ML01_プロローグ.mp4

まずはじめに

問題設定が非常に重要。
必ずしも機械学習を使う必要はない
ルールベースでできるならルールベースでも良い。
ルールーベースにするとデメリットとしては、
設定した数の分だけ設定する必要になる。数が多いと辛い。

また、機械学習を行うデメリットとしてはテストを行うことが難しい
通常のテストの場合は、Aに対してBが来ることを前提としているが、
機械学習はそうとは限らない。

機械学習モデリングプロセス

機械学習だけに限らず、深層学習も同じような過程を踏んでモデルを作成してゆく。
プロセスは6つある。
・問題設定
・データの選定
・データの前処理
・機械学習モデルの選定
・モデルの学習
・モデルの評価

問題設定

どのような課題を機械学習で解決するか

データのの選定

問題を解決するためにどんなデータを使うか。

データの前処理

モデルに学習させれるようにデータを変換する。
自然界やビジネスの中で得られるデータはあまり成形された状態で取得することはなかなかない。
例えば、データの欠損値などが挙げられる。

機械学習モデルの選定

どの機械学習モデルを利用するか
特定の問題に対してどんなモデルを使うか、

モデルの学習

パラメータの設定の仕方。

モデルの評価

ハイパーパラメータの選定。
モデルの精度を測る

大体の大枠はDLも同じ

ML02_本編

機械学習とは

・コンピュータプログラムを経験によって自動的に改善していく方法について研究

・コンピュータプログラムが、タスクT(アプリケーションにさせたいこと)を性能指標Pで測定し、その性能が経験E(データ)により改善される場合、タスクTおよび性能指標Pに関して経験Eから学習す　ることと言われている。(トム・ミチェル 1997)

・学習用データセット(経験E)を利用して訓練した後に、未知のデータに対して正確に予測・分類できるアルゴリズムのことを言う

ML03_線形回帰モデル

回帰モデルとは

回帰問題を解くための機械学習モデル
ある入力(数値)から出力(連続値)を予測する問題
回帰は、直線や曲線を用いて予測される。
線形回帰：データを直線で予測
非線形回帰：データを曲線で予測

回帰で扱うデータ

入力はm次元ベクトル(m=1の場合はスカラ)
入力ベクトルの各要素は説明変数(または特徴量という)
出力はスカラー値(目的変数と呼ぶ)

■説明変数
$x = (x_1, x_2, \cdots, x_m)^T \in R^m$
■目的変数
$y \in R^1$

線形回帰モデル

教師あり学習(正解付きデータから学習)
入力m次元パラメータの線型結合を出力するモデル
M次元のパラメーターとは、説明変数と同じ話

線型結合とは
入力と説明変数orパラメータ(いずれも特徴量)の内積をとり、切片を加えたもの
慣例として**予測値にハット**を付ける (正解データと異なる)

■パラメータ
パラメータ(特徴量)が予測値に対してどくらい影響を与えるかを決定する重みの集合
$\omega = (\omega_1, \omega_2, \cdots, \omega_m)^T \in R^m$
■線型結合
$\hat{y} = \omega^T x + \omega_0 = \Sigma_{j=1}^{m} \omega_j x_j + \omega_0$

線形回帰の行列表現(連立方程式を行列で表現)

説明変数(特徴量)の集合{ $x_1, \cdots, x_n$}と，それらに対応する目的変数の集合{y_1, $\cdots$,y_n} が与えられているとき，それらに対応する誤差の集合を {$\varepsilon_1, \cdots, \varepsilon_n$} とすると，線形回帰モデルは以下のように表される

$$y = X\omega + \varepsilon$$

$y = (y_1, \cdots, y_n)^T$
$X = (x_1, \cdots, x_n)^T$
$x = (1, x_{i1}, \cdots, x_{im})^T$
$\varepsilon = (\varepsilon_1, \cdots, \varepsilon_n)^T$
$\omega = (\omega_1, \omega_2, \cdots, \omega_m)^T$

パラメータの最適化問題は、最小二乗法により決まる。
パラメータの最適化問題とは、各パラメータの微分値が最小値になる値を探すこと。

ML04_データ分割/学習

機械学習をする際に手持ちのデータを全て学習データとして使うことはなく、
学習用データと検証用データに分けなければならない。

なぜ分割するか

機械学習が本来したいことは、学習したデータに対する評価ではなく、未知のデータに対しても精度よく予測するモデルを作成することである。
つまり、モデルの**汎化性能**が高いモデルを作成することである。
当たり前のことであるが、学習したモデルに学習用データを使って評価をすると精度が高くなる。
上記のことを考慮すると、学習用データと検証用データに分ける必要があるよね。

データの分割

学習用データ : 機械学習モデルの学習に利用するデータ
検証用データ : 学習済みモデルの精度を検証するためのデータ

線形回帰における学習の評価指標について

線形回帰の評価指標は、**平均二乗誤差 (残差平方和)**を用いる。
平均二乗誤差とは、データとモデルの出力の二乗誤差の和で表せすものである。
言い換えると、平均二乗誤差の結果が小さいほど直線とデータの距離が近い(似てる)と言える。
式は、下記で表すことができる。

$$MSE_{train} = \frac{1}{n_{train}} \Sigma_{i=1}^{n_{train}} (\hat{y}_i^{train} - y_i^{train})^2$$

検証誤差に関しては、添字のtrainがtestに変更した式になる。

最小二乗法を用いて、パラメータの最適化をする

最小二乗法とは、学習データの**平均二乗誤差**を最小とするパラメータを探索する方法である。
学習データの**平均二乗誤差の最小化**は、その勾配が0になる点を求めれば良い。
つまり、平均二乗誤差の微分した結果(勾配)がゼロになるようなωを探すこと。
例としてデータがプロットされてるとすると、その点群にいい感じに直線や曲線がフィットするようなパラメータを探すことである。今回は線形回帰なので直線。

$$\hat{\omega} = arg\ min_{\omega \in R^m+1} MSE_{train}$$
$$\Rightarrow \frac{\partial}{\partial \omega} MSE_{train} = 0$$

ML06_線形回帰

非線形回帰

これまで、線形回帰を取り扱って来た。
しかし、様々な現象というのは線形回帰で説明できないケースが実際多い。
大抵は、非線形で様々な現象が表現されている。
そのため、非線形回帰について学習する必要がある。
非線形回帰であっても、線形回帰と同様に考えることが可能。
具体的には、下記の３つを行う。

1.基底関数を選択
2.説明変数(特徴量)を選択した**基底関数**に入力(写像する)して特徴空間に飛ばす。
　基底関数の例⇨多項式関数、ガウス型基底関数、スプライン関数 / Bスプライン関数
3.基底関数の出力値と推定パラメータの線形結合を考える。

ML07_正則化法

機械学習や深層学習のモデルを学習をした際に、注意するべきことがある。
それは、過学習や未学習である。

未学習

　学習データとの誤差が十分小さくならない。
　言い換えると、学習データに対して学習がうまくできていない。

過学習

　学習データとの誤差は少ないが、検証データに対して誤差が大きくなる。
　言い換えると、学習データに対しては学習できているが未知のデータに対して対応できていない。(汎化性能が低い)

未学習や過学習に対する対策

　正則化法により、未学習や過学習を回避することができる。
　正則化項という考え方を導入し、回帰直線・曲線の複雑化を調整する。(曲線なめらかさを調節)

正則化法 (罰則化法)

モデルの複雑さに伴って、その値が大きくなる**正則化項(罰則項)を課した関数の最小化**
正則化項で利用する関数としては、例えば次のようなものがある。
L2ノルム: Ridge推定量 → パラメータ全体の総量を低く抑える。(縮小推定)
L1ノルム: Lasso推定量 → いくつかのパラメータを0に推定する。(スパース推定)

制約面と損失関数の最小問題を解くこと(微分する)になるが、イメージとしては制約面と損失関数が接する点が最小の点となることがわかっている。
具体的な計算方法は、ラグランジュの未定乗数法（KKT条件）を解くとわかるらしい。

ML08_モデル選択

制約面の度合いはどのように決めればよいか

クロスバリデーション(交差検証)を用いて決めていく。

検証の方法

検証方法は2つある。
・ホールドアウト法
訓練データを用いて学習を行い、訓練に用いていないテストデータを用いて精度検証を行う。
・クロスバリデーション法
データをk個の重複しない集合に分割し、そのうちの1つをテストデータ、残りを訓練データとして訓練し、精度計算を繰り返す。
CV値が一番大きいモデルを最終的に使用する。

ML09_ロジスティック回帰

クラス分類に使われる機械学習モデル(深層学習の活性化関数としても利用される。)
ロジスティック回帰の**入力**は $x = \in R^m$ で、m次元である。
ロジスティック回帰の**出力**は $y = \in (0, 1)$ で、0 or 1の2値である。
具体的には、0 or 1の結果を出力する前の処理で確率を出力。(シグモイド関数)
そして、その確率の結果から0 or 1を出力する。他クラス分類にはあまり使わない。

ロジスティック回帰の計算自体は、線形回帰と比較すると線型結合するところまでは同じである。**異なる部分は、線型結合後にシグモイド関数を扱うところである。**
このシグモイド関数が線型結合を確率に変換する処理である。

シグモイド関数

シグモイド関数　：$\frac{1}{1+exp^{(-ax)}}$
シグモイド関数の特徴
・入力は-∞ ~ ∞の値を取る。出力は0~1の値を取る。(確率値に変換する)
x = 0の時は出力が0.5。
・aを増大させていくと勾配が急になる。
・シグモイド関数の微分はシグモイド自身で表現することが可能。
$\frac{\partial \sigma(x)}{\partial x} = a\sigma(x)(1 - \sigma(x))$
・シグモイド関数の出力から分類結果を決める。
例) 0.5以下はクラス0, 0.5より大きい場合クラス 1 などと定義する。

~を最大化する、最小化するというのは最適化問題という。
最適化の実施には微分が必要であり、微分後の形がわかることは計算を楽にさせる。

ML10_最尤推定

最尤推定を実施するにあたり、**尤度関数**という概念が必要になる。
尤度関数とは、今まで取り扱っていた確率関数と考え方が逆のプロセスのものと考えると良い。

確率関数

確率関数は、固定されたパラメータ分布(事象が固定化されている？)から特定の確率変数がどれだけ得られやすいか表したもの。
既知の分布(確率分布)から得られやすいデータについて知るために使う。

尤度関数

尤度関数とは、確率分布を仮定し、あるデータからどの確率関数がどれほど尤もらしいかを表現したもの。
既知のデータから分布がどんな形か(確率分布)を知るために用いる。

ML11_勾配降下法

ロジスティック回帰の場合、尤度関数が最小となるパラメータを解析的に求めることができない。(手計算で算出がかなり困難)
そこで解析的に求められない場合に用いられる方法の１つとして、勾配降下法という手法がある。

勾配降下法

勾配降下法とは、反復計算（勾配降下）によって(逐次的に)パラメータを更新する手法である。
デメリットとしては、１回のパラメータ更新のために、すべての入力データが必要であることがあげられる。
入力データが膨大になると、計算時間・記憶容量が問題となる。
この点を解決するために、確率的勾配降下法というものが存在する。

確率的勾配降下法

データを一つずつランダムに選んでパラメータを更新する。
勾配降下法でパラメータを1回更新するのと同じ計算量でパラメータを n回更新できる。
確率的勾配法は、勾配降下法に比べ大域的最適解を見つけやすい。
言い換えると、勾配降下法に比べ、局所最適化にハマりにくい。

ミニバッチ勾配降下法

勾配降下法と確率的勾配法の良いところを合わせたようなもの。
全データをn個の塊に分ける。
そして、分けた塊をランダムに選んで、塊ごとに学習をさせる方法。
塊をn=1にすると確率的勾配降下法と同じ意味になる。

ML12_確率的勾配法

デモ動画を見て。。。

パラメータに初期値を与え、徐々にパラメータを更新し、収束した時点でそのときのパラメータを最適値として採用する。
学習率ηは、学習の収束のしやすさを表す。パラメータ更新の歩幅が大きくなるイメージ。
ηが小さいと収束するまでに時間がかかる。
しかし、ηを大きくしてしまうと、大域的局所解を飛び越えるのが難しくなる。
逆に小さすぎるのもよろしくない。

ML13_モデルの評価

混同行列

・予測値とデータの結果が合致
  - 予測が Positive: True Positive (TP)
  - 予測が Negative: True Negative (TN)
・予測値とデータの結果が合致しない
  - 予測が Positive: False Positive (FP)
  - 予測が Negative: False NegativFN)

正解率、適合率、再現率、F値

正解率：$\frac{TP}{TP+TN+FP+FN}$
再現率：$\frac{TP}{TP+FN}$・・・Recallとも呼ばれる
適合率：$\frac{TP}{TP+FP}$・・・Precisionとも呼ばれる
F値　　:$2\times\frac{適合率\times再現率}{適合率+再現率}$
　　F値はRecallとPrecisionの調和平均
ケースによって、再現率や適合率どちらを重視するか変わる。

RecallとPrecisionはトレードオフの関係なため、
Recallを維持したままPrecisionを上げる、Precisionを保ったままRecallを上げることは難しい。

ML17_主成分分析

主成分分析とは、
多次元データが持つ構造をより少数の指標にまとめる手法

主に下記に使用される。
・次元圧縮
・少数変数にまとめることによって可視化が可能（100次元→2,3次元）

どうやって多次元データが持つ構造を捉えるか？
学習データの分散が最大になる方向への線形変換を求めれば良い。
主成分分析では、分散の大きさを情報量の大きさとして考えている。
そのため、分散が大きい方向への一次変換を考えている。

【再喝】まとめると。。。
多次元データの情報(構造又は特徴)を、
出来るだけ損失せずに低次元で表現する手法

情報量の大きさ＝分散の大きさと考えること。
なぜ、分散が情報量の大きさを表すかは後に後述する。

数式に落とし込む

学習データ
$x=(x_1,x_2,\cdots,x_m)^T\in R^m$

平均ベクトル

$\bar{x} = \frac{1}{n}\Sigma_{i=1}^{n} x_i$

データ行列を平均ベクトルとの差分から作成

$X = (x_1 - \bar{x_1}, x_2 - \bar{x_2}, \cdots, x_m - \bar{x_m})^T$

分散共分散行列

$\Sigma = Var(\bar{X}) = \frac{1}{n}\bar{X}^T\bar{X}$

n個のデータを係数ベクトルを用いて線形変換

$s_j = (s_{1j}, s_{2j}, \cdots, s_{nj})^T = \bar{X}a_j$

線形変換後の分散が最大となる射影軸（線形変換）を探索する

線形変換後の分散は、

下記の通りでこれを**最大化する係数ベクトル**a_jを求める

$$\begin{aligned} Var(s_j) &= \frac{1}{n}s_j^T s_j \\ &= \frac{1}{n}(\bar{X}a_j)^T\bar{X}a_j \\ &= \frac{1}{n}a_j^T\bar{X}^T\bar{X}a_j \\ &= \frac{1}{n}a_j^T\Sigma a_j \\ &= \frac{1}{n}a_j^T Var(\bar{X})a_j \end{aligned}$$

以下の最適化問題を解く (**ノルムに制約を入れないと無限に解がある**)

$arg\ max_{a\in R^m} a_j^T Var(\bar{X})a_j$

subject to $a_j^T a_j = 1$ ⇐ ノルムの制限

上記の制約付き最適化問題はラグランジュ関数を最大にする係数ベクトルを見つければいい

・係数ベクトルで微分して0と置き解を求める

・分散共分散行列の固有ベクトルが解となる → 固有値問題に帰着する

ラグランジュ関数(⇨参考サイト：https://mathtrain.jp/mlm (https://mathtrain.jp/mlm))

$E(a_j) = a_j^T Var(\bar{X})a_j - \lambda(a_j^T a_j - 1)$

$\frac{\partial E(a_j)}{\partial a_j} = 2Var(\bar{X})a_j - 2\lambda a_j = 0$ ―※

$Var(\bar{X})a_j = \lambda a_j$ ⇐ **分散共分散行列の固有値問題を解く**

上記の式からわかることが下記

・射影ベクトル($Var(\bar{X})a_j$)の向きは固有ベクトル

・分散の値は固有値と一致

$Var(s_1) = a_j^T Var(\bar{X})a_j = \lambda_1 a_j^T a_j = \lambda_1$

よって、

線形変換後の分散は、分散共分散行列の固有値と一致することを証明した。

分散共分散行列は実対象行列なので，固有ベクトルはすべて直行する。

主成分

最大固有値に対応する固有ベクトルで線形変換された特徴量を第1主成分と呼ぶ

k番目の固有値に対応する固有ベクトルで変換された特徴量を第k主成分と呼ぶ

寄与率

k番目の分散の全分散に対する割合を第k主成分の寄与率という

※行列が対称行列で、横ベクトルと縦ベクトルの成分が同じであるようなものを二次形式と呼ぶ。この二次形式の微分については下記のサイトを参照した。
https://mathtrain.jp/quadraticform (https://mathtrain.jp/quadraticform)

余談ではあるが
対称行列Aがある場合、下記の性質がある
$A^T = A$

例題： $(A^T A)^T = A^T A$

ML18_K近傍法/K平均法

k近傍法

k近傍法とは、データの近傍周辺におけるデータを見て、多数決によりクラスを決定する手法
分類のために使われる手法である。

例えば：
ある点の近傍周辺にクラス1が2個、クラス2が1個、クラス3が3個となっている場合。
そのときある点のクラスは3と決定する。
あらかじめ、kは決定する必要がある。

k近傍法のkは、あるデータ近傍周辺で何個のデータを見るかという意味のk。
上記の例ではkがk=6である。
k=1の時の呼び名は、最近傍法である。

・kが小さい場合は、近傍の点に大きく左右されてしまうので、バリアンスが高いような状態となる。
・kを大きくすると境界線はなめらかになる。

k平均法(k-means)

k平均法とは、与えられたデータをk個のクラスタに分類する手法である
k平均法は教師なしで、クラスタリングをする手法として分類される。
Kは事前に決める必要がある。
クラスタリングとは、特徴が似ているもの同士をグループ化してくれる手法。

中心の初期値を変えるとクラスタリングの結果も変わりうる。
そのため、何回か初期値を変えて結果が変わらないことを確認する必要がある。

Kの値を変えるとクラスタリング結果も変わる。
Kの数を決め方は難しい。大体はナレッジ次第らしい。。。

k平均法の計算手順

(1) 各クラスタ中心の初期値を設定する
(2) 各データ点に対して、各クラスタ中心との距離を計算し、 最も距離が近いクラスタを割り当てる
(3) 各クラスタの平均ベクトル(中心)を計算する
(4) 収束するまで2, 3の処理を繰り返す

In [ ]:

In [ ]:

```
```