PythonとKerasによるディープラーニング

第二章　予習：ニューラルネットワークの数学的要素

2.4　ニューラルネットワークのエンジン：勾配ベースの最適化

19ss317u趙一

Output＝relu(dot(w,input)+b)

w,bは重みまたは訓練可能パラメータと呼ばれる。

w,bは初期値として小さな乱数値が設定される(ランダム初期化)。

次にフィードバックに基づいて、これらの重みを少しずつ調整する(学習)。

訓練ループ

- 1.訓練データ x と対応する目的値yをバッチデータとして抽出する。

- 2.ネットワークを x で実行し、予測値y_predを取得する。この手順はフォワードパスと呼ばれる。

- 3.このバッチでの損失値を計算する。損失値は、予測値y_predと目的値yとの不一致の目安となる指標である。

- 4.このバッチでの損失値が少し小さくなるようにネットワークのすべての重みを更新する。

手順4

- 手順4では、ネットワークの重みを更新する。

効率が悪い方法：調整している1つのスカラーの係数を除いて、ネットワークの重みを全て凍結してしまい、その係数でのみさまざまな値を試してみる。例えば、係数の初期値が0.3であるとしましょう。データバッチでのフォワードパスの後、そのデータバッチでのネットワークの損失値は0.5になります。

係数=0.30 → 損失値=0.5

係数=0.35 → 損失値=0.6

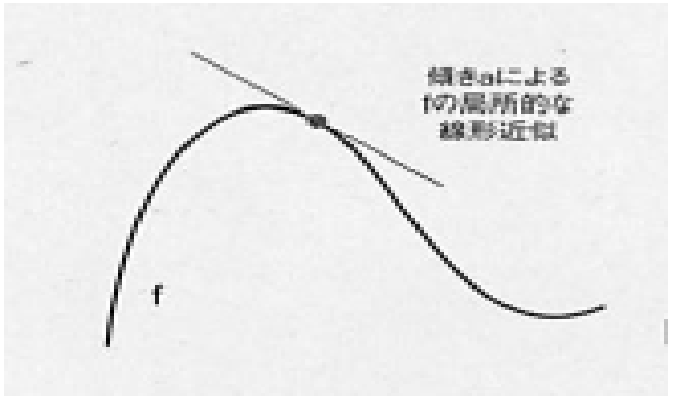
係数=0.25 → 損失値=0.4

この係数値を0.05減らしたことが、損失値の最小化に貢献したようです。ネットワークの全ての係数で繰り返すことになります。

手順４

- 効率がいい方法：「ネットワークで使用される演算がすべて微分可能であること」を利用して、ネットワークの係数に関する損失値の勾配を計算することである。あとは、それらの係数を勾配とは逆方向に移動させれば、損失値を減らすことができる。

2.4.1　導関数



図：ｆのｐにおける導関数

f(x+epsilon_x)=y+epsilon_y

epsilon_x,epsilon_y:小さな値

ｘが点ｐの近傍に収まるほどepsilon_xが小さければ、fを傾きaの一次関数として近似することが可能である。したがって、epsilon_yはa*epsilon_xになる。

f(x+epsilon_x)=y+a*epsilon_ x

傾きaは、ｆのｐにおける導関数呼ばれる。

微分可能は、「導関数が得られる」ことを意味する。

2.4.2　テンソル演算の導関数：勾配

- 勾配：テンソル演算の導関数であり、導関数の概念を多次元入力の関数として一般化したものである。
- y_pred=dot(w,x)

loss_value=loss(y_pred,y)

入力ベクトルｘ、行列ｗ、目的値ｙ、損失関数lossがあるとすれば、ｗを使って目的値の候補y_predを計算し、y_predと目的値ｙの間の損失値を求めることができる。

loss_value＝f(ｗ)

ｘ、ｙが凍結されている場合は、「ｗの値を損失値に写像する関数」として解釈できる。

2.4.2　テンソル演算の導関数：勾配

- ｆの点ｗ0での導関数は、ｗと同じ形状を持つテンソルgradient(ｆ)(ｗ0)である。つまり、係数gradient(ｆ)(ｗ0)[i,j]はそれぞれ、ｗ0[i,j]を変更したときに観測されるloss_valueの変化の向きと大きさを表す。テンソルgradient (ｆ)(ｗ0)は、関数ｆ(ｘ)=loss_valueのｗ0での勾配である。

- このため、勾配とは逆方向にｗを移動させることで、ｆ(ｗ)の値を減らすことができる。例えば、stepが小さなスケーリング係数であるとすれば、ｗ1＝ｗ0-step*gradient (ｆ)(ｗ0)のようになる。スケーリング係数stepが必要なのは、gradient (ｆ)(ｗ0)が滑らかさを近似するのはｗ0の近傍だけであるため、ｗ0から離れすぎないようにするためである。

2.4.3　確率の勾配降下法

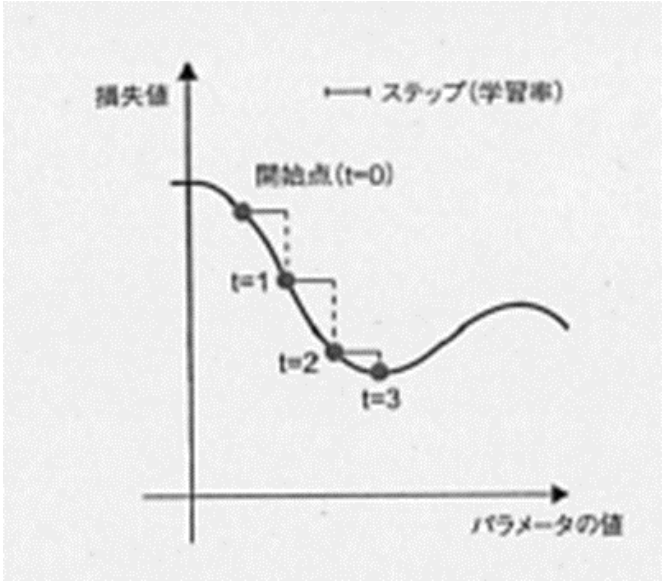
- 関数の最小値は導関数が 0 に向かう点をすべて洗い出し、それらのうち関数の値が最小になる点を調べれば良い。

- ニューラルネットワークに置き換えて考えてみると、gradient(f)(w)=0を解くことによって可能だが、パラメータの数が数千を下回ることはないので、実際には手に負えない。これに代わる方法として、ミニバッチ確率的勾配降下法を使うことができる。

ミニバッチ確率的勾配降下法

- 1.訓練データ x と対応する目的値 y をバッチデータとして抽出する。
- 2.ネットワークを x で実行し、予測値y_predを取得する。
- 3.このバッチでの損失値を計算する。損失値は予測値y_predと目的値 y との不一致の目安となる指標である。
- 4.ネットワークのパラメータを調整するために損失関数の勾配を計算する。この手順はバックワードパスと呼ばれる。
- 5.例えば、w-=step*gradientを実行するなど、それらのパラメータを勾配とは逆方向に少し移動させることで、このバッチでの損失値を少し小さくする。

ミニバッチ確率的勾配降下法



- この図は1次元の確率的勾配降下法を表している。
- Step係数の値を適切に選択することが重要である。例えば、値が小さすぎる場合、極小値で止まってしまう可能性がある。大きすぎる場合は、更新によって曲線上の完全にランダムな位置へ移動してしまうかもしれない。

確率的勾配降下法とバッチ確率的勾配降下法

- 確率的勾配降下法：ミニバッチ確率的勾配降下法の1つで、データバッチを抽出するのではなく、イテレーションごとにサンプルと目的値の1つだけ抽出する方法である。

- バッチ確率的勾配降下法：利用可能なすべてのデータですべての手順を実行する方法である。

モーメンタム

- 確率的勾配降下法(SGD)には、次の重みの更新を計算するときに、以前の重みの更新を考慮に入れるものがある(モーメンタムSGD、AdaGrad,RMSPro)。これらの手法の多くでモーメンタムという概念が使用される。

モーメンタム

- モーメンタム：収束の速度と極小値という確率的勾配降下法の2つの問題に対処する。
- 問題：パラメータが学習率の低い確率的勾配降下法を通じて最適化される場合、最適化プロセスは大域的最小値へ向かうではなく、極小値で止まってしまう。

- こうした問題はモーメンタムを使用することによって回避できます。

2.4.4導関数の連鎖：バックプロバゲーションアルゴリズム

- 導関数は明示的に計算することができる。例えば、３つのテンソル演算a,b,cと、重み行列w1,w2,w3からなるネットワーク f は、次のように定義される。

$$f(w1,w2,w3)=a(w1,b(w2,c(w3)))$$

　　微積分学により、そうした関数の連鎖は連鎖率と呼ばれる恒等式を使って微分できることが分かっている。

$$f(g(x))'=f'(g(x))*g'(x)$$

2.4.4導関数の連鎖：バックプロパゲーションアルゴリズム

ニューラルネットワークの勾配値の計算に連鎖率を適用するとバックプロパゲーション(誤差逆伝播法、リバースモード微分)と呼ばれるアルゴリズムが得られる。バックプロパゲーションでは、最終的な損失値を出発点として、出力側の層から入力側の層に向かって逆方向に進む。そして、連鎖率を適用することで、各パラメータがその損失値にどのような影響を与えたのかを計算する。