Gradient descent with momentum

Definition

- gradient descent with momentum is an optimization algorithm which relies on computing the **exponentially weighted (moving) averages** of gradients and using that gradient to update the weights
- build up "velocity" as a running mean of gradients
- step in the direction of the velocity over time

Why

- move faster towards the minimum loss goal.

Formulation

The computation of the exponentially weighted averages

- $V_0 = 0$
- ...
- $V_t = \beta * V_{t-1} + (1-\beta) * \theta_t$

V_t is approximately averaging over $\frac{1}{1-\beta}$ previous data points

- for $\beta = 0.5$, V_t is averaging over the last 2 values
- for $\beta = 0.9$, V_t is averaging over the last 10 values
- for $\beta = 0.98$, V_t is averaging over the last 50 values

Bias correction

- problem: fix the initial low estimates due to initializing V_0 to zero
- solution: replace V_t with $\frac{V_t}{1-\beta^t}$ (take into account the current time step)
- not often used in practice; people usually prefer waiting the exponentially weighted averaged to simply finish warming up

Variations

Mini-batch GD with momentum: smooth out the steps of gradient descent

Implementation

- initialize
    - $V_{dw} = 0$
    - $V_{db} = 0$
- compute dw and db for curent minibatch
- compute the exponentially weighted averages

- $V_{dw} = \beta * V_{dw} + (1-\beta) * dw$
- $V_{db} = \beta * V_{db} + (1-\beta) * db$

- update the weights

  $w = w - \alpha * V_{dw}$

  $b = b - \alpha * V_{db}$

Hyperparameters

- α: needs to be tuned
- $\beta = 0.9$ (average over ~ 10 gradients)

RMSprop (Root Mean Squared prop): can also speed up gradient descent

Implementation

- initialize
  - $S_{dw} = 0$
  - $S_{db} = 0$
- compute dw and db for curent minibatch
- compute the exponentially weighted averages
  - $S_{dw} = \beta * V_{dw} + (1-\beta) * dw^2$ (element-wise squaring operation)
  - $S_{db} = \beta * V_{db} + (1-\beta) * db^2$ (element-wise squaring operation)
- update the weights
  - $w = w - \alpha * \frac{dw}{\sqrt{S_{dw} + \varepsilon}}$
  - $b = b - \alpha * \frac{db}{\sqrt{S_{db} + \varepsilon}}$

Hyperparameters

- α: needs to be tuned
- $\beta = 0.999$
- $\varepsilon = 1\mathrm{e}-8$ (just to avoid zero-division errors)

ADAM (ADAptive Moment estimation): combines momentum with RSMprop

Implementation

- initialize
  - $V_{dw} = 0$
  - $V_{db} = 0$
  - $S_{dw} = 0$

- $S_{db} = 0$

- compute dw and db for curent minibatch
- compute the exponentially weighted averages
  - $V_{dw} = \beta_1 * V_{dw} + (1 - \beta_1) * dw$
  - $V_{db} = \beta_1 * V_{db} + (1 - \beta_1) * db$
  - $S_{dw} = \beta_2 * V_{dw} + (1 - \beta_2) * dw^2$ (element-wise squaring operation)
  - $S_{db} = \beta_2 * V_{db} + (1 - \beta_2) * db^2$ (element-wise squaring operation)
- apply bias correction
  - $V_{dw}^{corrected} = \frac{V_{dw}}{1 - \beta_1^t}$
  - $V_{db}^{corrected} = \frac{V_{db}}{1 - \beta_1^t}$
  - $S_{dw}^{corrected} = \frac{S_{dw}}{1 - \beta_2^t}$
  - $S_{db}^{corrected} = \frac{S_{db}}{1 - \beta_2^t}$
- update the weights
  - $w = w - \alpha * \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected} + \varepsilon}}$
  - $b = b - \alpha * \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected} + \varepsilon}}$

Hyperparameters

- α: needs to be tuned
- $\beta_1 = 0.9$
- $\beta_2 = 0.999$
- $\varepsilon = 1\mathrm{e}-8$ (just to avoid zero-division errors)