Hands on Machine Learning with Scikit-Learn &TensolFlow

Chapter 5

Support Vector Machines

Created by Yusuke FUJIMOTO

はじめに

- この資料は「Hands-On Machine Learning with Scikit-Learn and TensorFlow - O'Reilly Media 」を読んだ際の（主にソースコードに関する）簡単な解説を残したものです。
- 全部を解説したわけではないので注意
- 余裕があればソースコード周りの背景知識もまとめたい
- 何かあったら yukkyo12221222@gmail.com まで

Chapter 5

Support Vector Machines

今回のポイント

- SVM（Support Vector Machine）は超メジャーな分類・回帰モデル
- ここまで扱った回帰モデルの損失関数（コスト関数、エラー関数）

$$\mathrm{cost}(\mathbf{x}) = \sum_{i=i}^{T} (f(\mathbf{x}_i) - \mathbf{y}_i)^2$$

- でもこれは、$f(\mathbf{x}_i)$ と $\mathbf{y}_i$ の距離を表しているわけではない

- 分類境界（ここでは超平面という）と各データの距離（マージン）に着目しているのが SVM の特徴
- あとサポートベクトルと呼ばれる一部のデータ（1 つとは限らない）で識別境界が決定されるので、疎なデータに強い
- また予測するときや新しくデータを追加したときは、サポートベクトルだけ見ればよいため効率的。

数学的補足

- (a, b) 行列（ベクトル）と (b, c) 行列（ベクトル）をかけると (a, c) 行列になる

- 識別境界 : $\omega^{\mathrm{T}}\mathbf{x} + b = 0$
    - 超平面からの距離 = ω の一次元ベクトルに写像しているだけ

Large margin classification

- 特にコメントなし

Sensitivity to feature scales

- SVM もスケーリング大事

Sensitivity to outliers

外れ値に対してどのように動作するのか確認する

- 外れ値が片方のクラスの散布に食い込んでいる場合
  - 線形分離不可能になっている
- 外れ値が片方のクラスの散布に非常に近い場合
  - 本来の境界より、片方のクラスに寄ったものになっている

Large margin vs margin violations

マージンを大きくとるかマージン違反を許すか？？？（該当箇所を読む）

- 実問題では綺麗に線形分離できるとは限らない
    - 綺麗に分離できる場合しか境界を引けない（マージン違反を決して許さない） = ハードマージンSVM
    - マージン違反したデータに対してはペナルティを与えて、一応境界が引けるようにした SVM = ソフトマージンSVM
    - LinearSVC はソフトマージンSVM。実は。

- LinearSVCのオプションには `C` がある。これが正則化の強さ（線形回帰モデルにおける α の逆数にあたる）を表す。大きいほど強い。
- 小さいほどマージン違反を許し、大きいほど許さない
    - $C = \infty$ の場合にハードマージンSVMと等価
    - 先ほどの「Sensitivity...」では $C = 10^9$ となっており実質ハードマージンSVM
- $C = 1$ の場合と $C = 100$ の場合でどう変わるのか確認する。

Non-linear classification

非線形な分離はできないのか？　→　できる

- higher_dimensions_plot : 一次元では線形分離ができなかったけど、二次元に射影したら線形分離できるようになった
- Chapter 4 では polynomial feature で高次元特徴空間に射影して線形分離した
    - polynomial svc
- 他のカーネルとして代表的なもの: RBFカーネル
    - gamma が小さいほど単純な境界になる

Regression

- クラス分類だけでなく回帰もできる。
- 予測誤差を最小化するような超平面（線）を求める
- ϵ は超平面からの距離の内エラーを許容する広さを示す
    - 実際 $\epsilon = 1.5$ の方が帯が広いのが分かる

Under the hood

- 学習や超平面の内訳。理論寄りのお話。
- SVM ではマージンを最小化するようなモデルを学習したい。
- つまりSVM における学習とは、マージン関数を最小化したいという最適化問題を解くことと同じ。
- また、最適化問題を解くにあたって、マージン関数をそのまま扱うのではなく、双対問題という、答えが同じになるけどより簡単に解けそうな問題に落とし込む

SVM で最適化したい対象

- マージン（境界に一番近い点から境界までの距離）を最大化したい
- 超平面 $f(\mathbf{x}) = \ \mathbf{w}^T\mathbf{x} + b = 0$ から、点 $\mathbf{a}$ までの距離は $\frac{|f(\mathbf{a})|}{\|\mathbf{w}\|}$ で与えられる（次ページ参照）
- よって次の問題を解くことと同じ

$$\arg\max_{\mathbf{w},b}\left(\frac{1}{\|\mathbf{w}\|}\min_n[t_n(\mathbf{w}^T\mathbf{x}+b)]\right)$$

- t_n はラベル（負なら-1、正なら1）

$\mathbf{w} \to k\mathbf{w}$、$b \to kb$ としても超平面からの距離 $\frac{|f(\mathbf{x})|}{\|\mathbf{w}\|}$ は変わらないので、最も近い点について以下を成立させて良い

$$t_n(\mathbf{w}^T\mathbf{x} + b) = 1$$

よって先程の問題は下記のようになる

$$\arg\max_{\mathbf{w},b} \left(\frac{1}{\|\mathbf{w}\|} \right)$$

$$\Rightarrow \arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$\frac{1}{2}$ はあとで計算が楽になるためつけてる

補足：超平面からの距離

- ある点 $\mathbf{x}_0$ と平面 $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$ との距離を考える。
- 点 $\mathbf{x}_0$ に k 倍にした $\frac{\mathbf{w}}{|\mathbf{w}|}$ を足すと平面にぶつかる。この点を $\mathbf{z}$ とする。また $\frac{\mathbf{w}}{|\mathbf{w}|}$ は大きさ 1 であることから、求める距離は $\left|k\right|$ である。
- 点 $\mathbf{z}$ は平面上の点なので $f(\mathbf{z}) = 0$

- よって以下の式が成り立つ

$$f(\mathbf{z}) = f(\mathbf{x}_0 + k\frac{\mathbf{w}}{|\mathbf{w}|})$$

$$= \mathbf{w}^T\mathbf{x}_0 + k\frac{\mathbf{w}^T\mathbf{w}}{|\mathbf{w}|} + b = \mathbf{w}^T\mathbf{x}_0 + b + k|\mathbf{w}| =$$

$$= f(\mathbf{x}_0) + k|\mathbf{w}| = 0$$

$$\Rightarrow |k| = \frac{|f(\mathbf{x}_0)|}{|\mathbf{w}|}$$

※ $\mathbf{w}^T\mathbf{w} = |\mathbf{w}|^2$

補足：超平面からの距離（PRML上より引用）

180　第4章　線形識別モデル

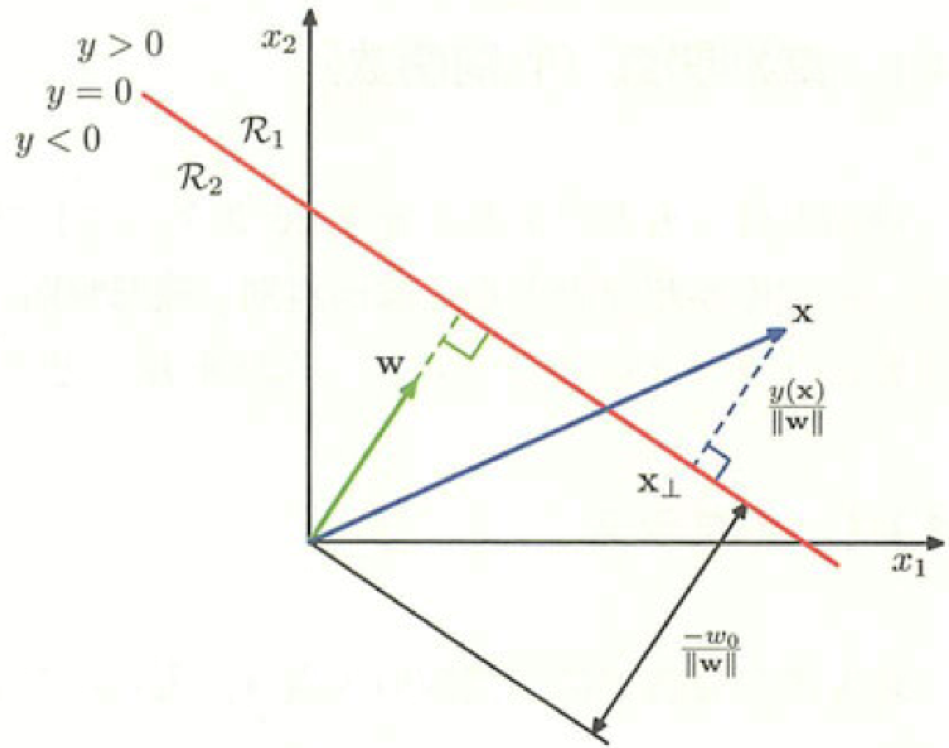

図 4.1　2次元線形識別関数の幾何．赤で示された決定面は $\mathbf{w}$ に垂直である．原点からの決定面までの距離は，バイアスパラメータ w_0 によって制御される．また，決定面から一般の点 $\mathbf{x}$ への直交距離は $y(\mathbf{x})/\|\mathbf{w}\|$ で与えられる．

ハードマージン線形SVM分類の最適化対象

- 目的関数

$$\text{minimize}_{(\mathbf{w},b)}\ \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

- 制約条件

$$t^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 \text{ for } i = 1, 2, \cdots, m$$

ソフトマージン線形SVM分類の最適化対象

- 目的関数

$$\text{minimize}_{(\mathbf{w},b,\zeta)}\ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\zeta^{(i)}$$

- 制約条件

$$t^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)} + b\right) \geq 1 - \zeta^{(i)}$$

$$\text{and } \zeta^{(i)} \geq 0$$

$$\text{for } i = 1, 2, \cdots, m$$

凸二次計画問題

- 目的関数: $\mathbf{min}\ \frac{1}{2}\mathbf{p}^T\mathbf{H}\mathbf{p}\ +\ \mathbf{f}^T\mathbf{p}$
- 制約条件: $\mathbf{A}\mathbf{p} \leq \mathbf{b}$
    - $\mathbf{p}: n_p$（パラメータ数）次元のベクトル
    - $\mathbf{H}: n_p \times n_p$ 行列
    - $\mathbf{f}: n_p$（パラメータ数）次元のベクトル
    - $\mathbf{A}: n_c \times n_p$ 行列。n_c は制約の数
    - $\mathbf{b}: n_c$ 次元のベクトル

ハードマージンSVMにおける各パラメータについて

- $n_p = n + 1 \quad n$ は特徴の数、$+1$ はバイアス分
- $n_c = m \quad m$ はトレーニングデータの数
- H は単位行列。ただし一番左上の要素は 0
  - バイアス分を無視するため
- $\mathbf{f} = \mathbf{0}$、$\mathbf{b} = \mathbf{1}$
- $\mathbf{a}^{(i)} = -t^{(i)}\dot{\mathbf{x}}^{(i)}$
  - $\dot{\mathbf{x}}^{(i)}$ は $i = 0$（バイアス分）のときだけ $\mathbf{1}$ で、他の場合は $\mathbf{x}^{(i)}$ と同じ

双対問題について

- 主変数がたくさんあって制約条件が少なければ、双対問題の方が変数が少なくできる
- すると、主問題より楽に解ける可能性が高い
- 双対問題の方はよく手に入るソルバーで解けるらしい

$$\mathrm{minimize}_{\alpha}\ \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j t_i t_j \mathbf{x}_i^T\mathbf{x}_j\ -\sum_{i=1}^{m}\alpha_i$$

$$\text{subject to: }\ \alpha_i \geq 1 \text{ for } i = 1, 2, \cdots, m$$

双対問題を解いたあと

以下のようにして $\mathbf{w}$ と b を計算できる。ハットは推定値などの意味を含む

$$\hat{\mathbf{w}} = \sum_{i=1}^{m} \hat{\alpha}_i t_i \mathbf{x}_i$$

$$\hat{b} = \frac{1}{n_S} \sum_{i \in S} \left(1 - t_i(\hat{\mathbf{w}}^T \mathbf{x}_i)\right)$$

- S は $\hat{\alpha}_i > 0$ となる i の集合。すなわちサポートベクトル一覧を表す。また $n_S = \left|S\right|$。その他は 0

カーネル SVM における予測（判定）

$$h_{\hat{\mathbf{w}},\hat{b}}(\phi(\mathbf{x}_{new})) = \hat{\mathbf{w}}^T\phi(\mathbf{x}_{new}) \ + \hat{b}$$

$$= \left(\sum_{i=1}^{m} \hat{\alpha}_i t_i \phi(\mathbf{x}_i)\right)^T \phi(\mathbf{x}_{new}) \ + \hat{b}$$

$$= \sum_{i=1}^{m} \hat{\alpha}_i t_i \left(\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_{new})\right) \ + \hat{b}$$

$$= \sum_{i \in S} \hat{\alpha}_i t_i K(\mathbf{x}_i, \mathbf{x}_{new}) \ + \hat{b}$$

サポートベクトルとカーネル関数で予測できる

カーネル使うと何が嬉しいの？

- 普通に高次元に変換してそのまま解けば良いのでは？
    - すごく大きい次元（例えば無限次元）に変換してから計算するのは無理
    - でも高次元上での **内積の値** であれば比較的簡単に計算できる場合が多い
    - SVMを学習する上で解きたい最適化問題では、最終的には高次元上でのベクトルは扱わず、 **内積の値** だけ扱っていた → カーネルトリック
    - 高次元のベクトルを直接扱わずに高次元上での計算ができた！

- 各カーネルとその中身

カーネル	数式
Linear	$K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \cdot \mathbf{b}$
Polynomial	$K(\mathbf{a}, \mathbf{b}) = \left(\gamma \mathbf{a}^T \cdot \mathbf{b} + r\right)^d$
Gaussian RBF	$K(\mathbf{a}, \mathbf{b}) = \exp\left(-\gamma \lVert \mathbf{a} - \mathbf{b} \rVert^2\right)$
Sigmoid	$K(\mathbf{a}, \mathbf{b}) = \tanh\left(\gamma \mathbf{a}^T \cdot \mathbf{b} + r\right)$

- γ と r は違うので注意

Hinge loss

誤差関数のひとつ。引数が正の数ならそのまま出力し、それ以外なら 0 を出力する。
大きい意味はないので説明は省略。

Extra material

Linear SVM classifier implementation using Batch Gradient Descent

そのうち追記する

Extra material questions

- 1: SVM の基本的な考えは？
  - マージン（超平面とデータの距離）を最大化
- 2: サポートベクトルって何？
  - 超平面から一番近い各クラスのデータ（ベクトル）。それ以外のベクトルは超平面の決定に関与しない
- 3: SVM に入力するときはどうしてスケーリングが大事なのか
  - SVM はなるべく広い帯を作ろうとするので、スケーリングがされていないと小さい特徴を無視する傾向があるため

- 4: SVM 分類器はデータの分類時に、確信度スコアのようなものを出力できるか。
    - できる。ただし直接確率に変換できるわけではない。Scikit-learn で SVM モデルを構築する際に `probability = True` を指定すると、学習後に SVM のスコアに対してロジスティック回帰によって確率を測定する。
    - `predict_proba()` や `predict_log_proba()`

- 5: データ数が100万、特徴数が数百のときには、主問題と双対問題のどちらを解くべきか？
  - まずカーネル SVM は双対問題しか使えない
  - よって線形 SVM の場合のみ考える
    - この場合は主問題を解く。
    - データの特徴次元よりデータ数のほうがはるかに大きいため

- 6: もし RBF カーネルSVM をトレーニングしたときに、訓練データに underfit していた場合は、γ を増やすべきか、減らすべきか？
    - 正則化が強すぎるので、これを弱めるためには gamma や C を増やす
- 7: 既存のソルバーを使ってソフトマージン 線形 SVM を解く際は、二次凸問題のパラメータをどのように設定するべきか？
    - よくわからないので省略

- 8: 線形分離可能なデータを LinearSVC で学習する。その場合同じデータセットを SVC と SGDClassifier で学習させると、大体同じモデルが得られることを確認せよ
- notebook

- 9: SVM 分類器で MNIST データセットを学習する。SVM は二値分類器なので、10 個のクラスを分類するために one-versus-all 分類器を使わなければならない。あなたはプロセスを速くするために小さい評価データセットを使ってハイパーパラメータを調整したい。どのようにやるか？そして精度はどうなるか？
    - 次ページの方法を試す

1. 線形SVM（LinearSVC）でそのまま学習　→　0.8263（悪い）
2. 上記のまま、データをスケーリング　→　0.9224（まあまあ）
3. 上記のデータのまま量は6分の1にして、RBFカーネルSVM（SVC）を適用　→　0.9462
4. 上記について、gamma と C はデフォルトのままだったので、 範囲を決めてデータ数 1000 で randomized search を行って良いパラメータを探す。その後全データで学習した　→　0.9710 （悪くない）。ただしテストデータだと 0.9997 なので、やや overfitting 気味か。

- 10: California housing dataset (Chapter 2 で扱った) に対し、サポートベクター回帰を使ってモデルを構築して学習させよ
    - notebook

Exercise solutions