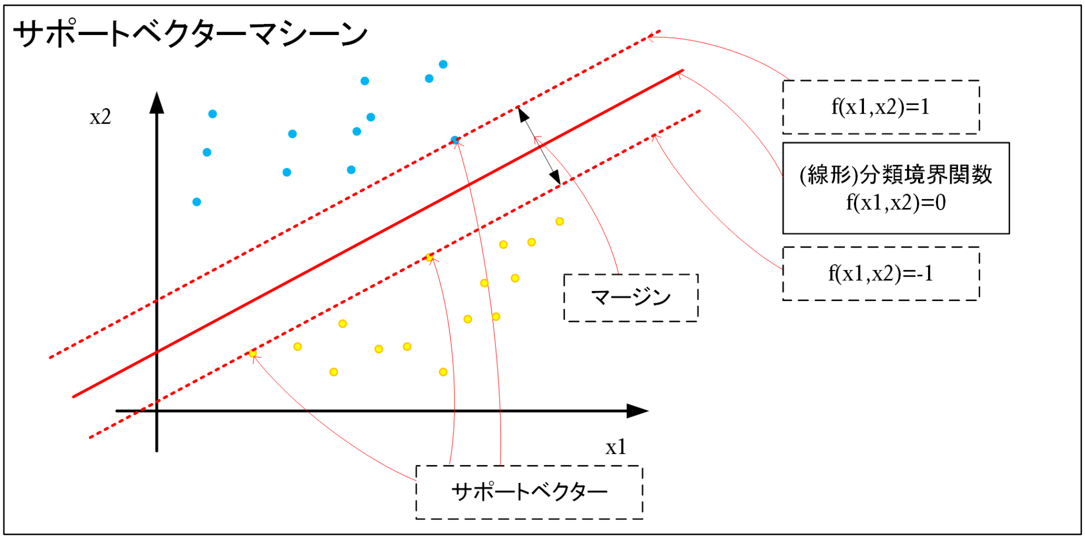
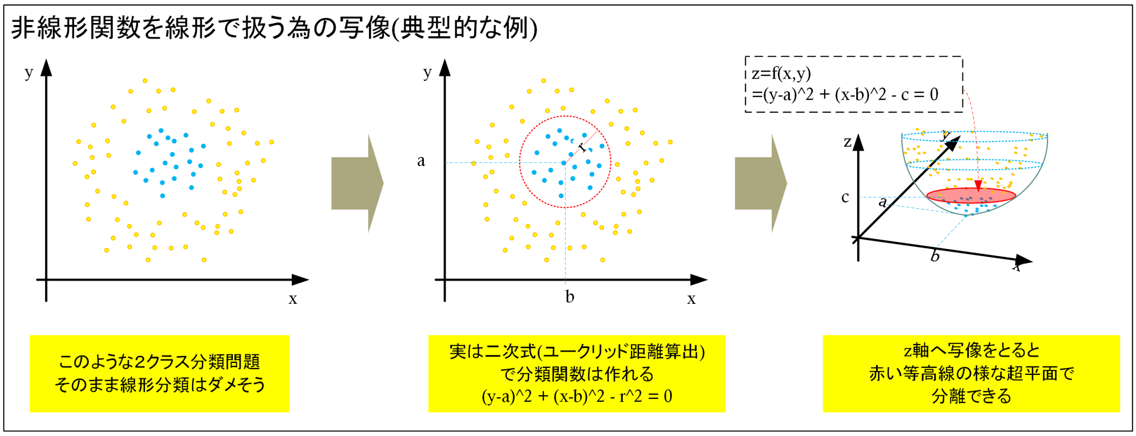
1　サポートベクターマシーンとは

・教師あり２クラス分類手法となるが、回帰や教師無し問題へも応用されている。
・２クラス分類の場合、分類境界関数を決定する (マージン付き)。
・分類境界は線形 (あるいは超平面) となる。
・マージン上のデータ (分類境界を確定する為のデータとなる) をサポートベクターと呼ぶ。



・その空間で線形分離不能の形ならば高次空間へ写像をとり線形な分類境界をとる。
・後述のカーネルトリックにて高次空間写像した分類境界関数の計算量を削減する。

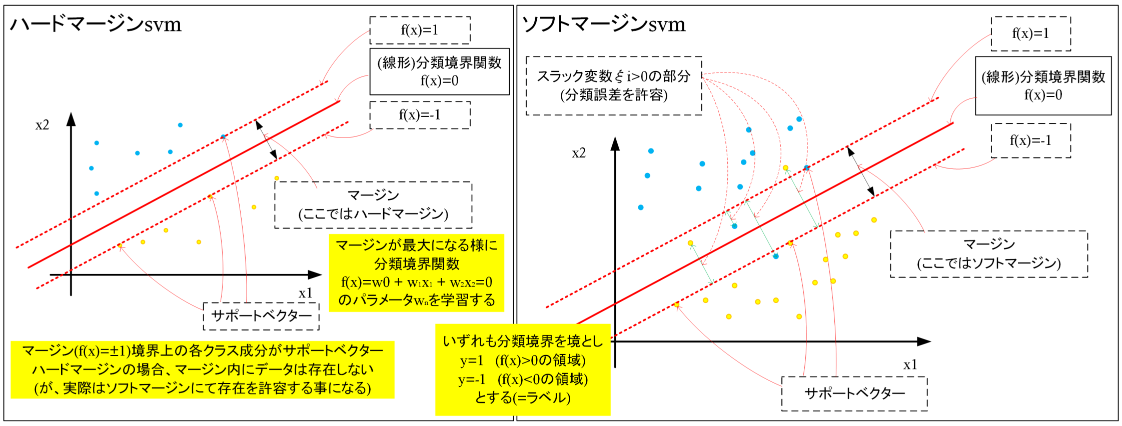


2 マージン

・マージンにはソフトマージンとハードマージンが定義される。

・ソフトマージンはスラック変数によるペナルティ項を設けて誤分類を許容させる。

・どちらもマージン最大になる様に分離境界を決定する。



・ソフトマージンを含んだ最適化問題は以下数式となる

(C は正則化係数 (ハイパーパラメータ))

目的関数: $\min[\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i]$

制約条件: $y_i[\mathbf{w}^T\mathbf{x}_i + b] \geq 1 - \xi_i, \qquad \xi_i \geq 0 \qquad (i = 1, \cdots, n)$

・ハードマージンの最適化問題はソフトマージンよりスラック変数 ξ_i を除去 (=0) した以下数式となる

(これは正則化係数 $C \to \infty$ としたソフトマージン svm と等価である)

目的関数: $\min[\frac{1}{2}\|\omega\|^2]$

制約条件: $y_i[\mathbf{w}^T\mathbf{x}_i + b] \geq 1, \qquad (i = 1, \cdots, n)$

3 双対問題

・ラグランジュの未定乗数法を用いて svm の主問題 (最適化問題) を双対表現に変換して解く。

(これにより変数削減でき、後述のカーネル関数が利用できる (=非線形データの分類が可能になる))

・ラグランジュの未定乗数法は

「目的関数 $f(x)$ を最小にする点を、n 個の不等式制約 $g_i(x) \leqq 0$ の元で求める」

・つまり svm の最適化問題をそのままラグランジュ関数に当てはめてしまえば双対関数に変換できる

・ソフトマージン svm の最適化問題をラグランジュ関数に当てはめると

$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i[w^T x_i + b] - 1 + \xi_i] - \sum_{i=1}^{n}\mu_i\xi_i$

(ただし、$\alpha_i \geq 0, \quad \mu_i \geq 0 \quad (i = 1, \cdots, n)$)

・双対問題とは上記ラグランジュ関数に対する最適化問題となる

$\max_{\alpha,\mu} \min_{w,b,\xi} L(w, b, \xi, \alpha, \mu)$

・最適解は上記ラグランジュ関数の主変数 w, b, ξ に対して偏微分した結果が 0 となるはず

(つまり以下の様に書き換えされる)

$\max_{\alpha,\mu} L(w^*, b^*, \xi^*, \alpha, \mu)$

・上記双対問題は結局以下の様に整理され、α のみの関数に置き換えできる

$\max_{\alpha,\mu} L(w^*, b^*, \xi^*, \alpha, \mu) = \max_{\alpha}[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j + \sum_{i=1}^{n}\alpha_i]$

(ただし $\sum_{i=1}^{n}\alpha_i y_i = 0, \; 0 \leq \alpha_i \leq C \quad (i = 1, \cdots, n)$)

4　カーネルトリック

・前述で得た双対関数を高次元空間について解く $\Longrightarrow$ データを高次関数に置き換える (以下)

$$\max_{\alpha,\mu} L(w^*, b^*, \xi^*, \alpha, \mu) = \max_{\alpha} \left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i\right]$$

・高次関数の内積計算量を減らす $\Longrightarrow$ 高次関数の内積をカーネル関数に置き換える (カーネルトリック)

$$\max_{\alpha} \left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{n} \alpha_i\right]$$

・決定境界関数もカーネル関数を用いて表現する

$$f(x) = w^T\phi(x) + b = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

・代表的なカーネル関数は以下の 3 つ (ただし c, d, γ はハイパーパラメータ)

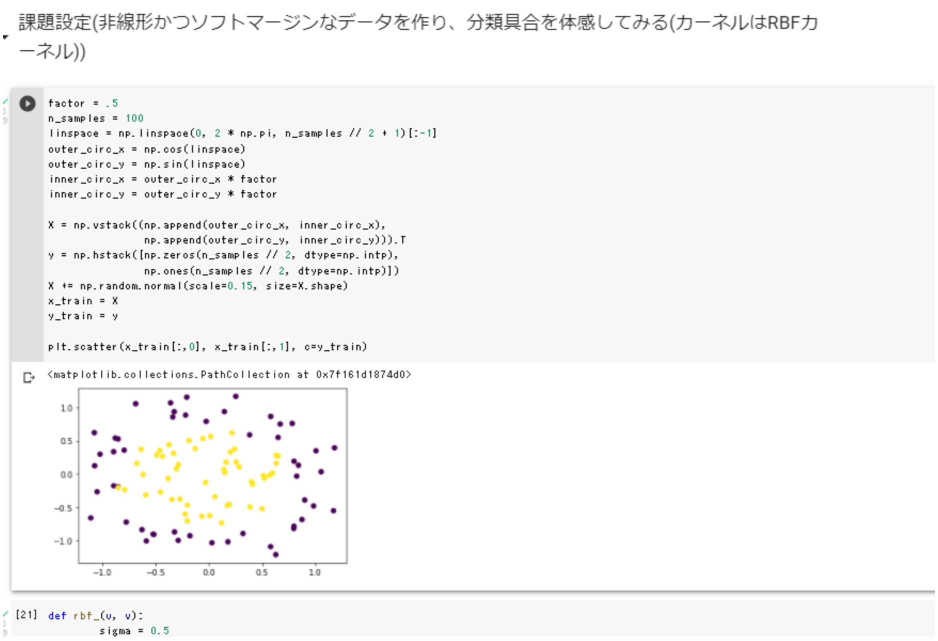
多項式カーネル：　$K(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{x}_i^T\mathbf{x}_j + c]^d$

ガウスカーネル：　$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$

シグモイドカーネル：$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(b\mathbf{x}_i^T\mathbf{x}_j + c)$

5 実習

ハンズオンコードに対して非線形かつソフトマージンの実装を追加 (以下)



（途中省略）

コードは本ドキュメントと同じレポジトリに提出する

https://github.com/toruuno/report_ml/blob/master/np_svm.ipynb