\Theta(x_i) = \Theta^T x_i$

$$\Rightarrow \Theta^{new} = \Theta^{old} - \alpha \left(f_\Theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

α: How much do you believe the local approximation of the gradient leads to a minimum?

⇒ You can keep memory of the best solution so far.

⇒ last iteration might not be the best

⇒ Might have to do multiple random restarts.

⇒ Can also use momentum:

$$V_{t+1} = \mu V_t - \alpha \nabla J(\Theta)$$

$$\Theta_{t+1} = \Theta_t + V_{t+1}$$

⇒ keep memory of previous gradient

⇒ helps with shallow ridges.

Genetic Algorithms:

- Motivated by Biology and evolution.

- We have:

  - a population of individuals
  - measure of fitness (optimality)
  - operations for change: reproduction

    crossover

    mutation
  - We change the population to gradually higher average fitness.

Reproduction:

- fitter individuals have more offspring
- population size remains constant
- less fit individuals die

<u>Crossover</u>: choose a random sized part of two individuals and swap:

$$\begin{bmatrix} A \\ B \end{bmatrix} \quad \begin{bmatrix} a \\ b \end{bmatrix} \Rightarrow \begin{bmatrix} A \\ b \end{bmatrix} \quad \begin{bmatrix} a \\ B \end{bmatrix}$$

chance for crossover is typically high ($0.7 \leq P_c \leq 1$).

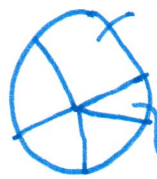
<u>Mutation</u>: randomly change part of an individual

P_m is typically small ~ 0.001

<u>GA</u>:
- compute fitness score of individuals
- Select survivers/reproduction candidates according to fitness level.
- apply crossover and mutations
- repeat all steps for this new generation.

Selecting Survivors according to fitness level:

Idea: Fortune wheel

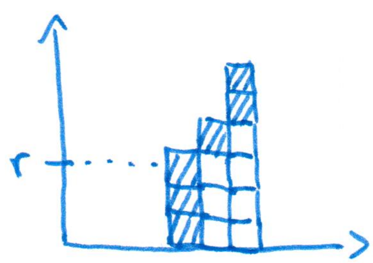


very fit, good chance of being selected

↳ not very fit ⇒ doomed (chance)

Example: 3 1 2 ⇒ fitness

3 4 6 ⇒ cumulative sum



draw random $r \sim [0, 1]$

the ▨ that you hit will have an offspring.

$$F = \sum_{i}^{n} f_i$$

↳ individual fitness

scaled fitness: $\tilde{f}_i = \frac{f_i}{F}$

incremental scaled fitness: $\tilde{g}_m = \sum_{i=1}^{m} \tilde{f}_i$

$\Rightarrow \tilde{g}_N = 1$