機械学習

機械学習は以下の3つに分類できる.

- 教師あり学習
- 教師なし学習
- 強化学習

教師あり学習

- 線形回帰
- 正則化
- ロジスティック回帰
- サポートベクターマシン（SVM）
- サポートベクターマシン（カーネル法）
- ナイーブベイズ
- ランダムフォレスト
- ニューラルネットワーク
- k近傍法（kNN）

教師なし学習

- 主成分分析（PCA）
- LSA
- NMF
- LDA
- k-means法
- 混合ガウス分布
- LLE
- t-SNE

強化学習

- TD学習
  - Q学習
  - SARSA

特徴

線形回帰

異なる分布を持つデータ（曲線的な傾向，外れ値をもつなど）に対しても同じ学習パラメータになることがある．データを事前に可視化して検討するべき.

正則化

https://aizine.ai/ridge-lasso-elasticnet/
過学習を防ぐための手法の一つ．過学習の原因として，学習パラメータが極端に大きい（または小さい）値をとってしまうことが挙げられる．そのため，正則化では罰則項（正則化項）を加えそれを抑える．代表的な正則化手法として Ridge 回帰と Lasso 回帰がある．2次式の線形回帰（多項式回帰）に利用した例を以下に示す.

Ridge 回帰

$$R(w) = \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i + w_2 x_i^2))^2 + \alpha(w_1^2 + w_2^2)(\alpha \geq 0)$$

αは正則化の強さを表す.

Lasso 回帰

学習パラメータが0になりやすいため，特徴量を選択できる.

$$R(w) = \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i + w_2 x_i^2))^2 + \alpha(|w_1| + |w_2|)(\alpha \geq 0)$$

上記2つを混ぜた Elastic Net というのもある.

Elastic Net

$$R(w) = \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i + w_2 x_i^2))^2 + \lambda \sum_{j=1}^{2} (\alpha |w_j| + (1-\alpha) w_j^2)(\lambda \geq 0, 0 \leq \alpha \geq 1)$$

ロジスティック回帰

「回帰」とついているが分類に適用される。シグモイド関数$\sigma(z)$を用いる。

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

確率pを$p = \sigma(w^T x + w_0)$で計算。損失関数は以下。
https://www.hellocybernetics.tech/entry/2017/06/20/135402

$$L(y, f(x, w)) = \log(1 + \exp(-y f(x, w)))$$

決定境界（分類結果が切り替わる境目）は50%になる箇所。

線形サポートベクトルマシン（Linear SVM）

分類と回帰どちらにも使える。マージン最大化により決定境界ができるだけデータから離れるように学習する。マージン内にデータが入り込むことを許容するかどうかでハードマージン、ソフトマージンに分けられる。マージン上とその内側のデータをサポートベクトルと呼ぶ。

サポートベクトルマシン（カーネル法）

曲線のような複雑な決定境界を学習できる。線形分離不可能なデータを線形分離可能な高次元空間に移すことで決定境界を得る。カーネル関数により高次元空間を構築する。線形カーネル、シグモイドカーネル、多項カーネル、RBFカーネルなどがある。何を特徴量として見ているのかわからなくなるので、精度が要求される場合に利用すると良い。

ナイーブベイズ（単純ベイズ）

自然言語の分類に利用されることが多い。文章を単語の集合に分解し、one-hot vectorとして扱う。分類（カテゴリ）毎に各単語の出現確率を計算する。このとき確率0は0.01などの小さい値にする（スムージング：学習データを増やすとその単語が出現した可能性があるため）。「すべての特徴量が互いに独立」という仮定があるが、それが成り立たないであろう実データに対しても、良い結果を出す場合がある。
https://avinton.com/academy/naive-bayes/
https://www.slideshare.net/HirotakaHachiya/14-122960312

ランダムフォレスト

ニューラルネットワーク

k近傍法（kNN）

- TD 学習
  - TD(0) 法
  - n ステップ TD 法
  - TD(λ) 法
    - Forward view
    - Backward view
  - Q 学習
  - SARSA

https://qiita.com/shionhonda/items/ec05aade07b5bea78081
https://qiita.com/triwave33/items/277210c7be4e47c28565

- 強化学習は動的計画法、モンテカルロ法、TD学習に分類される

TD 学習

- Temporal Difference（時間的差分）
- Q学習、Sarsaが有名
- 状態価値関数$V(s)$を以下の式で更新する（**TD(0)法**と呼ぶ）
- 行動はランダム

TD(0)法

$$\begin{aligned} V(s_t) &\leftarrow V(s_t) + \alpha\{R_{t+1} + \gamma V(s_{t+1}) - V(s_t)\} \\ &= V(s_t) + \alpha\delta_t \end{aligned}$$

nステップTD法

- http://yagami12.hatenablog.com/entry/2019/02/22/210608
- TD(0)法において報酬をnステップ先まで拡張したもの

$$V(s_t) \leftarrow V(s_t) + \alpha\{R_t^{(n)} - V(s_t)\}$$
$$R_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(s_{t+n})$$

TD(λ)法

- https://qiita.com/sconeman/items/b1aacfa924e52102e9d6
- https://yamaimo.hatenablog.jp/entry/2015/12/11/200000

Forward view

- モンテカルロ法同様、エピソード完了まで学習できない
- 以下のn個の報酬を考える
  - $R_t^{(1)} = R_{t+1} + \gamma V(s_{t+!})$
  - $R_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(s_{t+!})$
  - ...
  - $R_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(s_{t+n})$
- これらを$\lambda^{n-1}(0 < \lambda < 1)$で重み付けする（近いステップの報酬を重視するため）

$$R_t^{(1)} + \lambda R_t^{(2)} + \lambda^2 R_t^{(3)} + \ldots$$

- $R_t^{(n)}$の係数について、$1 + \lambda + \lambda^2 + \ldots \rightarrow \frac{1}{1-\lambda}$なので、正規化するために$(1-\lambda)$をかけて、これを収益とする

$$R_t^{\lambda} = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t$$

  - 時間ステップTを終端とする
  - R_t：終端に達した後のnステップ収益
  - $\lambda = 0$のとき、$R_t^{\lambda} = R_t^{(1)}$：TD(0)法
  - $\lambda = 1$のとき、$R_t^{\lambda} = R_t$：モンテカルロ法

Backward view

- http://incompleteideas.net/book/first/ebook/node75.html
- 1ステップ毎に全状態を更新する（これがForwardと違うところ）
- Eligibility traces （適格度トレース） $E_t(s)$ を導入する
  - 初期値は0

$$E_0(s) = 0$$

  - 現在の状態のときのみ1を足す（γ：時間割引率、λ：trace-decay parameter）

$$E_t(s) = \begin{cases} \gamma\lambda E_{t-1}(s) & (s \neq s_t) \\ \gamma\lambda E_{t-1}(s) + 1 & (s = s_t) \end{cases}$$

- 状態価値関数 $V(s)$ を以下の式で更新する

$$V(s) \leftarrow V(s) + \alpha\delta_t E_t(s) \quad (\forall s)$$

Q 学習

- 方策オフ型（off-policy）
- 行動価値関数$Q(s, a)$を以下の式で更新する
- ε-greedy法などで行動を決定

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\{R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\}$$

SARSA

- 方策オン型（on-policy）
- 行動価値関数$Q(s, a)$を以下の式で更新する
- Q学習と違うのは、Q値の更新に次の行動a_tに対応する$Q(s_t, a_t)$を使うかどうか
  - Q学習：更新→行動選択
  - SARSA：行動選択→更新

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\{R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\}$$