現場で使えるディープラーニング基礎講座

DAY3

SkillUP AI

前回の復習

- 前回は何を学びましたか？

学習の最適化 前半

目次

1. 勾配法の学習を最適化させる方法
2. 重みの初期値
3. 機械学習と純粋な最適化問題の差異
4. ニューラルネットワーク最適化の課題
5. 最適化戦略とメタアルゴリズム

勾配法の学習を最適化させる方法

確率的勾配降下法

- 確率的勾配降下法(Stochastic Gradient Descent, 以下SGD)とは、無作為に選びだしたデータを用いてパラメータを更新していく勾配降下法のこと。
- ミニバッチ学習を行うと、SGDで解いていることになる。
- ミニバッチ学習の特徴については、DAY2の資料を参照。

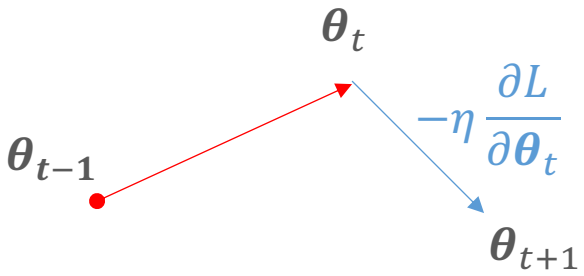
SGDの更新式

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

$\boldsymbol{\theta}$: パラメータ
L : 損失
η : 学習率

ここでのパラメータ$\boldsymbol{\theta}$は、ネットワーク全体の重みwバイアスbなどを並べたベクトルを表す。
また、tは時点を表す。

SGD



勾配法の学習を最適化させる方法

- SGDの学習をより効率的に行う方法があり、それらは1次の方法と2次の方法に分類される。
- 一般的に、DNNでは1次の方法が用いられるが、2次の方法を用いている事例もある。
  - 1次の方法
    - 1階の微分のみを用いる。
    - 学習率を事前に決める必要がある。
  - 2次の方法
    - 1階の微分と2階の微分の両方を用いる。
    - 1次の方法のように学習率を事前に決める必要がない。
    - 2階の微分が学習率と同様の働きをする。
    - 2階の微分の行列(ヘッセ行列)の逆行列を計算する必要があるため、計算負荷が大きい。

勾配法の学習を最適化させる方法

1次	Momentum	物理法則に準じる動きをする。進む方向と勾配方向が同じであれば、移動量が大きくなる。
	Nesterov AG	Momentumの改良版。次の位置を予測し、その位置の勾配も加味する。
	AdaGrad	学習が進むにつれ、見かけの学習率が減衰していく。パラメータ毎に見かけの学習係数が異なる。
	AdaDelta	AdaGradの改良版。過去の勾配情報よりも直近の勾配情報をより重視する。
	RMSProp	AdaDeltaの簡易版。過去の勾配情報よりも直近の勾配情報をより重視する。
	Adam	RMSPropとMomentumを組み合わせたような方法
2次	Newton法	ヘッセ行列を使用
	共役勾配法	共役方向によって逆ヘッセ行列の計算を効率化する
	BFGS	各ステップでのヘッセ行列の近似を改良したもの
	L-BFGS	省メモリ化したBFGS

Momentum

- 物理法則に準じる動きをする。
- 真っ直ぐに進んでいるときは移動量が大きくなり、曲がりながら進んでいるときは緩やかに曲がる。

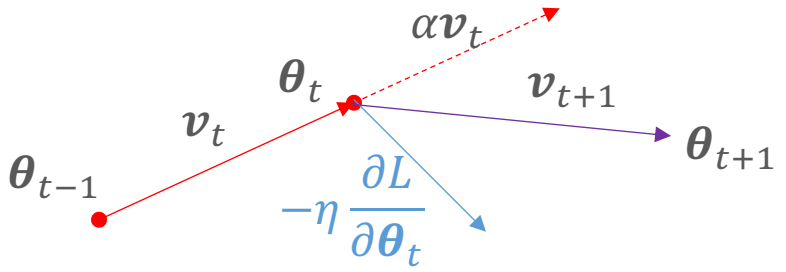
Momentumの更新式

$$\boldsymbol{v}_{t+1} = \alpha \boldsymbol{v}_t - \eta \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_{t+1}$$

$\boldsymbol{\theta}$: パラメータ
L : 損失
η : 学習率
α: モーメンタム係数(0以上1未満)

Momentum



1期前の更新量$\boldsymbol{v}_t$を利用する

Nesterov Accelerated Gradient

- Momentumの改良版。
- 仮の1期先の位置を求め、その位置の勾配を利用する。

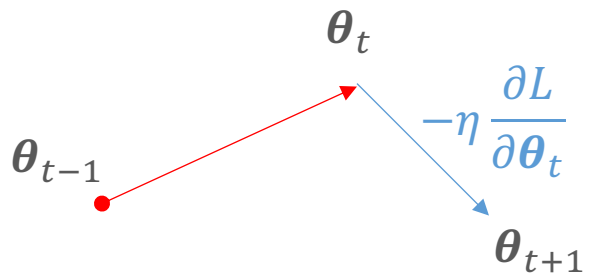
Nesterov Accelerated Gradientの更新式

$$\boldsymbol{v}_{t+1} = \alpha \boldsymbol{v}_t - \eta \frac{\partial L}{\partial(\boldsymbol{\theta}_t + \alpha \boldsymbol{v}_t)}$$

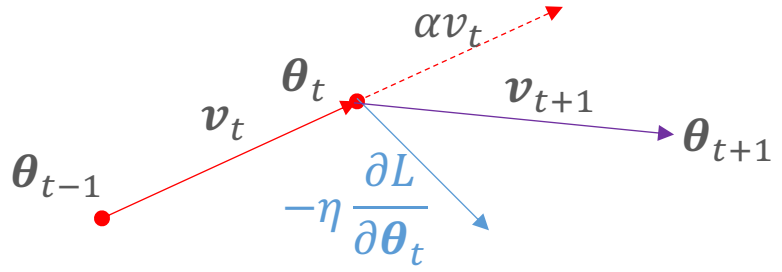
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_{t+1}$$

$\boldsymbol{\theta}$: パラメータ
L : 損失
η : 学習率
α: モーメンタム係数(0以上1未満)

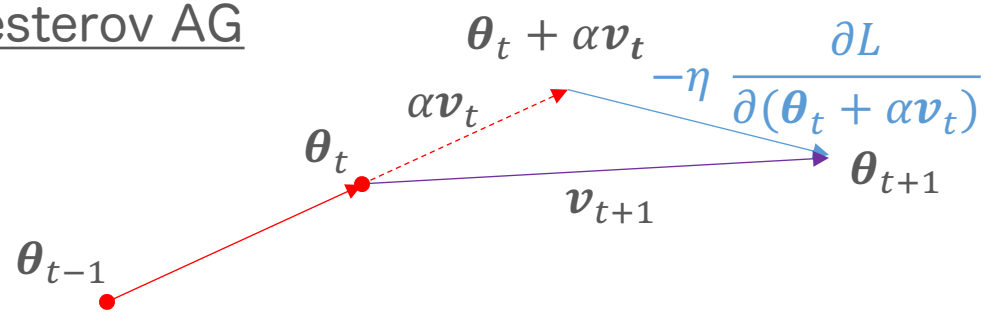
SGD



Momentum



Nesterov AG



暫定的な1期先の勾配を利用することにより、学習を加速させようというアイデア

Nesterov Accelerated Gradient

- Nesterov AGは、定義式通りに計算しようとすると、暫定的1期先の勾配が必要になり、計算が煩雑になる。
- Yoshua Bengioらが効率よく計算する方法を提案しており、kerasやchainerはこの方法を採用している。
  - https://arxiv.org/pdf/1212.0901v2.pdf

Nesterov Accelerated Gradient

Nesterov Accelerated Gradientの更新式(実装版)

$\Theta_{t+1} = \boldsymbol{\theta}_{t+1} + \alpha \boldsymbol{v}_{t+1}$とおくと、
$\Theta_t = \boldsymbol{\theta}_t + \alpha \boldsymbol{v}_t$なので、定義式は以下のように変形できる

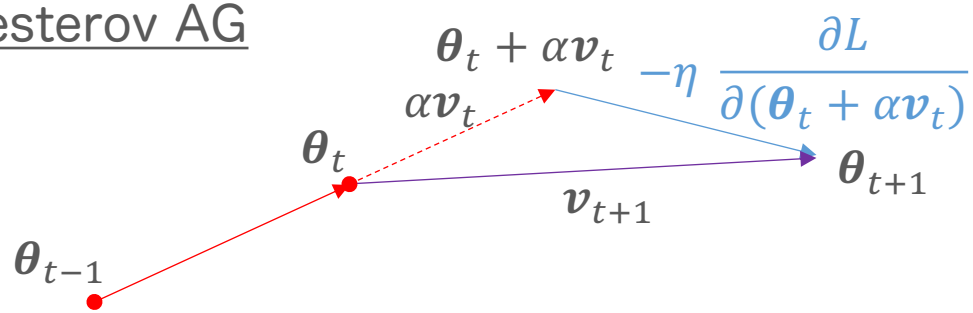
$$\boldsymbol{v}_{t+1} = \alpha \boldsymbol{v}_t - \eta \frac{\partial L}{\partial \Theta_t}$$

$$\begin{aligned}\Theta_{t+1} &= \Theta_t - \alpha \boldsymbol{v}_t + \alpha \boldsymbol{v}_{t+1} + \boldsymbol{v}_{t+1} \\ &= \Theta_t - \alpha \boldsymbol{v}_t + (\alpha + 1)\boldsymbol{v}_{t+1} \\ &= \Theta_t - \alpha \boldsymbol{v}_t + (\alpha + 1)\left(\alpha \boldsymbol{v}_t - \eta \frac{\partial L}{\partial \Theta_t}\right) \\ &= \Theta_t + \alpha\alpha \boldsymbol{v}_t - (\alpha + 1)\left(\eta \frac{\partial L}{\partial \Theta_t}\right) \\ &= \Theta_t + \alpha\left(\alpha \boldsymbol{v}_t - \eta \frac{\partial L}{\partial \Theta_t}\right) - \eta \frac{\partial L}{\partial \Theta_t}\end{aligned}$$

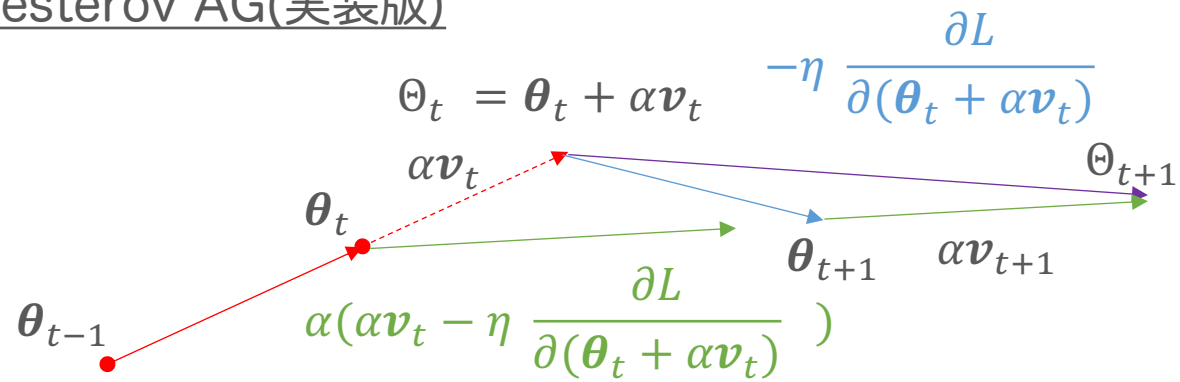
暫定的1期先のパラメータを更新するように変形することで、定義式と等価な計算を実現している

$\boldsymbol{\theta}$：パラメータ
L：損失
η：学習率
α: モーメンタム係数(0以上1未満)

Nesterov AG



Nesterov AG(実装版)



※Θもベクトル

$\boldsymbol{\theta}$を更新するのではなく、暫定的1期先のパラメータΘを更新していく

AdaGrad

- パラメータ毎に学習率を適応させる(adaptive)方法。
- 学習が進むにつれ、見かけの学習率が減衰していく。

AdaGradの更新式

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t + \frac{\partial L}{\partial \boldsymbol{\theta}_t} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \underbrace{\frac{1}{\varepsilon + \sqrt{\boldsymbol{h}_{t+1}}}}_{\text{見かけの学習率}} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

$\boldsymbol{\theta}$: パラメータ
L : 損失
η : 学習率
ε : 計算を安定化させるための係数
$\odot$: アダマール積記号(同じ要素同士の掛け算)

$\boldsymbol{h}$はベクトルであることに注意
計算が進むにつれ、パラメータ毎の見かけの学習率が小さくなっていく

AdaDelta

- AdaGradの改良版。
- 勾配の2乗の移動平均と、更新量の移動平均を用いて、見かけの学習率を変化させていく。
- 移動平均をとると、過去の情報が少しずつ薄れていき、新しい情報の影響がより大きくなっていく。

AdaDeltaの更新式

$$\boldsymbol{h}_{t+1} = \rho \boldsymbol{h}_t + (1-\rho)\frac{\partial L}{\partial \boldsymbol{\theta}_t} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$ 勾配の2乗の移動平均

$$\Delta \boldsymbol{\theta}_t = -\frac{\sqrt{\varepsilon + \boldsymbol{r}_t}}{\sqrt{\varepsilon + \boldsymbol{h}_{t+1}}} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$ 更新量

見かけの学習率

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta \boldsymbol{\theta}_t$$

$$\boldsymbol{r}_{t+1} = \rho \boldsymbol{r}_t + (1-\rho)\Delta \boldsymbol{\theta}_t \odot \Delta \boldsymbol{\theta}_t$$ 更新量の移動平均

$\boldsymbol{\theta}$: パラメータ
L : 損失
ρ : 減衰率, 0.95など
ε : 計算を安定化させるための係数
$\odot$: アダマール積記号(同じ要素同士の掛け算)

ここでの移動平均とは、指数平滑化移動平均のこと。減衰率ρの割合で足していくことによって、過去の情報が指数関数的に薄れていく。

RMSProp

- AdaDeltaの簡易版。
- 勾配の2乗の移動平均を用いて、見かけの学習率を変化させていく。
- 移動平均をとると、過去の情報が少しずつ薄れていき、新しい情報が反映されていく。

RMSPropの更新式

$$\boldsymbol{h}_{t+1} = \rho \boldsymbol{h}_t + (1-\rho)\frac{\partial L}{\partial \boldsymbol{\theta}_t} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{1}{\sqrt{\varepsilon + \boldsymbol{h}_{t+1}}} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$

見かけの学習率

$\boldsymbol{\theta}$: パラメータ
L : 損失
ρ : 減衰率, 0.9など
η : 学習率
ε : 計算を安定化させるための係数
$\odot$: アダマール積記号(同じ要素同士の掛け算)

ここでの移動平均とは、指数平滑化移動平均のこと。減衰率ρの割合で足していくことによって、過去の情報が薄れていく。

Adam

- RMSPropとMomentumを組み合わせたような方法

Adamの更新式

$$\boldsymbol{m}_{t+1} = \rho_1 \boldsymbol{m}_t + (1-\rho_1)\frac{\partial L}{\partial \boldsymbol{\theta}_t}$$ Momentumに似た部分

$$\boldsymbol{v}_{t+1} = \rho_2 \boldsymbol{v}_t + (1-\rho_2)\frac{\partial L}{\partial \boldsymbol{\theta}_t} \odot \frac{\partial L}{\partial \boldsymbol{\theta}_t}$$ RMSPropに似た部分

$$\hat{\boldsymbol{m}}_{t+1} = \frac{\boldsymbol{m}_{t+1}}{1-\rho_1^t}$$

$$\hat{\boldsymbol{v}}_{t+1} = \frac{\boldsymbol{v}_{t+1}}{1-\rho_2^t}$$

バイアス修正式(学習初期の計算を安定化させるのが目的)

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \frac{1}{\sqrt{\hat{\boldsymbol{v}}_{t+1}} + \varepsilon} \odot \hat{\boldsymbol{m}}_{t+1}$$

見かけの学習率

$\boldsymbol{\theta}$：パラメータ
L：損失
ρ_1：減衰率, 0.9など
ρ_2：減衰率, 0.999など
η：学習率
ε：計算を安定化させるための係数
$\odot$：アダマール積記号(同じ要素同士の掛け算)

[演習] 勾配法の学習を最適化させる手法の実装

- 3_1_SGD_trainee.ipynb
  - SGDおよびその学習を最適化させる手法について、NNでの利用を想定したクラスを実装しましょう。

[演習] 確率的勾配降下法などを用いたNNの実装

- 3_2_SGD_withNN_trainee.ipynb
  - 確率的勾配降下法などを用いたNNを実装しましょう。

SGD更新式の別の表現

- SGD更新式の表現は、書籍によって異なるので注意すること。
- 例えば、Goodfellow著の深層学習では以下のように表現されている。

学習率ε

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\epsilon}{m} \nabla_{\boldsymbol{\theta}} \sum_i L\left(f\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \boldsymbol{y}^{(i)}\right)$$

勾配　損失関数　予測結果　正解データ

データ数mで割って損失を平均化

[グループワーク]

- 3_1_SGD_trainee.ipynbの最下部に記載している演習にとりくみます。
  - 学習率やモーメンタム係数を変更して、結果がどのように変わるか確認しましょう。
  - 手法、学習率、係数をいろいろ変化させて、収束までのステップ回数を比較してみましょう。一覧表をつくると比較しやすいでしょう。
- 2~3名のグループに分かれて、上記テーマに取り組んでみましょう。(45分)
- 最後に、グループごとに発表していただきます。 (15分)

[グループワーク]

- 3_2_SGD_withNN_trainee.ipynbの最下部に記載している演習にとりくみます。
  - 最適化手法を変更し、結果がどのように変わるかを確認しましょう。
- 2~3名のグループに分かれて、上記テーマに取り組んでみましょう。(15分)
- 最後に、グループごとに発表していただきます。 (15分)

Any Questions ?

重みの初期値

重みの初期値

- ここでは、重み(パラメータW)の初期値について、どのように設定するのが適正かを考える
- 良い初期値の条件とはなにか？

重みの初期値

- 例えば、重みの初期値をある値(０とか１)に固定するとどうなるか？
  - ノードを沢山設定することの意義は、それぞれのノードに別々の役割を担わせることができること。これにより、複雑な関係性を表現することができる。
  - もし、ノードの出力値が同じになるのであれば、それは、ノードを1つだけ配置していることと同じ意味になる。
  - つまり、初期値がバラけることによって、ノードを沢山設定する意味が出てくる。
- 逆に、重みの初期値を適当に大きくしたり小さくしたりするとどうなるか？
  - 重みの初期値が0から離れると、計算が収束するまでにより多くの時間がかかってしまうことが考えられる。重みの初期値は、最終的に求まる値に近い方が計算の収束が早いはず。 NNの計算ではデータを標準化することが多く、標準化すると入力層に入力される値は0付近に分布することになる。この場合、最終的に求まる重みの値も0付近に分布することが多い。
  - 重みの初期値を0に近づけすぎると、ノードの多様性がなくなる。
- これより、「重みの初期値は、0付近で適度なバラつきをもっていると良さそうである」と言える。
- では、適度にバラけさせるにはどうすればいいか？

重みの初期値

n_1 は前の層のノード数
n_2 は後ろの層のノード数

- Xavierの初期値
  - 初期値を適度にバラつかせる方法として、Xavier Glorotらが提案する方法がある。
  - Xavierの方法では、$\sqrt{\frac{2}{n_1+n_2}}$を標準偏差とする。
  - sigmoid関数やtanh関数のように点対称で中央付近で線形関数としてみなせる活性化関数に向いている。
  - http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf
- Heの初期値
  - 初期値を適度にバラつかせる方法として、Kaming Heらが提案する方法がある。
  - Heの方法では、$\sqrt{\frac{2}{n_1}}$を標準偏差とする。
  - Xavierの初期値にくらべ、広がりを持った初期値になる。
  - ReLU関数を用いる場合に向いている。
  - https://arxiv.org/pdf/1502.01852.pdf

[演習] 初期値の影響の確認

- 3_3_initial_value_fixed.ipynb
  - 重みの初期値(固定値)の影響を確認しましょう。
- 3_4_initial_value_random_trainee.ipynb
  - 重みの初期値(ランダム値)の影響を確認しましょう。

[演習] 初期値の実装

- 3_5_initial_value_Xavier_trainee.ipynb
  - Xavierの初期値を利用するための関数を実装しましょう。
- 3_6_initial_value_He_trainee.ipynb
  - Heの初期値を利用するための関数を実装しましょう。

[演習] ニューラルネットワークにおける初期値の影響

- 3_7_two_layer_NeuralNetwork_regression_trainee.ipynb
  - パラメータの初期値を変えながら、2層ニューラルネットワークで単純な回帰問題を解いてみましょう。

Any Questions ?

機械学習と純粋な最適化問題の差異

機械学習と純粋な最適化問題の差異

- 機械学習における学習とは、本質的には最適化問題を解くことである。
- では、機械学習における最適化は、純粋な最適化問題とどのように異なるのか？

純粋な最適化問題の例

- 資源配分問題
  - 限られた資源で最大の収益を得るにはどうすればいいか？
- 車体の形状最適化問題
  - 風の抵抗を最小にするにはどうすればいいか？
- 最短ルート探索問題
  - 目的地まで最も早く辿り着くにはどの道を通ればよいか？

機械学習における最適化の特徴

- 機械学習では極小点が最適解とは限らない
  - 純粋な最適化問題は、勾配が小さくなり極小点が見つかったら計算が終了。
  - 機械学習では、極小点が最善の結果とは限らない。
    - 過学習を防ぐために、早期終了や正則化を行うことが多い。
- 機械学習は経験損失を最小化する
  - 純粋な最適化問題は、あらかじめ決められた不変な目標に近づけることが目的になる。
    - 例、最短ルート探索問題は、距離を0に近づけることが目的。
  - 機械学習では、観測されたデータに近づけることが目的になる。よって、観測されたデータとの誤差(経験損失)を最小化する問題になる。

機械学習における最適化の特徴

- 機械学習では代理損失関数を設定する。
  - 機械学習では、最適化したい対象(目的関数) を直接最適化しないことがある。この場合、学習が進行しやすくなる損失関数を考え、その損失関数の最小化を図る。
  - この損失関数は、代理損失関数とも呼ばれる。
    - 例えば、最適化の対象(目的関数)を誤判定率にすると、 学習中の目的関数の値は6/10、5/10、4/10など離散的な値になり、重みの収束が不安定になる。この場合、クロスエントロピー誤差関数を損失関数に設定することで、目的関数を連続的な数値に置き換えることができ、目的関数の変化が滑らかになる。

機械学習における最適化の特徴

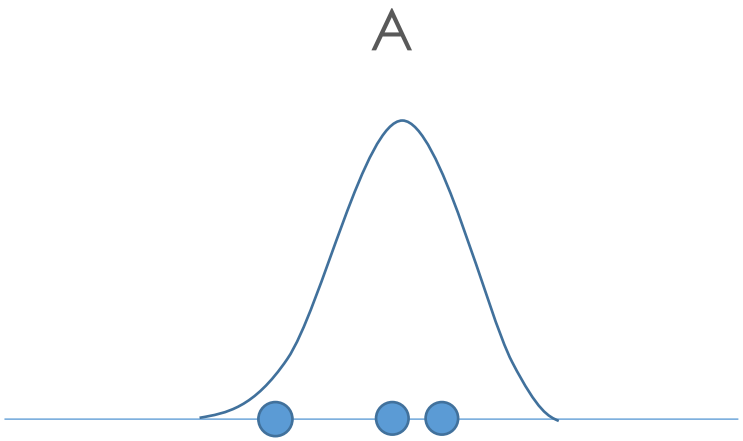
- 機械学習では目的関数が訓練事例の和に分解される。
  - 機械学習での最適化問題を最尤推定問題として考えてみると、以下のように表現できる。
    - $\boldsymbol{\theta}_{ML} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p\left(\mathrm{y}^{(\mathrm{i})} \middle| \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$
      あるθの時の, $\boldsymbol{x}^{(i)}$のもとで$y^{(i)}$が観測される確率
      - $\boldsymbol{\theta}_{ML}$：対数尤度の和が最大になる時のパラメータ
      - $\boldsymbol{\theta}$：パラメータ
      - $\boldsymbol{x}^{(i)}$：i番目のデータの説明変数, $y^{(i)}$：i番目のデータの目的変数
  - この式が意味しているのは、「訓練事例(学習データ)に対する対数尤度の和が最大になるときのパラメータ$\boldsymbol{\theta}$が学習結果として採用される」ということである。なお、ここでは過学習のことは考えていない。
  - 目的関数を訓練事例の和に分解できるという考え方は、ミニバッチ学習でもバッチ学習と同等の結果が得られていることと整合がとれる。
  - 純粋な最適化問題では、目的関数を分解できないことが多い。

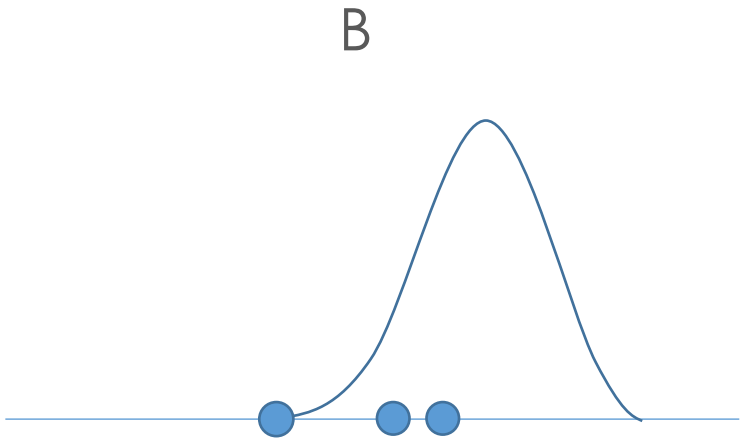
最尤法の確認

- 以下のデータが観測されたとする。
- このデータに最もよく当てはまる正規分布を求めたい。

最尤法の確認

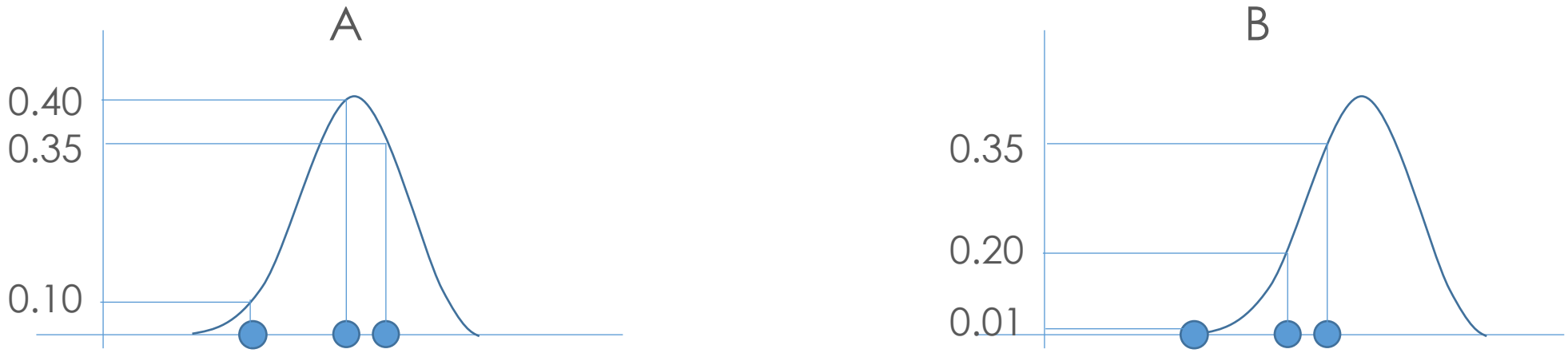
- 以下のどちらの正規分布の方がデータによく当てはまっているか？





最尤法の確認

- 正規分布A、正規分布Bのそれぞれについて、尤度を算出する。
- 尤度とは、確率(密度)を掛け算した値。



尤度=0.40×0.35×0.10=0.014 > 尤度= 0.35×0.20×0.01=0.0007

Aの正規分布の方が尤度が大きいので、Aの方がよく当てはまっていると判断する。
これが最尤法。実際には、解析的に最もよく当てはまる正規分布を求める。

ニューラルネットワーク最適化の課題

ニューラルネットワーク最適化の課題

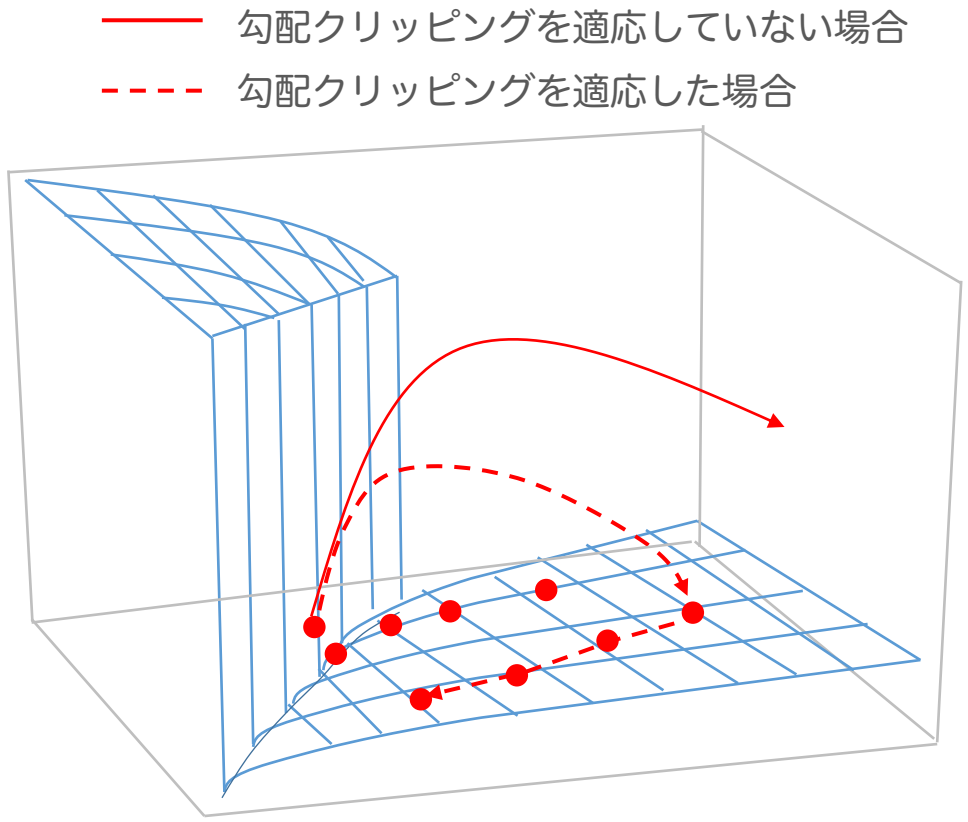
- 局所解
  - ニューラルネットワークは、一般に非凸最適化問題になる。
  - 最小点ではなく、極小点で計算が終了してしまうことがある。極小点でも最小点と同じくらいに損失が小さくなっていれば大きな問題にはならない。
  - 大きなニューラルネットワークでは、極小点の損失が最小点の損失と同程度に小さい値になるため、極小点に陥ることは大きな問題にならない、という報告もある。(Goodfellow,深層学習,p.204)
  - ミニバッチ学習を行うと、局所解に陥りにくくなる傾向がある。(DAY2で説明済み)
- 鞍点(あんてん)
  - 極小点の一種。ある方向でみると極小値、他の方向でみると極大値になっている場所のこと。最小値でないにも関わらず勾配が0になるため、探索が終了してしまう。(DAY2で説明済み)
- プラトー
  - ほとんど平らな場所。プラトーに入ると、学習が停滞する。(DAY2で説明済み)

ニューラルネットワーク最適化の課題

- 崖と勾配爆発
  - 探索領域に急峻な崖があり、そこを登ってしまうと、次のステップの勾配が非常に大きくなり(勾配爆発)、更新時の移動距離が大きくなる（遠くに飛ばされてしまう）ことがある。
  - RNNで発生することが多い問題。
  - この問題を回避する方法として、勾配クリッピングという方法がある。
- 長期依存性
  - 計算グラフが深くなる場合、同じ操作を何度も繰り返し適用することになる。これにより、誤差逆伝播時の勾配がどんどん大きく(勾配爆発)なったり、どんどん小さく(勾配消失)なったりすることがある。
  - RNNで発生することが多い問題。

勾配クリッピング

- 崖のイメージと勾配クリッピングの効果を以下に示す。



崖のイメージ

勾配クリッピングとは、勾配の値が大きくなりすぎないように調整する手法。通常は、パラメータ更新の直前に行う。勾配クリッピングには、要素ごとにクリッピングする方法や全ての要素をクリッピングする方法などがある。全ての要素をクリッピングする方法の例を以下に示す。

$$if\ \|\boldsymbol{g}\| > v$$
$$\boldsymbol{g} = \frac{\boldsymbol{g}v}{\|\boldsymbol{g}\|}$$

$\boldsymbol{g}$: 勾配ベクトル(ここでは、NN上の全ての勾配がベクトル状に並んでいるとしている)
v : 閾値(一度計算して決めたりする)
http://proceedings.mlr.press/v28/pascanu13.pdf
https://arxiv.org/pdf/1211.5063.pdf

ニューラルネットワーク最適化の課題

- 不正確な勾配
  - 例えば、目的関数が非連続であると、勾配が不正確になる。この場合は、代理損失関数を設定すれば回避できる。
- 局所構造と全体構造の不十分な対応関係
  - 難しい全体構造をもつ問題の場合、良い初期点を見つけなければ、局所解に陥る可能性が高くなってしまう。
  - 良い初期点を確実に見つける手法はまだ確立されていない。

最適化戦略とメタアルゴリズム

最適化戦略とメタアルゴリズム

- バッチ正規化(batch normalization)
  - 各層のアクティベーション分布を適度な広がりを持つように調整する手法。
  - 学習を速く進行させることができる。
  - 初期値に神経質にならなくてもよくなる。
  - 過学習を抑制する。
- 座標降下法(coordinate descent)
  - 重みを1つずつ更新していく方法。
  - 重みが2つの場合、w1を更新し、w2を更新し、w1を更新、、、これを繰り返す。
- ポルヤック平均化(Polyak–Ruppert averaging)
  - 通過した点を平均化したものを最適な点とする方法。
  - どうしても谷底にたどり着かない場合に、谷底まわりを行ったり来たりしている点を平均化すると、谷底に極めて近い点になるはずであるという発想。

最適化戦略とメタアルゴリズム

- 教師あり事前学習
  - 単純なタスクを解くモデルを訓練した後で、目的とするタスクを解くという戦略。
  - 目的とするタスクが複雑な場合に効果的。
- 再学習(fine tuning)
  - 学習済みモデルの重みを初期値にして、目的とする訓練データで学習させていく方法。
  - 目的とする訓練データで学習を始めるよりも収束が早くなることが多い。
- 転移学習(transfer learning)
  - ある問題設定で学んだことを別の問題設定の汎化性能向上に役立てること。
  - 異なる問題設定間において、共通の特徴量を仮定できるとき、転移学習は有効である。
  - 例えば、囲碁が強いモデルに将棋を覚えさせるなど。

Any Questions ?

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。