Difference b/w Gradient Descent V.s Regularization

(we use this when model overfits)

↓

(model learns by minimizing a cost function)

(1) Regularization methods have a pre-defined cost function, unlike Gradient Descent in which the cost function is the gradient of the given function.

(2) Regularization is used when a model overfits, unlike Gradient Descent which is used regardless of overfitting (we may find the model overfits after its validation but we do not use Gradient Descent to prevent overfitting)

6 Gradient Descent d.

⁂ It is an optimization algorithm used for minimizing the cost function in various ML algos.

⁂ It is basically used for updating the parameters of the learning model.

Types of gradient descent:

① Batch Gradient Descent: (All in one go)

→ All training examples are processed in single iteration.

→ If the no. of training examples is large, then batch gradient descent is computationally expensive & not preferred.

② Stochastic Gradient Descent (one by one)

→ It processes 1 training example each iteration. Hence, the parameter is updated even after one iteration. Hence, this is quite faster than batch gradient.

→ But, when training examples are high even then it processes only one example which can be additional overhead of the system as the no. of iterations will be quite large.

③ Mini-Batch G.D.: (in batches) Best

→ It works faster than both batch & stochastic

→ Here, b example where $b < m$ are processed per iteration.

→ So, even if the no. of ~~iteration~~ training example is large, it is processed in batches of b training examples in one go.

Conclusion: Mini Batch G.D. works for larger training examples & that too with lesser no. of iterations.

6 Gradient Descent d.

⭐⭐ It is an optimization algorithm used for minimizing the cost function in various ML algos.

⭐⭐ It is basically used for updating the parameters of the learning model.

Types of gradient descent:

① Batch Gradient Descent: (All in one go)

→ All training examples are processed in single iteration.

→ If the no. of training examples is large, then batch gradient descent is computationally expensive & not preferred.

② Stochastic Gradient Descent (one by one)

→ It processes 1 training example each iteration. Hence, the parameters is updated even after one iteration. Hence, this is quite faster than batch gradient.

→ But, when training examples are high even then it processes only one example which can be additional overhead of the system as the no. of iterations will be quite large.

③ Mini-Batch G.D.: (in batches) Best

→ It works faster than both batch & stocha

→ Here, b. example where b < m are processed per iteration.

→ So, even if the no. of ~~iteration~~ training example is large, it is processed in batches of b training examples in one go

Conclusion: Mini Batch G.D. works for larger training examples & that too with lesser no. of iterations.

Convergence trends in diff. variants of Gradient Descents:

① In case of Batch Gradient Descent (all in one go), the algorithm follows a straight path towards the minimum. If the
↳ cost function is convex, then it converges to global min. &
↳ if the C.F. is not convex, then it converges to a local min. Here, the learning rate is typically held constant.

② In case of SGD (one by one) & mini-gradient descent (in batches), the algo. does not converge but keeps on fluctuating around the global minimum. Therefore, in order to make it converge, we have to slowly change the learning rate.
(However, the convergence of SGD is much noisier as in one iteration, it processes only one training example.)

Optimization techniques for Gradient Descent (Class Notes)

① G.D. is an iterative optimization algo. used to find the min. value for a function.

② The general idea is to ~~minimize~~ initialize the parameters to random values, and then take small steps in the dir^n of slope at each iteration.

③ G.D. is highly used in supervised learning to min. the error function & find the optimal value of para

Optimization techniques for G.D.: 1. Momentum method 2. RMS prop 3. Adam optimization

Various extensions have been designed for the gradient descent algo.:

① Momentum method: this method is used to accelerate the G.D. algo. by taking into consideration the exponentially weighted average of the gradients.
Using averages makes the algo. converges towards the minima in a faster way, as the gradients towards the uncommon dir^n's are cancelled out.

Pseudocode:

$V = 0$

for each iteration i:

compute dW

$V = \beta V + (1-\beta)\, dW$

$W = W - \alpha V$

→ V & dW are analogous to velocity & accelerations.

→ α is learning rate

→ β is momentum. generally kept 0.9.

RMSprop: The intitution of this method is to apply an exponentially weighted average method to the second moment of the gradient dW^2.

Pseudocode:

$S = 0$

for each iteration i:

compute dW

$S = \beta S + (1-\beta)\, dW^2$

$W = W - \alpha \left(dW / \sqrt{S} + \varepsilon \right)$

$\varepsilon = 10^{-8}$

Adam optimization: this algo incorporates the momentum method & RMSprop along with bias correction.

Pseudo: $V = 0$, $S = 0$

for each iteration i:

compute dW

$V = \beta_1 S + (1-\beta_1)\, dW$, $S = \beta_2 S + (1-\beta_2)\, dW^2$

$V = V / [1-\beta_1^i]$, $S = S / [1-\beta_2^i]$,

$W = W - \alpha V / \sqrt{S} + \varepsilon$

where;

$\beta_1 = 0.9$

6 More Limitation of G.D.d.