ოპტიმიზაციის ტექნიკები

ლაშა ფანცხავა

ლექციის გეგმა

1. Gradient Descent:

Batch Gradient Descent (BGD): ყოველ იტერაციაზე გრადიენტი გამოითვლება მთელი ტრენინგ ნაკრებისთვის

Stochastic Gradient Descent (SGD): ითვლება ერთ ცალ საწვრთნელ წერტილზე. არის გაცილებით ჩქარი თუმცა ფლუქტუაციაა ბევრი

Mini-batch Gradient Descent: გრადიენტი ითვლება ერთ ცალ ბაჩზე და რაღაც მხრის არის შუალედური რგოლი SGD-სა და BGD-ს შორის

2. Momentum:

Momentum: აჩქარეს გრადიენტის ვექტორს მინიმუმისაკენ

Nesterov Accelerated Gradient (NAG): ერთგვარი წინასწარმეტყველება მინიმუმისკენ მიდის თუ არა ფუნქცია

3. Adaptive Learning Rate Methods:

Adagrad: სწავლის სიჩქარის მოდიფიცირება იმის მიხედვით თუ როგორ იცვლება გრადიენტი

RMSprop: მონოტონურ სწავლის სიჩქარის შემცირებას ცვლის

Adam (Adaptive Moment Estimation): კომბინაცია ზედა ორის.

Batch Gradient Descent

BGD (Batch Gradient Descent) არის გრადიენტული დაშვების ერთ-ერთი ვარიანტი, სადაც ოპტიმიზაციის ყოველ იტერაციაზე გამოითვლება გრადიენტი მთელი საწვრთნელი მონაცემთა ნაკრებისთვის.

პარამეტრები დროის მომენტში t+1 გამოითვლება შემდეგი ფორმულით:

$$W_{t+1} = W_t - \eta \nabla J(W_t)$$

რომელშიც:

- η არის სასწავლო სიჩქარე (learning rate)
- $\nabla J(W_t)$ არის ფუნქციის გრადიენტი პარამეტრების W(t) მიმართ

გრადიენტის გამოთვლა ხდება შემდეგნაირად:

$$\nabla J(W) = \frac{1}{m} \sum_{i=1}^{m} \nabla J(W, x_i, y_i)$$

სადაც:

- m არის მონაცემთა ნაკრების ზომა
- x_i არის i-ური სამწვრთნელი ვექტორი შემავალი მონაცემების
- y_i არის i-ური სამწვრთნელი ვექტორი გამომავალი მიზნობრივი მონაცემების
- $\nabla J(W, x_i, y_i)$ არის შეცდომის ფუნქციის გრადიენტი კონკრეტული სამწვრთნელი წყვილისთვის (x_i, y_i)

Batch Gradient Descent

BGD (Batch Gradient Descent) არის გრადიენტული დაშვების ერთ-ერთი ვარიანტი, სადაც ოპტიმიზაციის ყოველ იტერაციაზე გამოითვლება გრადიენტი მთელი საწვრთნელი მონაცემთა ნაკრებისთვის.

პარამეტრები დროის მომენტში t+1 გამოითვლება შემდეგი ფორმულით:

$$W_{t+1} = W_t - \eta \nabla J(W_t)$$

რომელშიც:

- η არის სასწავლო სიჩქარე (learning rate)
- $\nabla J(W_t)$ არის ფუნქციის გრადიენტი პარამეტრების W(t) მიმართ

გრადიენტის გამოთვლა ხდება შემდეგნაირად:

$$\nabla J(W) = \frac{1}{m} \sum_{i=1}^{m} \nabla J(W, x_i, y_i)$$

სადაც:

- m არის მონაცემთა ნაკრების ზომა
- x_i არის i-ური სამწვრთნელი ვექტორი შემავალი მონაცემების
- y_i არის i-ური სამწვრთნელი ვექტორი გამომავალი მიზნობრივი მონაცემების
- $\nabla J(W, x_i, y_i)$ არის შეცდომის ფუნქციის გრადიენტი კონკრეტული სამწვრთნელი წყვილისთვის (x_i, y_i)

Momentum

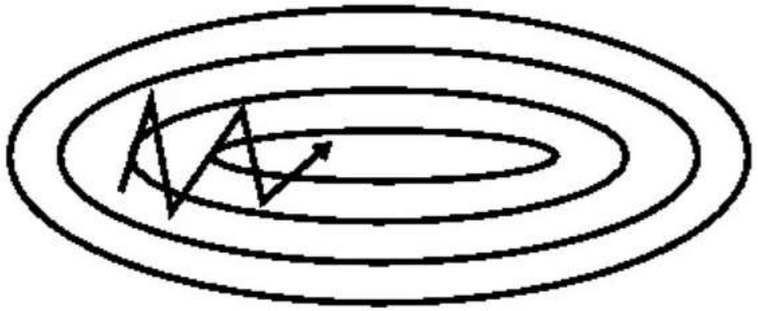
მომენტუმი არის ტექნიკა, რომელიც დამატებულია სტოქასტური გრადიენტული დაშვების ალგორითმში გასაუმჯობესებლად ოპტიმიზაციის პროცესის სტაბილურობისა და კონვერგენციის სიჩქარის. იგი წარმოადგენს ინერციის დამატებას გრადიენტული განახლების მიმართულებაში, რაც აძლიერებს ძლიერ გრადიენტებს და ამცირებს ოსცილაციებს პროცესის დროს.

$$v_{t+1} = \gamma v_t + \eta \nabla J(W_t, x_i, y_i)$$

$$W_{t+1} = W_t - v_{t+1}$$

SGD

მომენტუმის გარეშე



მომენტუმთან ერთად

NAG Momentum

NAG-ის შემთხვევაში კი სხვაობაა იმაში, რომ გრადიენტი გამოითვლება არა მიმდინარე პარამეტრების W_t მიხედვით, არამედ მომენტუმის გათვალისწინებით გადანაცვლებული პარამეტრების მიხედვით:

$$v_{t+1} = \gamma v_t + \nabla J(W_t + \gamma v_t)$$
$$W_{t+1} = W_t - v_{t+1}$$

NAG-ს უპირატესობა ისაა, რომ ახდენს გრადიენტის მიახლოებით გათვლას მომავალ პარამეტრებზე და არა მიმდინარეზე. ეს ეხმარება ალგორითმს უკეთ გაითვალისწინოს "მომენტუმის" გავლენა და შეამციროს რხევები კონვერგენციის პროცესში.

NAG ხშირად უფრო სწრაფადაც კონვერგირდება ვიდრე ჩვეულებრივი მომენტუმი, თუმცა მისი იმპლემენტაცია შედარებით უფრო რთულია.

Adagrad

Adagrad (Adaptive Gradient Algorithm) არის გრადიენტული დაშვების ოპტიმიზაციის ალგორითმი, რომელიც ავტომატურად მუშავდება სასწავლო მანძილის მორგებაზე თითოეული პარამეტრისთვის.

$$g_{t+1} = g_t + \nabla J(W_t)^2$$
$$W_{t+1} = W_t - \frac{\eta}{\sqrt{g_{t+1} + \epsilon}} \nabla J(W_t)$$

სადაც:

- g(t) არის კვადრატული გრადიენტების კუმულაციური ჯამი დროის მომენტში t
- η არის სასწავლო სიჩქარე
- ε არის პატარა რიცხვი თავიდან ავიცილოთ არნულირება

თუ გრადიენტი რომელიმე პარამეტრისთვის დიდია, მაშინ კვადრატული გრადიენტი კუმულაციური ჯამისთვის იქნება დიდი და გრადიენტის განახლება ამ პარამეტრისთვის შემცირდება.

მეორე მხრივ, თუ გრადიენტი რომელიმე პარამეტრისთვის პატარაა, მაშინ მისი კუმულაციური ჯამიც იქნება პატარა და განახლება ამ პარამეტრისთვის გაიზრდება.

ამრიგად, Adagrad ავტომატურად ამცირებს მანძილის სიდიდეს "ხმაურიანი" პარამეტრებისთვის და ზრდის მას "რბილი" პარამეტრებისთვის დროის განმავლობაში.

RMSprop

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{S_{dW}+\epsilon}} dW$$

$$S_{dW_{t+1}} = \beta S_{dW_t} + (1 - \beta) dW^2$$

სადაც:

- S_{dW} არის კვადრატული გრადიენტების ექსპონენციალური სარგებლიანი სათანადო საშუალო დროის მომენტში t
- β არის ექსპონენციალური შეწონვის კოეფიციენტი (0.9 ტიპიურად)
- η არის სასწავლო სიჩქარე
- ε არის პატარა რიცხვი თავიდან ავიცილოთ არნულირება

განსხვავება Adagrad-თან არის ის, რომ RMSprop არ აგროვებს გრადიენტების კვადრატულ ჯამს დროის მანძილზე, არამედ ინახავს მათ ექსპონენციალურ სარგებლიან სათანადო საშუალოს წინა გრადიენტების კვადრატებზე. ეს ნიშნავს, რომ უფრო ახლო გრადიენტებს აქვთ უფრო დიდი გავლენა ვიდრე ძველებს.

β კოეფიციენტი კონტროლდება გრადიენტების ისტორიის ხანგრძლივობას. უფრო დიდი β ნიშნავს უფრო გრძელვადიან ისტორიას, ხოლო უფრო პატარა β - უფრო მოკლევადიანს.

Adam

$m_{t+1} = \beta_1 m_t + (1 - \beta_1)\nabla J(W_t)$ $v_{t+1} = \beta_2 v_t + (1 - \beta_2)\nabla J(W_t)^2$

$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1}$

$\hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2}$

$$W_{t+1} = W_t - \frac{\eta \hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}+\epsilon}}$$

სადაც:

- m(t) და v(t) არის გრადიენტისა და კვადრატული გრადიენტის ექსპონენციალური სარგებლიანი საშუალოები შესაბამისად
- β1 და β2 არის ექსპონენციალური შეწონვის კოეფიციენტები (ტიპიურად 0.9 და 0.999)
- η არის სასწავლო სიჩქარე
- ε არის პატარა რიცხვი თავიდან ავიცილოთ არნულირება

როგორც ვხედავთ, Adam-ი პირველ ორ განტოლებაში ახდენს მომენტუმისა და RMSprop-ის შეფასებებს. შემდეგ მესამე და მეოთხე განტოლებები ახდენენ ამ შეფასებების კორექტირებას იმის გამო, რომ საწყისი შეფასებები იქნება არაზუსტი. მეხუთე განტოლება კი წარმოადგენს ფაქტობრივ პარამეტრების განახლებას.