このメモでは、Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)法について解説する。

# Taylor展開

以降の説明で何度も出てくるので、Taylor展開について解説しておく。

## 1次のTaylor展開

まず、1次については、以下のようになる。これについては、何も補足は要らないよね。

(1)

**例題**の時、、でTaylor展開し、の近似値を計算しなさい。

**解答**  
まず、、。次に、をTaylor展開して、  
ちなみに、実際の解は、で、Taylor展開による解が実は厳密解であった。☺

## 2次のTaylor展開

2次元になっても、基本的には同じだが、慣れるまで難しいよね。

(2)

ちょっと複雑だけど、なぜこうなるのか証明してみれば、自ずと複雑な形の理由が見えてくるので、ちょっと簡単に証明してみよう。

まず、について、

(3)

とおくと、関数は、の関数として、と表されるので、1次のTaylor展開をして、

(4)

※式(1)と表現の仕方が違うけど、同じ意味だよ！

ここで、は、の時のの微分だ。つまり、式(3)を見ると、

同様に、

よって、式(4)は、

とおくと（ココ、ポイントだよ！）、

ここで、式(3)を思い出せば、

同様に、

ってことだ。つまり、

(5)

さて、以下の変形を覚えているかな？大学の数学で習ったよね？

(6)

また、式(3)より、

これを式(6)に代入すると、

(7)

式(7)をさらに微分すると、

(8)

以上から、式(5)に、式(7)、(8)を代入して、

証明終わり。■

さて、式(2)は、以下のようにカッコ良く表すことができる。しかも、この形にしておけば、N次元への拡張も容易だ。

(9)

ただし、はヘッセ行列。出ました！

※つまり、ヘッセ行列は、Taylor展開において、関数の2階微分の項に表れる。

**例題**の時、、でTaylor展開し、の近似値を計算しなさい。

**解答**  
まず、、。次に、をTaylor展開して、  
ちなみに、実際の解は、で、Taylor展開による解が実は厳密解であった。☺

# Newton Method

## 1次のNewton Method

ニュートン法は、の解を反復法により解くアルゴリズムの1つだ。各ステップにおいて、以下の式に基づいてを更新する。

(1)

この覚え方はすごく簡単だ。関数をTaylor展開すると、

今現在のをに更新することで、にしたいわけだから、

よって、

**例題**の時、からスタートして、の解を求めなさい。

**解答**  
まず、。以下のように、から更新していく。

ステップ1:

ステップ2:

ステップ3:

ステップ4: ちなみに、実際の解は、。上記のステップを続ければ、徐々に厳密解に近づいていくことが予想できるね。

## 2次のNewton Method

2次のNewton Methodが扱うのは、変数が2つであると同時に、関数も2つのケースだ。なんで関数が1個じゃないの？って思うよね。俺も最初は混乱した。でも、変数が2個で関数が1個だと、underdeterminedだから解けないよね。だって、解が無限個あるから。以下の例を見ると、もっと分かりやすいかな。

2次のNewton Methodでは、を以下のように更新する。

(2)

ただし、はヤコビ行列で、以下のように定義される。

(3)

これも、Taylor展開を使うと簡単に導出できるよ。まず、

(4)

1次の時と同様で、今現在のをに更新することで、にしたいわけだから、

2つの式は、以下のようにまとめて表現できる。

よって、

(5)

導出終わり。■

**例題**以下の連立方程式を、からスタートして、ニュートン法を使って求めなさい。

**解答**  
まず、関数は、以下のように定義できる。

関数のヤコビ行列を計算する。

そして、逆行列は、

幸い、このケースでは、ヤコビ行列はstaticなので、逆行列もstatic。（通常は、毎ステップで計算する必要があるので、めちゃくちゃ大変なんだよね）

から更新していく。

ステップ1:

ステップ2:

ちなみに、実際の解は、。運よく、1ステップで厳密解を得ることができた！☺

# Newton Methodで最適化

ニュートン法は、関数を求めるアルゴリズムだが、これを使って関数を求めることで、極値を求めることもできる。

※極値を求めるアルゴリズムと言えば、Gradient Descent（勾配法）や共役勾配法などがある。勾配法は、一階微分のみを使うため収束が遅い。二階微分を使うニュートン法は、収束は速いものの、ヘッセ行列の逆行列を毎ステップ計算する必要があるため、計算量が多い。これに対して、共役勾配法や準ニュートン法は収束が速く、且つ、計算量が比較的少ないので、広く使われている。

## 1次のNewton Methodで極値を求める

早速、Taylor展開してみよう。

今現在のをに更新することで、にしたいわけだから

よって、

つまり、以下の更新式に従ってを更新していくことで、の極値が求まる。

(1)

**例題**の時、からスタートして、の解を求めなさい。

**解答**  
まず、、。以下のように、から更新していく。

ステップ1:

ステップ2:

厳密解は、。運よく、1ステップで厳密解を得ることが出来た。☺

## 2次のNewton Methodで極値を求める

2章で、2次のNewton Methodを使ってを解く時、関数も2つあると言った。でも、極値を求める問題の場合は、関数は1つ。これがまた混乱しちゃうよね。

なぜかと言うと、2次元変数の関数の微分は、2次元ベクトルだ。つまり、1つの関数について、という式は2つ出来る。なので、変数が2個、式が2個で、解を求めることができるわけだ。

もう一度繰り返すね。Newton Methodで極値を求める場合は、関数は1個！

それでは、各ステップでの更新式を先に出しちゃうよ。

(2)

ただし、は関数のヘッセ行列で、以下のように定義される。

なぜこうなるか。Taylor展開をして導出してみよう。

これは、以下のようにカッコ良くまとめることが出来る。

(3)

例によって、をに更新することで、にしたいわけだから、

(4)

よって、

導出終わり。■

**例題**の極値を、からスタートして求めなさい。

**解答**  
まず、、。

また、ヘッセ行列の逆行列は、。この問題に限って言うと、運よくヘッセ行列がstaticなので、各ステップ毎にヘッセ行列の逆行列を計算する必要がない。

以下のように、から更新していく。

ステップ1:

厳密解も。運よく、1ステップで厳密解を得ることができた。☺

# Quasi Newton Method

ニュートン法は、関数を求める場合にはヤコビ行列の逆行列を、を求める場合にはヘッセ行列の逆行列を、各ステップにおいて計算する必要がある。これは非常に計算が大変だ。そこで、逆行列を求めない近似的な方法が準ニュートン法である。

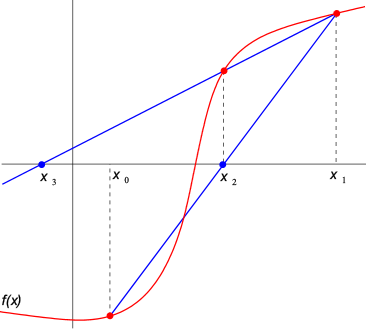
## Secant Method

アイデアは、以下のような、3章の式(3)を満たすように近似するということだ。

(1)

これは、Secant methodと呼ばれる（Numerical Analysisで出てきたけど、ほとんど覚えてないなぁ）。

Newton法と同様にの解を反復法で解くアルゴリズムだが、Newton法と違って、微分を必要としない。下図のように、範囲からスタートし、とを結ぶ直線ととの交点を計算し、範囲をに更新する。さらに、とを結ぶ直線ととの交点を計算し、範囲をに更新する。こうして、範囲が十分小さくなるまで繰り返す。



なお、交点は、以下の式により計算できる。

(2)

**例題**の解を、、からスタートしてSecant methodで求めなさい。

**解答**  
ステップ1: 、。との交点は、。。新たな範囲は、。

ステップ2: との交点は、。。新たな範囲は、。

ステップ3: との交点は、。。新たな範囲は、。

厳密解は、。徐々に厳密解に近づいていくのが予想できるね。

## BFGS Algorithm

準ニュートン法のアルゴリズムは、以下の通り。

1

2

3 、を更新する

4

とを、共に平行して更新していくところがポイントだ。つまり、逆行列の計算は行わず、それぞれを独立して更新していくので、計算が楽なのだ。

との更新方法は、いくつか方法が提案されているが、BFGS法の場合、以下のように計算する。

(3)

ちなみに、Wikipediaや多くのウェブサイトでは、としてを使っているので混乱する。を、ヘッセ行列の近似（つまり）の逆行列として使っているのだ。なので、上の式(3)も以下のように表現されている。

(4)

## L-BFGS Algorithm

ようやく前置きが終わった。いよいよ、本題のL-BFGSアルゴリズムの話だ。

以下のウェブサイトにも説明があるように、基本的なアイデアは、やを計算しない代わりに、との履歴を過去個分（ぐらい）保持し、それを使って直接を計算する。が次元だとすると、通常のBFGSだと、のためにのメモリを使用するのに比べ、L-BFGSだとで済むわけだ。従って、非常に高次元のデータ（）を扱う場合、L-BFGSを使用することで大幅にメモリの節約ができる。

※例えばAutoencoderの場合、だけでも次元なので、かなりメモリを使うことが想像できるね。

とりあえず、今はこれ以上細かいことは気にせず、ライブラリを使っちゃおう！

<http://www.alglib.net/optimization/lbfgsandcg.php>