1 線形回帰モデル

回帰問題

直線で予測→線形回帰
曲線で予測→非線型回帰

入力(m次元のベクトル、m=1の時はスカラ)
↓
出力(スカラー値→目的変数)

線形回帰とは

- 教師あり学習
- 入力とm次元パラメータの線形結合(内積)を出力する
- 予測値はハット^をつける

$$\hat{y} = w^T x + w_0 = \sum_{k=1}^{n} w_j x_j + w_0$$
$$(\hat{y} \quad \text{は予測値})$$

- 線形結合→入力とパラメータの内積
- 切片も足すw0
- 出力はスカラ
- パラメータは最小二乗法により推定
- 説明変数が一次元→線形単回帰モデル
- 説明変数が多次元→線形重回帰モデル
- データは回帰直線(または回帰曲線)に誤差が加わり観測されていると仮定

$$y = w_0 + w_1 x_1 + \epsilon$$
$$y = \text{目的変数}$$
$$w_0 = \text{切片}$$
$$w_1 = \text{回帰係数}$$
$$x_1 = \text{説明変数}$$
$$\epsilon = \text{誤差}$$

データの分割

データを学習用と検証用を分ける
→汎化性能で測る

パラメータの推定

- 平均二乗誤差(残差平方和)
  →データとモデル出力の二乗誤差の和

$$\mathrm{MSE}_{train} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- 最小二乗法
- 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
- 平均二乗誤差の最小化は勾配が0になる点を求めれば良い

ハンズオン 線形回帰モデル(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/線形回帰モデル.ipynb)

2 非線形回帰モデル

基底展開法

- 回帰関数として、基底関数と呼ばれる既知の非線形関数とパラメータベクトルの線型結合を使用
- 最小2乗法や最尤法により推定

$$y_i = w_0 + \sum_{i=1}^{m} w_j \phi_j(x_i) + \epsilon_i$$

よく使われる基底関数

- 多項式関数
- ガウス基底関数
- スプライン関数/Bスプライン関数
- 多項式(1~9次)

$$\phi_j = x^j$$

- ガウス型基底

$$\phi_j(x) = \exp\left\{\frac{(x-\mu_j)^T(x-\mu_j)}{2h_j}\right\}$$

未学習

学習データに対して、十分小さな誤差が得られない

- 対策　モデルの表現力が低いため、表現力の高いモデルを利用する

過学習

小さな誤差は得られたけど、テスト集合誤差との差が大きい

- 対策１　学習データの数を増やす
- 対策２　不要な基底関数(変数)を削除して表現力を抑止
- 対策３　正則化法を利用して表現力を抑止

正則化法

モデルの過学習を緩和する

ノルム→距離
無い→最小二乗推定量
L2ノルム→Ridge推定量　縮小推定　いくつかのパラメータを0に推定

- リッジ回帰コスト関数

$$\mathrm{J}(\theta) = \mathrm{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

L1ノルム→Lasso推定量　スパース推定　パラメータを0に近づけるよう推定

- ラッソ回帰コスト関数

$$\mathrm{J}(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

正則化パラータの役割

- 小さく→制約面が大きく
- 大きく→制約面が小さく

適切なモデル

- 基底関数の個数
- 基底関数の位置
- 基底関数のバンド幅
- 正則化パラメータ

↓

適切なモデル(汎化性能が高いモデル)は交差検証法で決定

ホールドアウト法

データを学習用とテスト用に二分割する

- 学習用を多くすればテスト用が減り(性能評価の精度が悪くなる)、テスト用を多くすればテスト用が減ってしまう(学習の精度が悪くなる)。
- 手元に大量のデータがある場合を除いて、良い性能を与えないという欠点がある

クロスバリデーション(交差検証)

手元にあるデータをそれぞれm個のグループに分割し、m-1個のグループのデータを使って識別器を学習し、残りの1つのグループで検証を行う

- m回繰り返してそれらの誤り率(平均2乗誤差)の平均を予測値とする
- すべてのデータを学習とテストに利用するので、良い性能予測を行うことができる

ハンズオン　非線形回帰モデル(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/非線形回帰モデル.ipynb)

3 ロジスティック回帰モデル

→分類問題
→教師あり学習

シグモイド関数

- 出力は0~1の値

$$\sigma(x) = \frac{1}{1 + \exp(-ax)}$$

シグモイド関数の性質

- 連鎖律

$$\frac{\partial \sigma(x)}{\partial x} = a\sigma(x)(1 - \sigma(x))$$

ベルヌーイ分布

- ロジスティック回帰ではベルヌーイ分布を利用
- 確率pで1、確率1-pで0をとる
- 離散確率分布(例:コイン投げ)

$$\mathrm{P(y)} = p^y(1-p)^{1-y}$$

最尤推定

- 尤度関数(データは固定しパラメータを変化させる)を最大化するようなパラメータを選ぶ
- 対数をとると微分の計算が簡単
- 同時確率の積が和に変換可能
- 指数が積の演算に変換可能
- 対数尤度関数と尤度関数、最大になる点が同じ
- 尤度関数にマイナスをかけたものを最小化し、最小二乗法の最小化と合わせる

勾配降下法

- 線形回帰モデル(最小2乗法)
- MSEのパラメータに関する微分が0になる値を解析に求める
- ロジスティック回帰モデル(最尤法)
- 対数尤度関数をパラメータで微分して0になる値を求める必要があるのだが、解析的にこの値を求めることは困難

$$w(k+1) = w^k - \eta\frac{\partial E(w)}{\partial w}$$

- パラメータが更新されない→勾配が0→最適解
- パラメータを更新するのにN個全てのデータに対する和を求める必要がある。
- データが巨大になった場合、計算時間がかかる↓

確率的勾配降下法(SGD)

- データを一つずつランダムに選んでパラメータ更新する
- 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回更新できるできるので効率よく最適な解を探索可能

F値

- 再現率(Recall)と適合率(Precision)の調和平均
- 適合率と再現率はトレードオフ→見逃し判定の最適値を適用

$$\frac{2}{\frac{1}{\text{適合率}} + \frac{1}{\text{再現率}}}$$

ハンズオン　ロジスティック回帰(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/ロジスティック回帰.ipynb)

4 主成分分析

- 多変量データの持つ構造をより小数個の指標に圧縮
- 可視化(2・3次元)可能
- 係数ベクトルが変われば線形変換後の値が変化
- 線形変換の変数の分散が最大となる射影軸を探索
- ラグランジュ関数を最大にする係数ベクトルを見つける

ラグランジュ関数

ラグランジュ関数を微分して最適解を求める

$$
\begin{aligned}
\mathrm{E}(a_j) &= a_j^T \mathrm{Var}(\overline{X}) a_j - \lambda(a_j^T a_j - 1) \\
\frac{\partial \mathrm{E}(a_j)}{\partial a_j} &= 2\mathrm{Var}(\overline{X}) a_j - 2\lambda a_j = 0 \\
&\Rightarrow \mathrm{Var}(\overline{X}) a_j = \lambda a_j
\end{aligned}
$$

寄与率

- 第k主成分の分散の全分散に対する割合
- 圧縮したデータがどのくらいロスしたか
- 主成分の分散は元データの分散と一致
- 固有値の和と元データの分散が一致

累積寄与率

- 第k-1主成分まで圧縮した際の情報損失量の割合

ハンズオン　主成分分析(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/主成分分析.ipynb)

5 アルゴリズム

k近傍法(kNN)

- 分類問題のための機械学習
- 多数決で決める

k-平均法(k-means)

- 教師なし
- クラスタリング手法

- 与えられたデータをk個にクラスタに分類する

1. 各クラスタ中心の初期値を設定
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

ハンズオン　k-means(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/k-means.ipynb)

7 サポートベクターマシン

- 2クラス分類のための機械学習手法
- 線形モデルの正負で2値分類
- 線形判別関数と最も近いデータとの距離をマージンという
- マージンが最大となる線形判別関数を求める

$$y = w^T x + b$$

ハンズオン　SVM(https://github.com/woodyinho531/JDLA-Deep-Learning-for-ENGINEER/blob/master/SVM.ipynb)