深層学習のための最適化

2021/5/17 研究室勉強会

工学院大学

目的

- 最適化問題の基礎知識およびその枠組みを知り，今回扱う最適化アルゴリズムの位置づけを理解する．（知ってる人は復習）
- それぞれの連続最適化アルゴリズムおよびそれらの特性を知る．
- 連続最適化アルゴリズムを，ニューラルネットワークの学習に適用する際に，どういった点に気を付ければよいかを理解する．

前提知識：誤差逆伝播法，最急降下法，勾配

主な参考書籍

「しっかり学ぶ数理最適化」 梅谷俊治・著． 講談社．

https://www.kspub.co.jp/book/detail/5212707.html

「これならわかる深層学習入門」（機械学習スタートアップシリーズ） 瀧雅人・著． 講談社．

https://www.kspub.co.jp/book/detail/1538283.html

目次

最適化とは問題解決を実現する手段の一つである

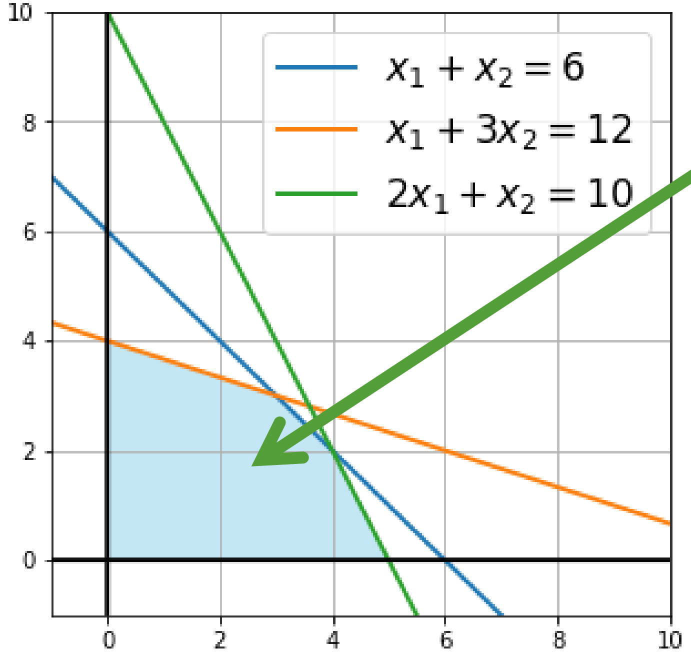
数理最適化(mathematical optimization)とは

与えられた制約条件の下で目的関数の最小(もしくは最大)
にする解を求める最適化問題を通じて,
現実社会における意思決定や問題解決を実現する手段である. [梅谷, 2020, p.1]

$$\underset{x}{\text{Minimize}} \quad x_1 + 2x_2$$ 目的関数

$$\text{subject to} \quad \left.\begin{aligned} x_1 + x_2 &\le 6 \\ x_1 + 3x_2 &\le 12 \\ 2x_1 + x_2 &\le 10 \\ x_1, x_2 &\ge 0 \end{aligned}\right\}$$ 制約条件

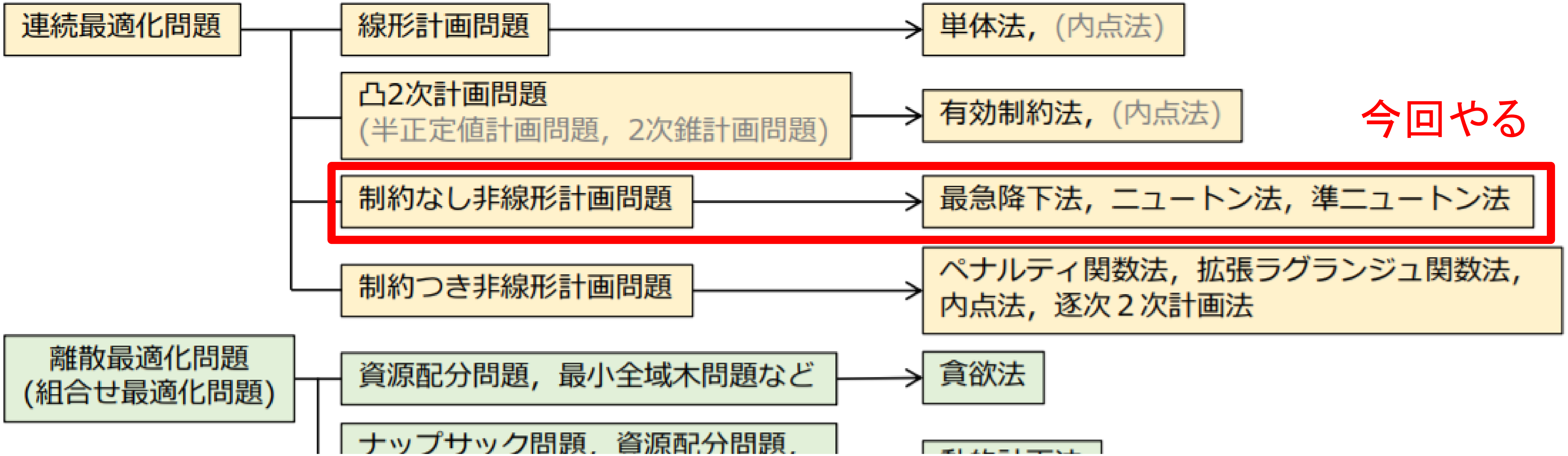
実行可能解：制約条件を満たす解



実行可能領域 S:
実行可能解の集合

最適解:
$\forall \boldsymbol{x} \in S$ に対して
$f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$を満たす
実行可能解 $\boldsymbol{x}^* \in S$

最適化問題の枠組み



今回やる

出典：https://speakerdeck.com/umepon/mathematical-optimization-in-60-minutes?slide=13

線形計画問題：目的関数が線形関数で，制約条件が線形等式or線形不等式

凸2計画問題：目的関数が2次関数で，制約条件が線形等式or線形不等式

非線形計画問題：非線形関数で表された目的関数や制約条件を含む最適化問題

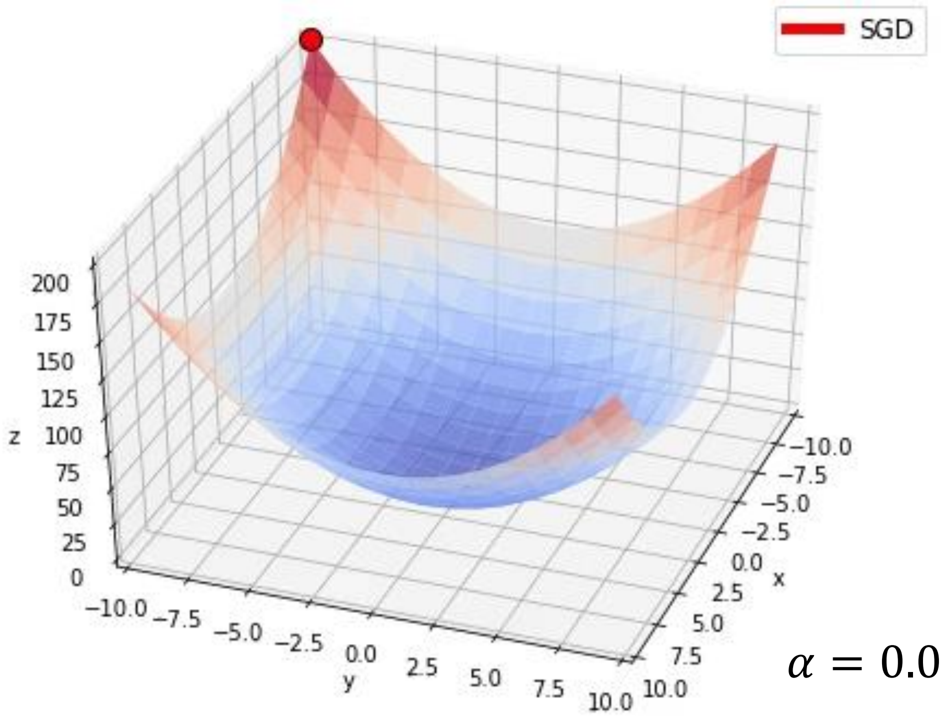
最急降下法

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$\Delta\boldsymbol{\theta}^{(t)} = -\alpha\nabla E\left(\boldsymbol{\theta}^{(t)}\right)$$

α : ステップ幅

$$\nabla E\left(\boldsymbol{\theta}^{(t)}\right) = \frac{\partial E\left(\boldsymbol{\theta}^{(t)}\right)}{\partial\boldsymbol{\theta}^{(t)}}$$ （ステップtにおける勾配）



$\alpha = 0.01$

ミニバッチ学習と確率的勾配降下法

最急降下法の問題点1

局所的な最小値に陥ってしまい，大域的な最小値に達することができない場合がある.

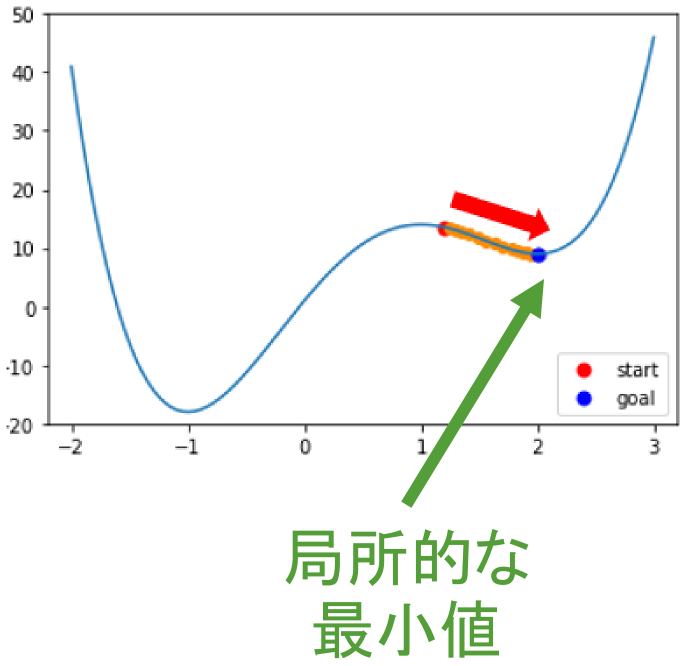
学習するデータを確率的に選ぶことで，臨界点（勾配が0になる点）に陥ることを避ける.

ミニバッチ$\mathcal{B}^{(t)}$をランダムに抽出し，損失の平均をとる

$$E\left(\boldsymbol{\theta}^{(t)}\right) = \frac{1}{\left|\mathcal{B}^{(t)}\right|} \sum_{n \in \mathcal{B}^{(t)}} E_n\left(\boldsymbol{\theta}^{(t)}\right)$$

$\left|\mathcal{B}^{(t)}\right| = 1$ のとき，「オンライン学習」または「確率的勾配降下法（Stochastic Gradient Descent method, SGD）」と呼ぶ.

$\left|\mathcal{B}^{(t)}\right| > 1$のときはミニバッチSGD



Momentum（momentum）

最急降下法の問題点2

誤差関数が深い谷を作っている場合，その谷に沿って激しく振動して学習が進まない

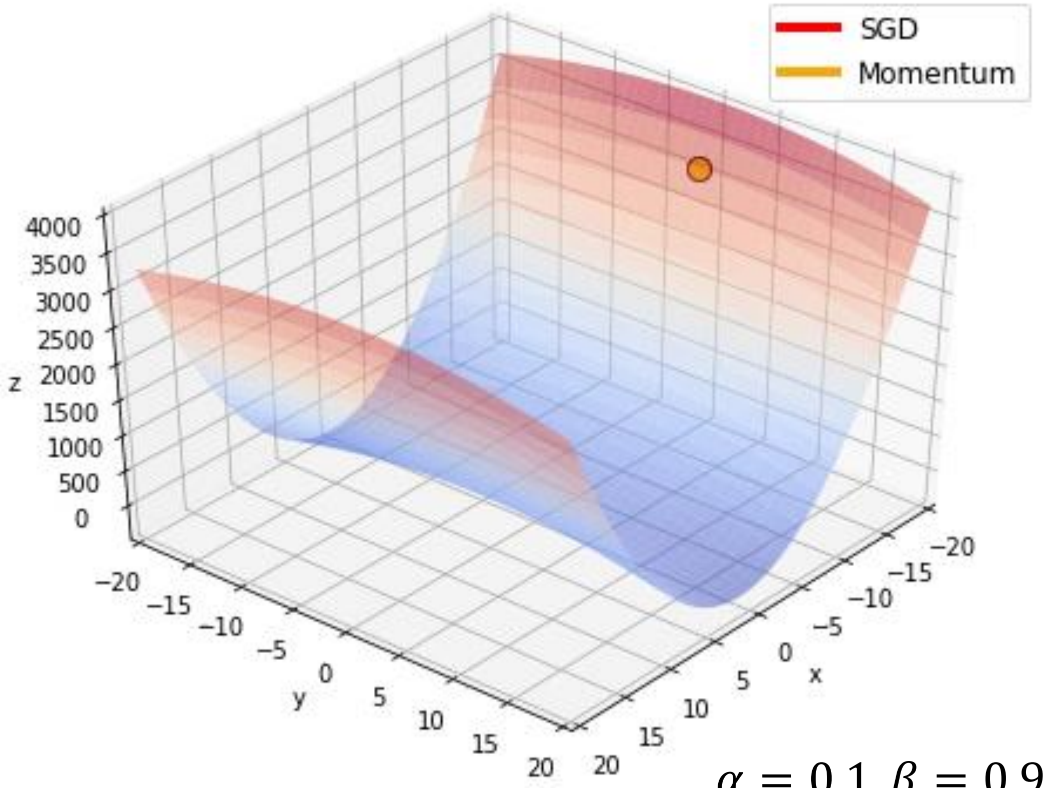
慣性項を加えることで，前時刻での勾配の影響を引きずらせるため，振動を抑える．

ハイパーパラメータ：α, β

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$\Delta\boldsymbol{\theta}^{(t)} = \beta\Delta\boldsymbol{\theta}^{(t-1)} - (1-\beta)\alpha\nabla E\left(\boldsymbol{\theta}^{(t)}\right)$$

$\beta \in [0,1]$であり，$\beta = 0$で勾配降下法と同じ



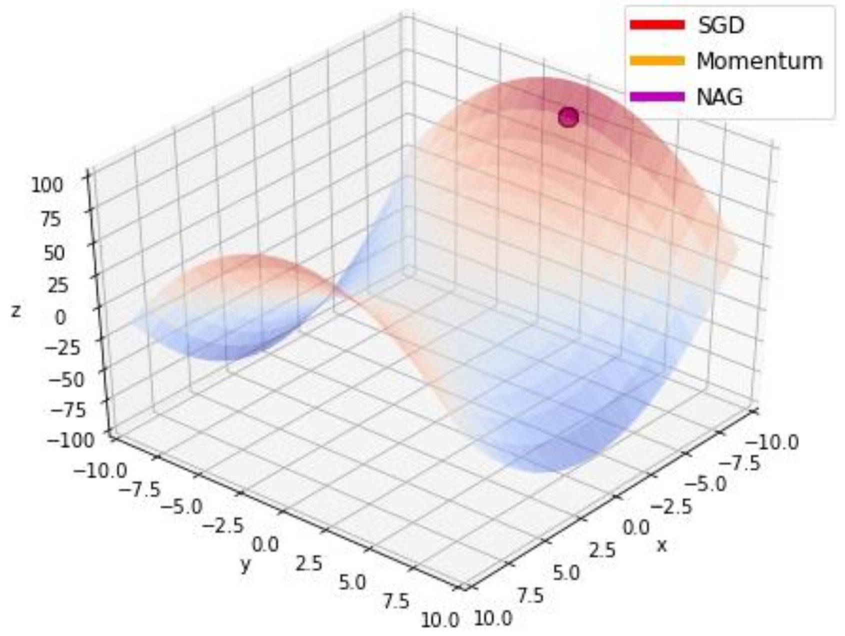
$\alpha = 0.1, \beta = 0.9$

ネステロフの加速勾配法（Nesterov’s accelerated gradient method, NAG）

Momentumの修正版.

更新後のパラメータを大雑把に見積もることで,

勾配が変化する前に, その変化に対応できる.



$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$\Delta\boldsymbol{\theta}^{(t)} = \beta\Delta\boldsymbol{\theta}^{(t-1)} - (1-\beta)\alpha\nabla E\left(\boldsymbol{\theta}^{(t)} + \underline{\beta\boldsymbol{\theta}^{(t-1)}}\right)$$

Momentum, ネステロフの加速勾配法を適用するかはSGDメソッドの引数で選択可

```
keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
```

ステップサイズα

Momentum, NAGの慣性$\beta \in [0, 1]$

$\beta = 0$で勾配降下法と同じ

NAGを適用するか : Trueで適用

出典 : https://keras.io/ja/optimizers/#sgd

AdaGrad（adaptive subgradient descennt）

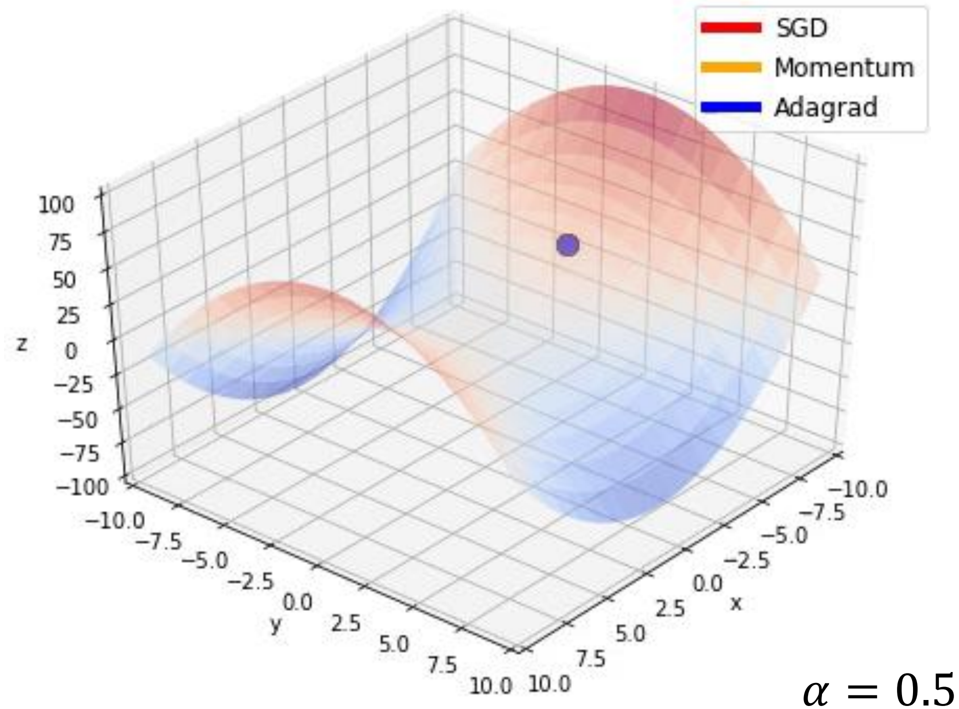
各パラメータ方向に対し，学習率を調整する.

大きな勾配値をとってきた方向→更新量を小さく

小さな勾配値をとってきた方向→更新量を大きく

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$\Delta\boldsymbol{\theta}_i^{(t)} = -\frac{\alpha}{\sqrt{\sum_{s=1}^{t}\left(\nabla E\left(\boldsymbol{\theta}^{(s)}\right)_i\right)^2}}\nabla E\left(\boldsymbol{\theta}^{(t)}\right)_i$$



$\alpha = 0.5$

欠点：

- 初期の勾配が大きいと更新量が小さくなってしまう.
- 更新量が下がってしまうと上げることはできない.

RMSprop

- 指数的な移動平均から決まる root mean square (RMS)を用いることで，AdaGradの欠点を改善.
- 最近の勾配の履歴ほど更新量に影響するようにする.
- ハイパーパラメータ：α, ρ, ϵ

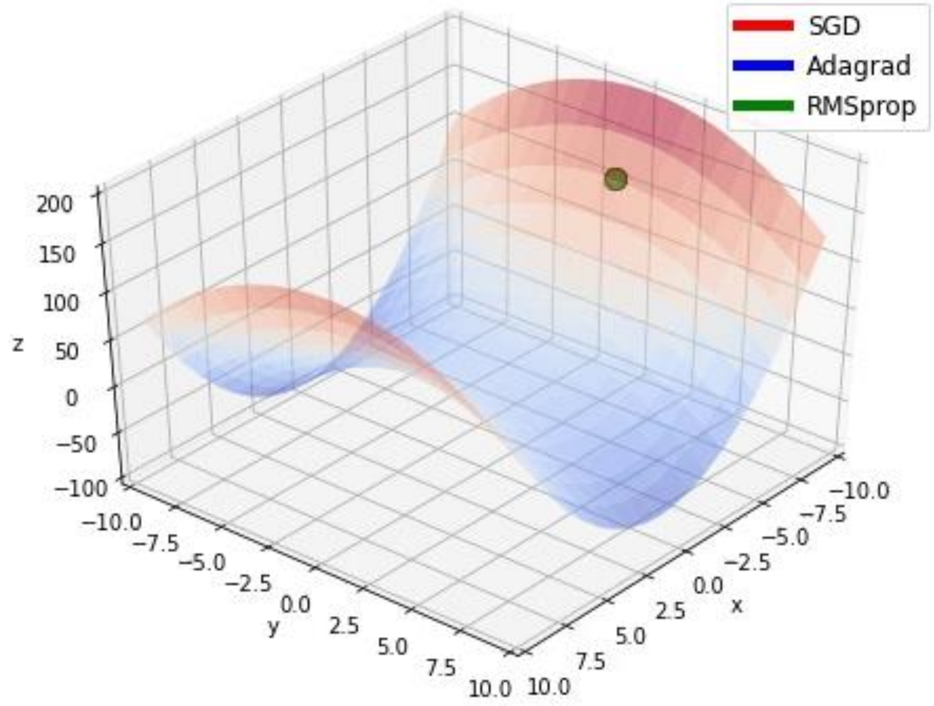
$$\alpha = 0.1$$
$$\rho = 0.99$$
$$\epsilon = 10^{-8}$$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$v_i^{(t)} = \rho v_i^{(t-1)} + (1-\rho)\left(\nabla E\left(\boldsymbol{\theta}^{(t)}\right)_i\right)^2$$

$$v_i^{(0)} = 0$$

$$\Delta\boldsymbol{\theta}_i^{(t)} = -\frac{\alpha}{\sqrt{v_i^{(t)} + \epsilon}}\nabla E\left(\boldsymbol{\theta}^{(t)}\right)_i$$



Adam

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

$$\boldsymbol{g}_i^{(t)} = \nabla E\left(\boldsymbol{\theta}^{(t)}\right)_i, m_0 = 0, v_0 = 0$$

$$m_i^{(t)} = \beta_1 m_i^{(t-1)} + (1-\beta_1)\boldsymbol{g}_i^{(t)}$$

$$v_i^{(t)} = \beta_2 v_i^{(t-1)} + (1-\beta_2)\left(\boldsymbol{g}_i^{(t)}\right)^2$$

$$\hat{m}_i^{(t)} = \frac{m_i^{(t)}}{1-\beta_1^t}, \qquad \hat{v}_i^{(t)} = \frac{v_i^{(t)}}{1-\beta_2^t}$$

$$\Delta\boldsymbol{\theta}_{\boldsymbol{i}}^{(t)} = -\alpha \frac{\widehat{m_i}^{(t)}}{\sqrt{\hat{v}_i^{(t)}} + \epsilon}$$

RMSpropを改良

勾配に対しても指数的な移動平均をとる.

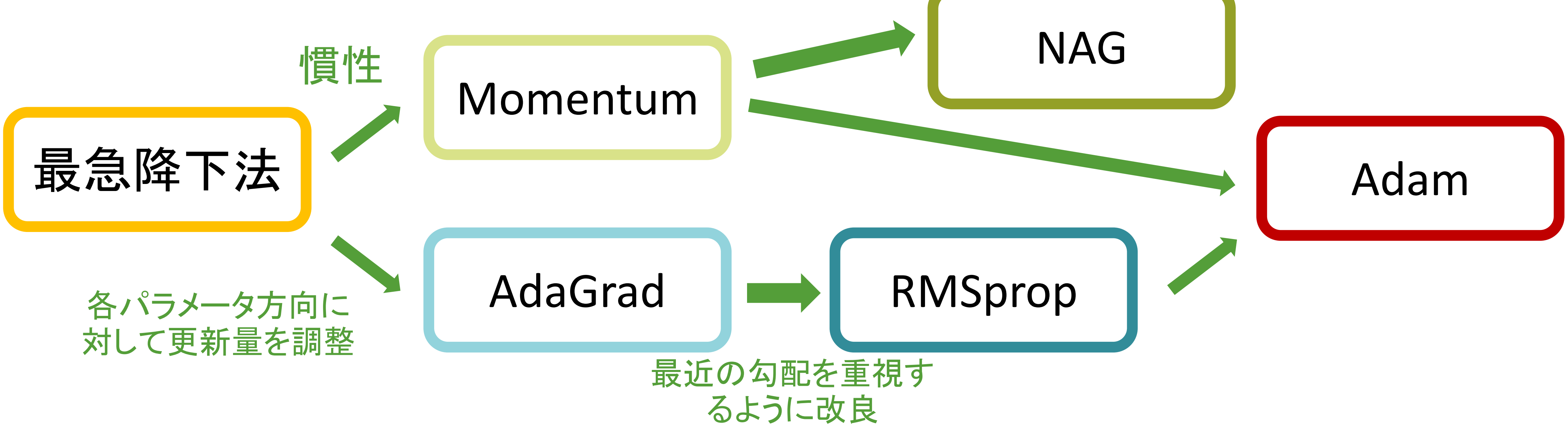
ハイパーパラメータの推奨値:

$\alpha = 0.001,\ \beta_1, = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^{-8}$

論文: https://arxiv.org/abs/1412.6980

まとめ

目的関数が非線形関数である「（制約なし）非線形計画問題」を解くためのアルゴリズムを紹介した



参考文献・関連情報

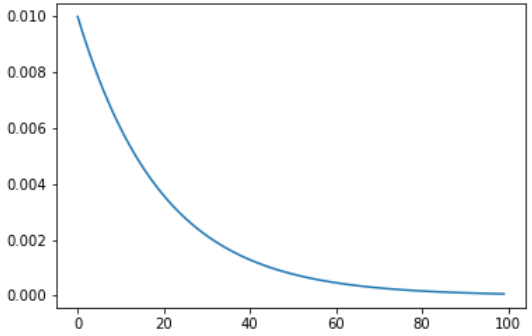
- @T-STAR. (2020年11月18日). 最適化アルゴリズムを単独実行で比較する(総合編). 参照先: Qiita: https://qiita.com/T-STAR/items/c77f5c318d3f3141aad4
- Diederik P. KingmaBaJimmy. (2014). Adam: A Method for Stochastic Optimization.
- omiita. (2021年2月11日). 参照先: Qiita: https://qiita.com/omiita/items/1735c1d048fe5f611f80
- UmetaniShunji. (2020年6月16日). Mathematical Optimization in 60 minutes. 参照先: Speaker Deck: https://speakerdeck.com/umepon/mathematical-optimization-in-60-minutes
- 斎藤康毅. (2016). ゼロから作るDeep Learning.
- 瀧雅人. (2017). これならわかる深層学習入門(機械学習スタートアップシリーズ). 講談社.
- 梅谷俊治. (2020). しっかり学ぶ数理最適化　モデルからアルゴリズムまで. 講談社.

ステップサイズのスケジューリング

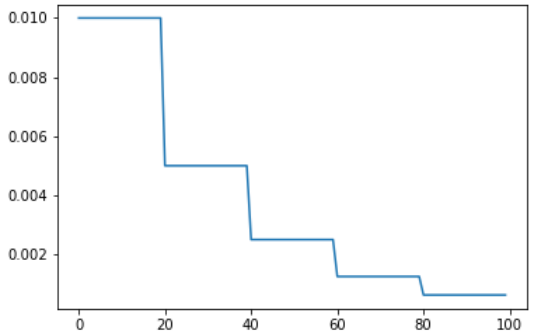
1ステップごとのステップサイズが一定だと，収束したい点に近づけず，収束が遅くなる．

→　時間依存する学習率を用いる．

LambdaLR, StepLRなど様々なステップサイズのスケジューリング方法がある．



LambdaLRを用いたときの学習率



StepLRを用いたときの学習率

出典: https://katsura-jp.hatenablog.com/entry/2019/01/30/183501

参考: https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate