深層学習(MLP)３章_後半

確率的勾配降下方

3.4汎化性能と過適合

学習の目的→ 「まだ見ぬ」サンプルに対して正しい推定を行う

そのためにも学習時に使用する訓練データと、使用しない訓練データ
それぞれに対する推定誤差を分けて、後者の誤差を最小化されているか確認する

学習に使用する訓練データに対する誤差:　**訓練誤差**
学習に使用しない訓練データに対する誤差:**テスト誤差**

訓練によりテスト誤差は訓練誤差と同じように減少するが、
途中で訓練誤差から離れたり、ひどい場合はテスト誤差が悪化する場合がある
(過適合、過剰適合、過学習と呼ばれる)

過適合を防ぐために学習の早期終了を行うテクニックがある

3.5過適合の緩和

3.5.1正規化

ネットワークの自由度が高い方が過学習が発生しやすい
　→ネットワークの自由度は表現力に直結し、自由度を下げるのは解決にならない

学習時に重みの自由度を制約する正規化による解決方法が提案されている

例えば？
　重みに制約をつけたり
　学習対象のユニットを確率的に選定したり

3.5過適合の緩和

3.5.2重みの制約

よく使われるのが誤差関数に重みの二乗和(二乗ノルム)を使用する方法

$$E_t(w) = \frac{1}{N_t}\sum_{n\in D_t} E_n(w) + \frac{\lambda}{2}||w||^2$$

λにはだいたい0.01〜0.0001の数値が使われる

こうしておくことで、パラメータの更新が自分自身の大きさに比例した速さ常に減衰するように修正される(重み減衰) 重み減衰はバイアスに対しては適用しない

$$w^{(t+1)} = w^t - \epsilon\nabla E_t(w) = E_t(w) - \epsilon\left(\frac{1}{N_t}\sum\nabla E_n + \lambda w^{(t)}\right)$$

3.5過適合の緩和

3.5.2重みの制約

重みの上限を決める制約もある

→各ユニットの入力側結合の重みの二乗和の上限を制約する方法

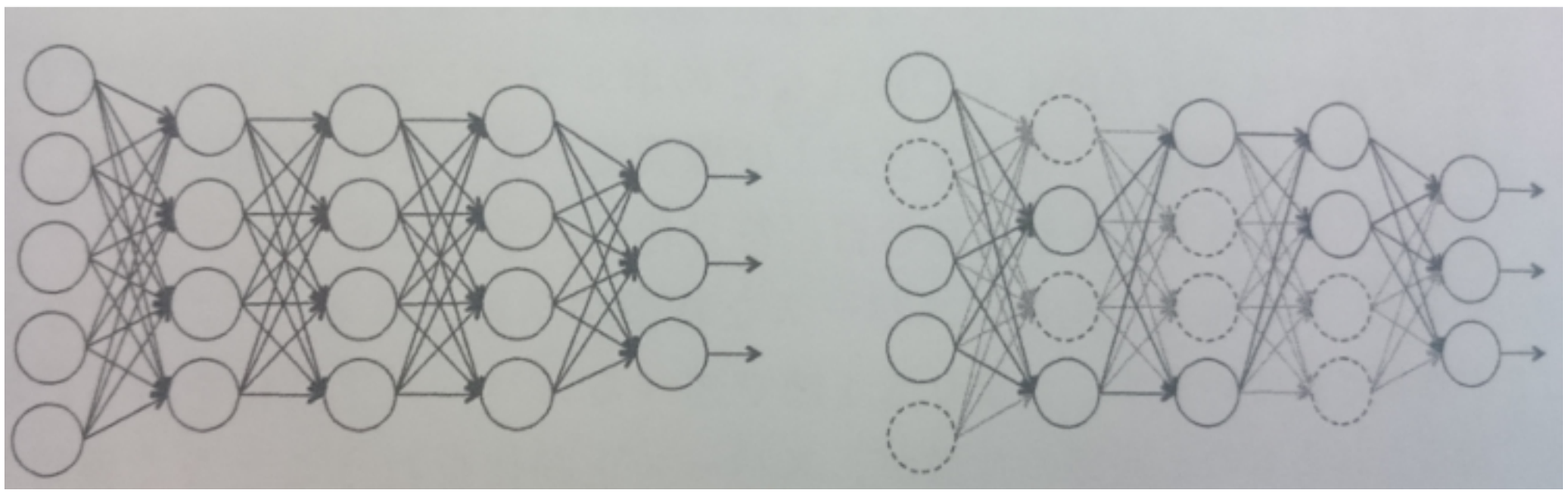
$$\sum_i w_{ji}^2 < c$$

ドロップアウトとともに用いることにより高い効果を発揮するらしい

3.5過適合の緩和

3.5.3ドロップアウト

ドロップアウトは多層ネットワークのユニットを確率的に選定して学習する方法
ドロップアウトが高性能な理由は複数の学習器を組み合わせた方法である
「アンサンブル学習」と類似しているためと言われている



3.5過適合の緩和

3.5.3ドロップアウト

学習時は中間層の各層と入力層のユニットを決まった割合pでランダムに選出する
選出したユニットのみを使用し学習させる。
ミニバッチを利用している場合はミニバッチ単位でユニットを選び直す
ユニットの選出確率pは相互とで異なる(入力層は0.9で中間層0.5とか)

学習後の推論には全てのユニットを使用する
学習時に向こうの対象となったユニットは出力をp倍する

ドロップアウトの狙いは学習時にネットワークの自由度を強制的に小さくし
過適合を避けること

3.5過適合の緩和

3.5.3ドロップアウト

多数のネットワークを独立に訓練し、推論時にはそれらの結果を平均するのと同じ効果があることからもアンサンブル学習に類似していると言える

ドロップアウトの他にもネットワークの一部を学習時にランダムに無効化する類似手法としてドロップコネクトや確率的最大プーリングがあるが、
実装の簡単さと適用範囲の広さの点でドロップアウトが概ね優れている

3.6学習のトリック

3. 6.1 データの正規化

訓練データが偏りを含む場合は学習の妨げになることがある

機械学習全般の手法として事前処理により偏りがなくなるように訓練データを変換する
訓練データに処理を加える場合、テストデータ及び推論時の各データにも前処理を施す必要がある

- **データの正規化あるいはデータの標準化**

各サンプルに線形変換を施し、成分具との平均、分散を揃える
まず各成分の平均を求めた上で、各成分を平均で割ることにより平均を0に揃える

xni ← xni - xi

それから標準偏差で各成分を割るこちにより分散が1になるようにする

3.6学習のトリック

3. 6.2 データ拡張

大量のサンプルデータを集めるのには通常かなりのコストを要し、時には不可能な場合も少なくない

その場合に手持ちのサンプルに何らかの加工を施し量を水増しするのがデータ拡張

データ拡張はサンプルのばらつきが予想できる場合に特に有効

　画像なら並行移動したり、左右に鏡像反転したり、若干の回転変動など

3.6学習のトリック

3. 6.3 複数ネットの平均

複数の異なるニューラルネットを組み合わせると一般に推定制度を向上できる
モデル平均と呼ばれる機械学習全般において有効な考え方
　→同じ入力を複数のニューラルネットに与え、複数の出力の平均を結果とする

入力に複数の返還を施しておいてそれぞれ異なるネットワークに入力して結果を平均する方法もある

学習及びテスト時の計算量が増加してしまうのが欠点

3.6学習のトリック

3. 6.4 学習係数の決め方

パラメータ更新量を決める学習係数は学習の成否を左右し特に重要。

今でも試行錯誤により手動で値を選ぶことが一般的

学習係数を手動で決める定番

学習の初期ほど大きな値を選び、進捗とともに小さくする

ネットワークの相互とで異なる学習係数を用いる

→ロジスティック関数のように値域が制約された活性化関数を用いる場合特に有効

学習係数を自動的に決める方法としてAdaGradがある

i成分の誤差関数の成分をgt,iとした時、パラメータの更新量を以下で求める

$$-\frac{\epsilon}{\sqrt{\sum_{t'=1}^{t} g_{t',i}^2}} g_{t,i}$$

*実際にはゼロ徐算しないように修正したものが使われる

3.6学習のトリック

3. 6.5 モメンタム

勾配降下の収束性能を向上させる方法の1つにモメンタムがある

具体的には、重みの修正量に前回の重みの修正量のいくばくかを加算する

$$w^{(t+1)} = w^{(t)} - \epsilon \nabla E_t + \mu \Delta w^{(t-1)}$$

μには0.5〜0.9の数値が使われる

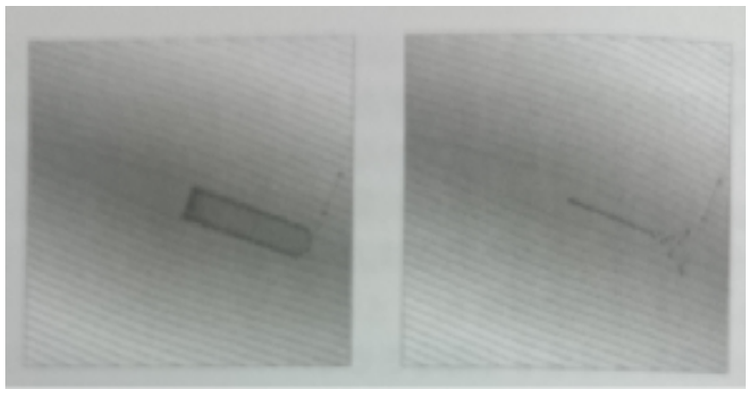
誤差関数が深い谷状の形状を持ち、かつ谷底に高低差がない場合にモメンタムを用いることで効率よく探索が行えるようになる

3.6学習のトリック

3. 6.5 モメンタム

モメンタムでは重みの修正量が過去の修正量の重み付き平均と見做すことができるので、
谷底を谷の方向に沿ってたどる修正が可能になる

$$\Delta w^{(t)} = \mu \Delta w^{(t-1)} - \epsilon \nabla E_t$$



左側はモメンタムなしの場合で、谷と直行するジグザグの動きをしている
右側はモメンタムありで谷底を一方こうにたどるため学習効率が高い

3.6学習のトリック

3. 6.6 重みの初期化

学習開始時に初期の重みを決める必要がある(バイアスの初期値は通常0)

一般的には**ガウス分布**から生成したランダム値を使用する

$$N(0, \sigma^2)$$

ガウス分布を使用する場合標準偏差σをどう選ぶかが重要になる

- σが大きいと初期の学習は早く進むが誤差関数の現象が早く停止してしまう傾向がある
- ロジスティック関数など値域に上下限のある場合はσも制約を受ける

3.6学習のトリック

3. 6.6 重みの初期化

重みの初期値に使うガウス分布の標準偏差σをどう決めるか？

小さいと0で初期化するのと変わらない

大きいと誤差関数の現象が早く停止する

(ユニットへの総入力の分散が大きいと誤差関数のパラメータによる微分値が0に近い値になる)

上記を踏まえユニットへの総入力が丁度良い分散を持つようにσを決めれば良いと言える

ユニットへの総入力: $u_j = \sum_i w_{ji} x_i$

ユニットへの総入力の分散: $V(u_j) = \sigma_{i=1}^{M} w_{ji}^2 V(x_i)$

*Mはユニットへの入力側結合の数

3.6学習のトリック

3. 6.6 重みの初期化

重みの初期値に使うガウス分布の標準偏差σをどう決めるか？

入力xiの変動を正規化(V(xi) = 1)して、総入力の分散V(uj)=σu^2にしたい場合、以下のようにσを決める

$$\sigma = \frac{\sigma_u}{M^{\frac{1}{2}}}$$

このほかには、事前学習の初期値を決める方法としてディープネットが用いられる

3.6学習のトリック

3. 6.7 サンプルの順序

サンプルデータをどの順序で取り出して学習に使うかが重要

一般的には「見慣れない」サンプルを用いると学習効率が高いと言われている
特にクラスごとのサンプル数に偏りのある分類問題では良い結果に繋がるらしい
ただし、訓練サンプルに誤った目標出力を持つものが含まれている場合は、むしろ逆効果になる

ディープネットを対象とする最近のケースでは、クラスが偏らないようにシャッフルしたサンプルを機械的に組み合わせてミニバッチを作ったりする