## ロジスティック回帰モデル(2値分類)

|  |
| --- |
|  |

1. 決定境界(予測関数)を定義()
   1. の形式に回帰させる
   2. w0の係数1を明示し、ベクトル内積の形にする
      1. , ()
      2. 学習フェーズ時には標本データの数を考慮するので、
   3. 決定境界が右上がりではない場合、
2. 予測式の作成
   1. 予測式は決定境界とシグモイド関数の合成関数となる(0,1で分類するため)
      * 1. 学習フェーズ時には標本データの数を考慮するので、
3. 確率値の定義
4. 尤度関数(Lk)の定義
   1. 対数を取り、計算しやすくする。
      1. ytの値によって、P(yt, yp)は異なることを考慮した式変形を行う
5. 最尤問題の定義
   1. 最尤問題は値を最大化することが目的だが、損失関数は値を最小化することを目的とするため、符号を反転する
      * 1. 上式は情報論理のエントロピーの式に似ているので、交差エントロピーと呼ばれる
   2. 最尤推定を行う
      1. 損失関数L()を各重みで偏微分する
         1. 合成関数の微分を利用する
            1. L(yp)は、決定境界u(wi)を入力とする予測関数yp(u)を入力とする合成関数だと考えれる。よって、

L()は交差エントロピーce()、yp()はシグモイド関数f()である

* + - * 1. 誤差 を定義する

学習フェーズには、標本データの数を考慮するので、

* 1. 勾配降下法の適用
     1. 偏微分式
        + 1. k:(勾配降下)繰返し回数、m:標本データ数、i:次元数
          2. ベクトル化し、全微分化する

1. 交差エントロピーの微分について
   1. 特定の交差エントロピー関数を微分して、後でΣ計算
      * 1. 特定の交差エントロピー式なので、index(m)は不要
   2. 学習フェーズなので、ytは定数。変数はypである
      * 1. (1-yt)(-1)/(1-yp)の導出((1-y)log(1-yp)の微分計算)は、u=1-xを使った合成関数の微分を利用する

## ロジスティック回帰モデル(多値分類)

|  |
| --- |
|  |

1. 考え方
   1. 1つの分類器に複数の値を予測させるのではなく、0-1の確率値を出力する複数の分類器を並列に作成し、確率値の最も高い分類器に対応するクラスをモデル全体の予測値とする
2. 正解値(yt)の定義
   1. OneHotベクトル
   2. 1対他分類器(One vs Rest Classifier)( 1要素が1になる場合、その他の要素がゼロであることが期待されるベクトル)
3. 決定境界(予測関数)を定義
   1. ytのOneHotベクトル化に合わせて、重みを行列化し、分類器を次元数作成する
      1. ダミー変数を用意し、ベクトル内積とする
         1. , ()
4. 予測式(yp)の作成
   1. 最終的にsoftmax関数に入力し、0-1のベクトル形式に変換する
5. 確率値の定義
   1. 2値分類の際、sigmoid()自体は分類器ではなく、正解値に応じて、class1=yp, class2=1-ypで定義するが、多値分類の場合はsoftmax()が分類器(yp=(yp0,yp1,yp2))のため、のそのような定義は不要
6. 尤度関数(損失関数)(Lk)の定義
   1. 2値分類と比べて、ytの値によって、P(yt, yp)は異なることを考慮する必要がないが、OneHotベクトルの各要素間の確率値の対数を合計する必要がある。
      1. (nはベクトル要素数、Mは標本データ数)  
         ※ 上式も交差エントロピーと呼ばれる
7. 損失関数の微分計算
   1. 1つの標本データに関しての交差エントロピー式で考える
   2. 損失値Lは、**W**からなる**u**(を入力とするg())からなる**yp**(を入力とするce())からなる値(合成関数)である。これをwijで偏微分する
      * 1. 水色部分は、中間変数がベクトルの場合の微分公式である
      1. Lすなわちce(), ypすなわちg()なので．．．
         * 1. g()の微分公式は、要素と偏微分対象の次元の関係で、出力が異なるが、Lの偏微分は最終的に以下にまとまる
           2. 次元の数を考慮し、誤差を定義すると
      2. 最終的な偏微分式は
   3. 標本データ数を考慮すると
8. 勾配降下法適用
   * 1. k:(勾配降下)繰返し回数、m:標本データ数、i:次元数

## ディープラーニングモデル

|  |
| --- |
|  |

* 概念の理解
  + 中間値ベクトルの定義
    - 前層ノードと重み行列の積を計算した直後のベクトルのこと
  + 活性化関数の定義
    - 中間値ベクトルに関数として作用し、各層の最終的な値(結果ノード)を得るためのもの。
  + 結果ノード
    - 最終的な値を持つノード。

|  |  |  |
| --- | --- | --- |
|  | 隠れ層 | 出力層 |
| 重みベクトル(行列) | ***V*** | ***W*** |
| 中間値ベクトル | ***a*** | ***u*** |
| 活性化関数 | sigmoid() | softmax() |
| 結果ノード | ***b*** | ***yp*** |

* 隠れ層
  + 中間値ベクトルの定義
  + 結果ノードの定義
* 出力層
  + 中間値ベクトルの定義
  + 結果ノードの定義
* 損失関数の定義
  + 出力層に関して、ロジスティック回帰(多値分類)のそれと全く同じモデルの為、損失関数をそのまま利用できる
* 損失関数の微分計算
  + 1つの標本データに関しての交差エントロピー式で考える
  + 出力層に関して、ロジスティック回帰(多値分類)のそれと全く同じモデルの為、損失関数の微分計算をそのまま利用できる
    - * (出力層への入力は**x**ではなく**b**である)
    - 入力層－隠れ層の重みVを偏微分する
      * aiはvijが関連する(入力とする)要素(関数)である。これで合成関数の微分を利用する
        + (緑式は、aiでvjiの係数がxjのみである)

biはaiが関連する(入力とする)要素(関数)である。これで合成関数の微分を利用する

biすなわちf(ai)なので

**u**はbiが関連する(入力とする)ベクトル(関数)である。これで偏微分を含む合成関数の微分を利用する

(緑式は、ulでbiの係数がwliのみである)

よって、

lは出力クラス(分類結果)数

* + - * 最終的に、
* 誤差逆伝播
  + WとVでの損失関数の微分結果を比較する
    - V…
    - W…
    - Vの黄色部分はWにおける予測値誤差に相当している。ディープラーニングモデルでは、これを隠れ層における誤差、としてbdで定義する
* 隠れ層2層の場合
  + 前式を利用して
    - (前式)
    - (今式)
    - uijは入力ではなく、重みなので注意
    - **c**はbiが関連する(入力とする)ベクトル(関数)である。これで偏微分を含む合成関数の微分を利用する
      * + (緑式は、clでbiの係数がcliのみである)
        + Hは**c**の要素数
        + δL/δclは、隠れ層における誤差分に相当する(図を参照)
    - 最終的には
  + 上式によって、以下の事が分かる
    - δL/δuijを計算するには、隠れ層1の誤差bdi=δL/δaiが分かればよい
    - bdi=δL/δaiは、隠れ層2の誤差ddlと重み行列vliの値から計算できる
    - 誤差の計算
      1. 出力層誤差**yd**(**yp**と**yt**から)
      2. 隠れ層2の誤差ベクトル**dd**(**yd**と重み行列**W**から)
      3. 隠れ層1の誤差ベクトル**bd**(**dd**と重み行列**V**から)
    - 偏微分の計算
      1. **W**の勾配計算(**yd**と**d**から)
      2. **V**の勾配計算(**dd**と**b**から)
      3. **U**の勾配計算(**bd**と**x**から)
* 勾配降下法
  + 1層の場合
    - (予測値誤差)
    - (予測値誤差から隠れ層誤差を計算)
  + 2層の場合
    - (予測値誤差)
    - (予測値誤差から隠れ層2誤差を計算)
    - (予測値誤差から隠れ層1誤差を計算)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ディープラーニングモデル | | |
| 予測値計算 |  | 入力層ノードと第1層重みの内積 |
|  | 隠れ層ノードの値 |
|  | 隠れ層ノードと第2層重みの内積 |
|  | 予測式 |
| 誤差計算 |  | 予測値誤差 |
|  | 隠れ層誤差 |
| 勾配計算 |  | 第2層の重み勾配計算 |
|  | 第1層の重み勾配計算 |
| ディープラーニングモデル(隠し層2層) | | |
| 予測値計算 |  | 入力層ノードと第1層重みの内積 |
|  | 隠れ層1ノードの値 |
|  | 隠れ層1ノードと第2層重みの内積 |
|  | 隠れ層2ノードの値 |
|  | 隠れ層2ノードと第3層重みの内積 |
|  | 予測式 |
| 誤差計算 |  | 予測値誤差 |
|  | 隠れ層2誤差 |
|  | 隠れ層1誤差 |
| 勾配計算 |  | 第3層の重み勾配計算 |
|  | 第2層の重み勾配計算 |
|  | 第1層の重み勾配計算 |

* softmax()の微分
  + ケース(入力x=3次元ベクトル、出力y=3次元ベクトル)
  + y1をx1で微分すると
    - exp(x1)=hと定義する
    - (右式は商の微分の公式を利用する)
      * よって
  + y2をx1で微分すると
    - exp(x1)=hと定義する、exp(x2)=iと定義する
    - (hはx2の関数のため、δh/δx1=0である)
  + 上２解より、
* 商の微分の公式
  + の微分
    - を定義する
    - よって
    - h’(x)は、 を定義して、合成関数の微分として考える
      * + 緑式は
    - よって

# 度数分布表の作成

* 階級
  + データ幅(最大値-最小値)を階級幅で割って分類する。
    - 階級幅はデータ数の平方根または、スタージェスの公式を使う  
       (スタージェスの公式)

|  |
| --- |
| # 平方根  sqrt(length(data))  # スタージェスの公式  nclass.Sturges(data) |

* + データは整列する

|  |
| --- |
| sd <- sort(data)  class1 <- subset(sd, 0 <= sd & sd < 10)  class2 <- subset(sd, 11 <= sd & sd < 20)  :  class9 <- subset(sd, 90 <= sd & sd < 99) |

* 階級値
  + 階級の中間値

|  |
| --- |
| cvalue = (max(class#) + min(class#))/2 |

* 度数

# 正規分布の確率密度関数

* exp(-x^2)…平均値を取る確率が一番大きく、平均値から離れるにつれてその値を取る確率が小さくなる事象を表せる
  + 平均値を取るxに任意の値を取れるようにする
    - μに任意の値を取ることによって、中央値のxは左右に平行移動できる
  + 幅を変える
    - σの値に依らず、常に正の値を取るようにするため、2乗値にする。
    - 係数が2だと、その後の計算結果がきれいになる
  + 密度関数は全区間の確率点の総和が1になる必要があるため、関数を定数倍して調整する
  + よって、最終的な正規分布の確率密度関数は