

Optimization Objective

alternative view of logistic regression

SVM(サポートベクトルマシン)というよりよいアルゴリズムの目的関数を考える。まず、論理回帰の場合の処理を思い出してみる。

$$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$$

と仮定した時に $y=1$ ならば $h_{\theta}(x) \approx 1$ かつ $\theta^T x \gg 0$ となってほしく、
 $y=0$ ならば $h_{\theta}(x) \approx 0$ かつ $\theta^T x \ll 0$ となってほしい。

ここで $y=1, y=0$ の場合の sigmoid 関数をそれぞれ 2 本の直線で近似することを考える。

すなわち、 $-\log h_{\theta}(x^{(i)})$ を 2 本の直線からなる $cost_1(\theta^T x^{(i)})$ で近似し、同様に $-\log(1 - h_{\theta}(x^{(i)}))$ で近似する。

また論理回帰では正規化項に重みをつけていたが、SVM では逆の項に重みをつける。

従って論理回帰の目的関数が

$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} (-\log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) (-\log(1 - h_{\theta}(x^{(i)})))) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

となり、SVM の目的関数が

$$\min_{\theta} C \left[\sum_{i=1}^m y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

となる。この定数の置き方において C と λ は逆数の関係にある。

SVM は確率を計算するのではなく、目的関数を最小化する θ をもとめ、かつ $y=0$ か1の予想までする。

Large Margin Intuition

「SVM は大きなマージンを持つ分類器」という命題を説明する。

説明しやすくするために C が十分大きいという条件で先の目的関数の最小値問題を考える。

前回の授業で行った近似を用いると以下の最小値問題を考えることと同値になる。

$$\min_{\theta} \sum_{j=1}^n \theta_j^2$$

$$\text{条件 } \begin{cases} \theta^T x \geq 1 & y=1 \text{ のとき} \\ \theta^T x \leq -1 & y=0 \text{ のとき} \end{cases}$$

この条件で SVM を実行するとデータセットからの距離(定義が不明)が最も大きい境界を得ることができる。

Mathematics Behind Large Margin Classification

SVM の裏にある数学

ベクトルの内積とは

$$\boxed{u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}}$$

$$\boxed{v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}}$$

に対して $u^T v$ を考える。 v の u への正射影ベクトルの長さを p (符号付長さ)とすると

$$u^T v = p \cdot \|u\| = u_1 v_1 + u_2 v_2$$

SVMでは以下の条件で $\theta^T p^{(i)}$ の最小値問題を考えることになる。

$$\begin{cases} \|\theta\| \cdot p^{(i)} \geq 1 & y=1\text{のとき} \\ \|\theta\| \cdot p^{(i)} \leq -1 & y=0\text{のとき} \end{cases}$$

$p^{(i)}$ は θ 方向へのそれぞれのデータセット(ベクトル)の正射影の長さである。

Kernel I

例えばデータセットの変数が x_1, x_2 しかなくとも分析に必要な関数に $x_1 x_2$ や $x_1^3 x_2$ が必要になるときもある。

どのように変数を選べばいいのかを今回考える。

変数が x_1, x_2 しかない場合を考える。

分析に用いるの関数を $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \dots$ とする。 f_i には x_1 か x_2 からなる関数が含まれる。この関数が0より大きければ $y=1$ と予想し、0より小さければ $y=0$ と予想する。

x_1, x_2 を座標軸にもつ座標平面上で任意の点を手動で選ぶ(ランドマークという)[選び方は次以降の授業で]。選んだランドマークを用いて以下のようにカーネル(これはガウスカーネル)を定義する。 f_i は x と l_i がどれだけ近いか示している。与えられた x に対して

$$f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right) = \exp\left(\frac{\sum_{j=1}^n \|x_j - l^{(i)}\|^2}{2\sigma^2}\right)$$

Kernel II

具体例を用いてカーネルを用いたSVMの実装を考える。データセットが m 個あるとき、その全てをランドマークとしてカーネルを考えるべきであり、 $f \in \mathbb{R}^{m+1}, \theta \in \mathbb{R}^{m+1}$ となる。

すなわち、 $\theta^T f = \theta_0 f_0 + \theta_1 f_1 + \theta_2 f_2 + \dots \geq 0 \Rightarrow y = 1$ となる。

SVMでは以下の式の最小値問題になる。

$$\min_{\theta} C \sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1 - y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^m \theta_j^2$$

正規化のところでも書かれた通り、正規化項で θ_0 は計算に含まない。

また、SVMを計算する実際のソフトでは正確には $\theta^2 = \theta^T \theta$ を計算しているわけではなく、カーネルに由来するMを用いて $\theta^T M \theta$ を計算している。

定数Cと σ^2 の選び方

定数Cについて

Cが大きい時(λが小さい時) lower bias, high variance → overfitしやすい

Cが小さい時(λが大きい時) high bias, lower variance → underfitしやすい

σ^2 について

σ^2 が大きい時 f_i がゆっくりと変化するのでランドマークに近いカーネルの値が大きくなりにくい。→higher bias, lower variance

σ^2 が小さい時 f_i が早く変化するのでランドマークから少しでも離れるとカーネルの値が大きくなる。→lower bias, higher variance

Using A SVM

SVM実装するときにしなくてはいけないこと

最適化問題をとくライブラリ(liblinear, libsvmなど)を用いることになる。

- 必要なのは定数Cを決める。
- カーネルを使うかどうかを決める。
 - カーネルを使わない(linear kernel)で実装する場合(変数が多く、データセットはそこまで多くない場合)は何も設定する値は存在しない。
 - カーネルを使うとき(変数は少ないものの、データセットは多い場合) σ^2 を決める必要がある

時にはカーネル(ガウスカーネル)を計算するコードを書く必要がある。そのときに変数のスケーリングをしなくてはならない。ex. 住宅の価格予想で敷地面積の項と部屋数の項をそのまま計算に用いると、敷地面積の項の寄与が非常に

大きくなりいい予測ができなくなってしまう。

その他のカーネル

全てのカーネルはmarcerの定理を満たさないといけないことになっている。
満たしていないとSVMがうまく動かなくなってしまう。

ガウスカーネル以外のカーネルを紹介する

- 多項式カーネル
 - similarity(x, l) = $(X^T l + C_1)^{C_2}$ で表され、パラメータは C_1 と C_2 の二つになる。
- String kernel
 - 文字列の類似性を測るために用いる。
- chi-square kernel
- histogram intersection kernel

multi-class classification

多くのSVMマシンではすでにマルチクラスの分類機能が実装されているはずだが、もし実装されていない場合onevsALLのときのようにして実装すればよい

どのアルゴリズムを使うか

- 変数が少なく、データセットの数が中程度のとき(変数の個数が1-1000, データセット数が10-50000くらい)
 - ガウスカーネルを用いたSVMを使うべき
- 変数の少なく、データセット数が多いとき(変数の個数が1~1000, データセット数が50000以上)
 - 変数の数を増やしてから論理回帰を使う(データセット数が増えて行くにつれてSVMが遅くなっていく。)-ニューラルネットワークは、ほとんどの場合でうまく行くが学習に時間がかかるてしまう