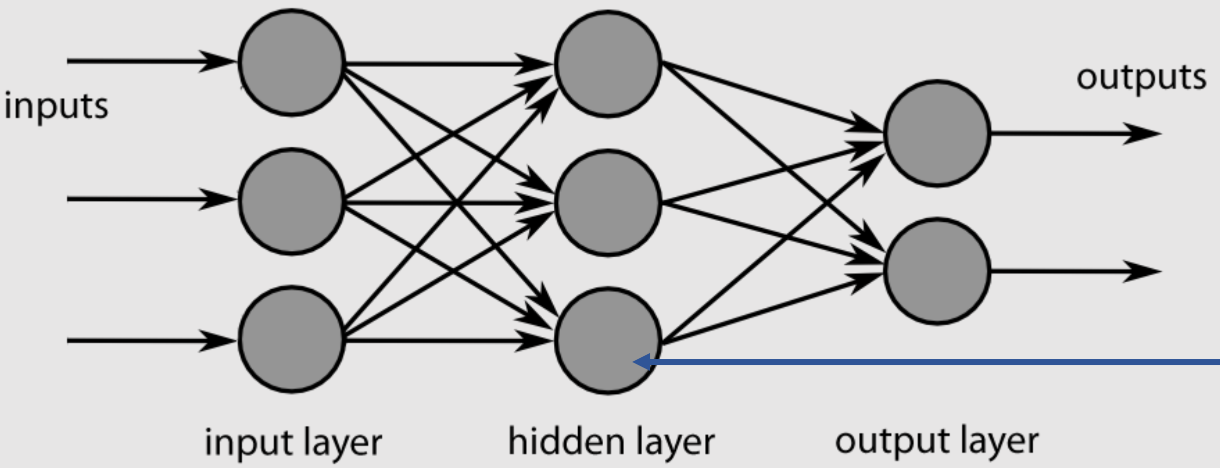
Optimization Methods - ADAM

수학과 오서영

Recall – neural network and gradient descent

Neural Network
Algorithm inspired by how the brain works
The role of neural network is to predict y_hat (We need to give the input x and output y)



$$g(wx + b)$$

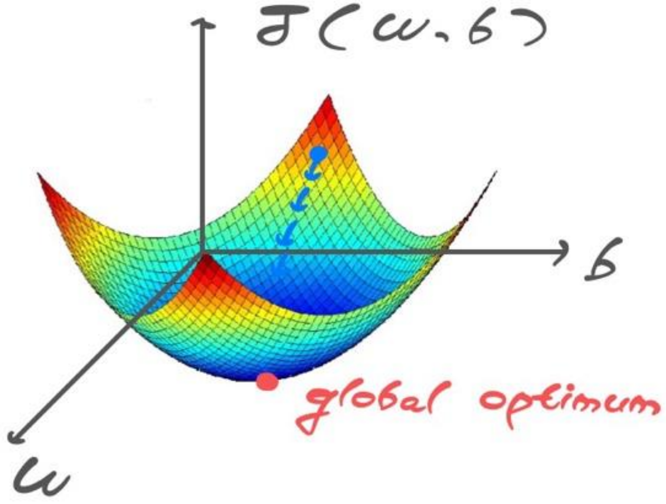
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$
$$= -\frac{1}{m}\sum_{i=1}^{m} (y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)}))$$

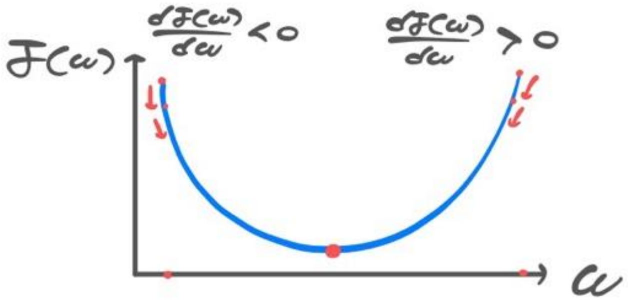
Cost Function

Recall – neural network and gradient descent

Gradient Descent

Want to find W, b that minimize J(W, b)





$$w \overset{\text{update}}{:=} w - \alpha \frac{\partial J(w, b)}{\partial w}$$

$$b := b - \alpha \frac{\partial J(w, b)}{\partial b}$$

α : learning rate

Mini-batch gradient descent

Mini-batch gradient descent

$$X = [\underbrace{x^{(1)}\ x^{(2)}\ \cdots\ x^{(1000)}}_{X^{\{1\}}} \mid x^{(1001)} \cdots \mid \underbrace{\cdots\ x^{(m)}}_{X^{\{5000\}}}]$$

(n_x, m)

"mini-batch"

i.e. mini-batch t : $X^{\{t\}}$, $Y^{\{t\}}$

$(n_x, 1000)$ $(1, 1000)$

What if m = 5,000,000 → mini-batches of 1000 epoch

Mini-batch gradient descent

Mini-batch gradient descent

> for $t = 1, \cdots, 5000$

Forward Prop on $X^{\{t\}}$

$$Z^{[1]} = W^{[1]} X^{\{t\}} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}(Z^{[L]})$$

Compute Cost $J = \frac{1}{1000} \sum_{i=1}^{l} L(\hat{y}^{(i)}, y^{(i)})$

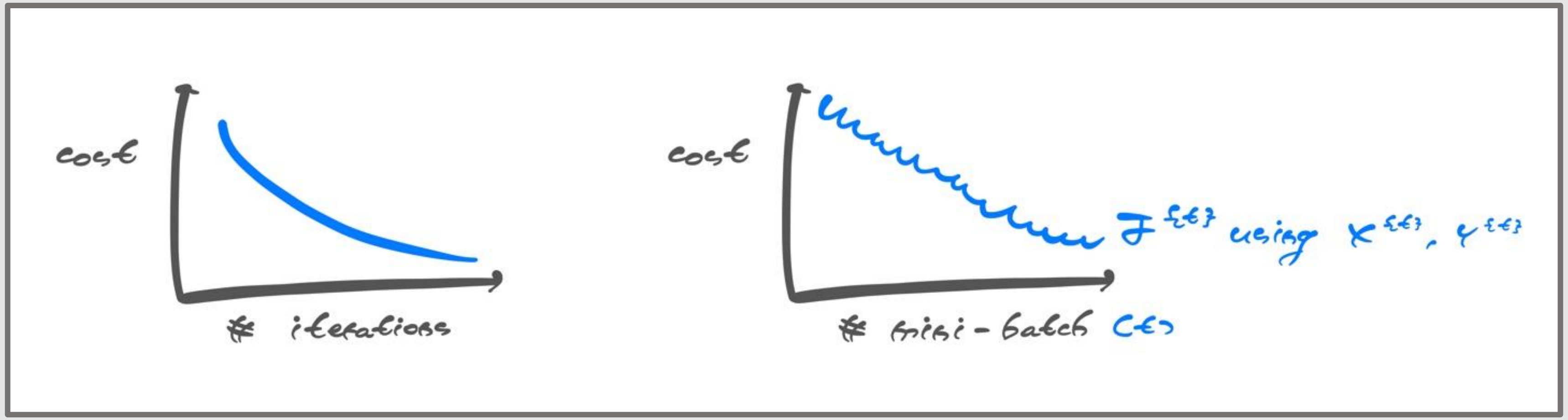
from $X^{\{t\}}, Y^{\{t\}}$

Backprop to compute gradient wrt $J^{\{t\}}$

$$W^{[l]} := W^{[l]} - \alpha dW^{[l]}$$
$$b^{[l]} := b^{[l]} - \alpha db^{[l]}$$

1 epoch
: a single pass through training examples

Mini-batch gradient descent



Batch gradient descent | **Mini-batch gradient descent**

Exponentially weighted averages

Temperature in KOREA

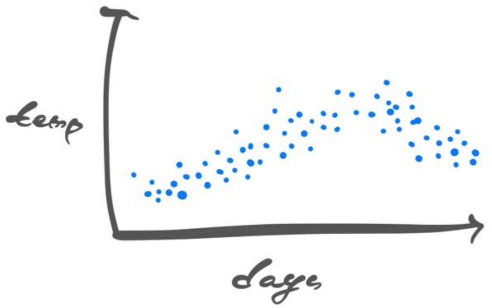
$\theta_1 = 4°C$

$\theta_2 = 4.95°C$

$\vdots$

$\theta_{180} = 15°$

$\theta_{181} = 14.6°C$



Noisy → We want to compute trend

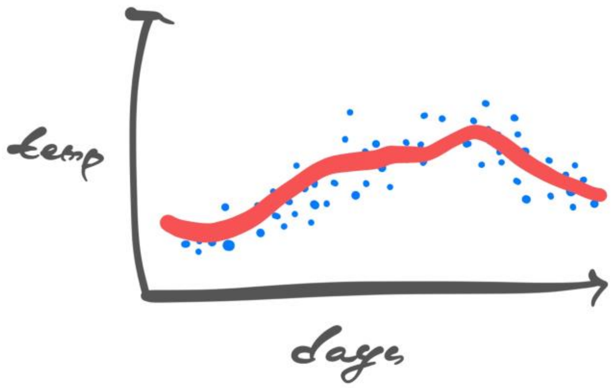
$v_0 = 0$ initialize

$v_1 = 0.9 v_0 + 0.1 \theta_1$

$v_2 = 0.9 v_1 + 0.1 \theta_2$ weighted average

$\vdots$

$v_t = 0.9 v_{t-1} + 0.1 \theta_t$



Exponentially weighted averages

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

$$\approx \frac{1}{1-\beta} \text{ day's temperature}$$

i.e.

if $\beta = 0.9$, $V_t \approx$ previous 10 days temp

if $\beta = 0.98$, $V_t \approx$ previous 50 days temp

if $\beta = 0.5$, $V_t \approx$ previous 2 days temp

$\longrightarrow$ β : hyperparameter

Bias correction in exponentially weighted averages

if $\beta = 0.98$, $v_0 = 0$,

$$v_1 = 0.02\,\theta_1$$

$$v_2 = 0.0196\,\theta_1 + 0.02\,\theta_2$$

if $v_2 < \theta_1, \theta_2$ then Bad Estimate

replace v_t with $\dfrac{v_t}{1-\beta^t}$

ex $t=2$: $1-\beta^t = 1-0.98^2 \approx 0.0396$

then $\dfrac{v_2}{1-\beta^2} = \dfrac{0.0196\,\theta_1 + 0.02\,\theta_2}{0.0396}$ Bias Correction

$\beta^t \to 0$ as $t \to \infty$

Momentum

learning rate ↑ → divergence

On iteration t :

Compute dw, db on current mini-batch

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

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

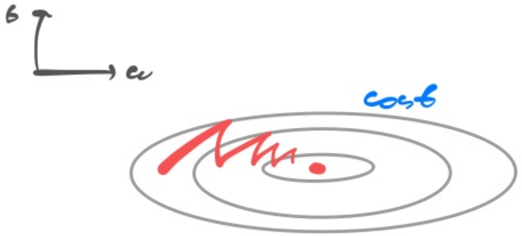
$$w = w - \alpha V_{dw} \; ; \; b = b - \alpha V_{db}$$

: momentum term → velocity

: derivative term → acceleration

→ α, β : hyperparameter

RMSprop



$\updownarrow$ slower (b)

$\longleftrightarrow$ faster (w)

On iteration t :

Compute dw, db on current mini-batch

$$S_{dw} = \beta S_{dw} + (1-\beta) dw^2$$

$$S_{db} = \beta S_{db} + (1-\beta) db^2$$

$$w = w - \alpha \frac{dw}{\sqrt{S_{dw}} + \epsilon} ; \quad b = b - \alpha \frac{db}{\sqrt{S_{db}} + \epsilon}$$

$\longrightarrow$ α, β : hyperparameter

ADAM

ADAM (Adaptive moment estimation)

- Algorithm for first-order gradient-based optimization of stochastic objective functions based on adaptive estimates of lower-order moments.
- Straightforward to implement, computationally efficient, little memory requirements, invariant to diagonal rescaling of the gradients, well suited for problems that are large in terms of data or params
- Appropriate for non-stationary objectives and problems with very noisy and sparse gradients
- Hyper params have intuitive interpretations and typically require little tuning.

ADAM computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients
-> designed to combine the advantages of AdaGrad and RMSProp
-> **Advantages** : magnitudes of param updates are invariant to rescaling of the gradient, its step sizes are approximately bounded by the step size hyperparameters

ADAM

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

ADAM

The algorithm updates exponential moving averages of the gradient (m) and The squared gradient (v) where the hyper-params beta1, beta2 control the exponential decay rates of these moving averages.

These moving averages are initialized as 0, leading to moment estimates that are biased towards zero, especially during the initial timesteps, and especially when the decay rates are small (i.e beta are close to 1)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = m_t / (1 - \beta_1^t)$$

$$\hat{v}_t = v_t / (1 - \beta_2^t)$$

References

[1] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
[2] Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization, deeplearning.ai