平成 28 年度 3 回生後期実験 (エージェント)
課題 3 SVR の作成

村田 叡

2016/11/4

1　プログラム概要

今回の課題では、SVR のプログラムを作成した。データファイルを読み込み、回帰式を出力することができる。また、引数を変えれば交差検定を行うこともできる。以下にその詳細を述べる。

1.1　実装言語の変更について

前回までの SVR で提出した課題は全て Python3 で実装していたが、速度に不満を感じたため、プロッティング部分は Python3 に任せ、計算のコアの部分は C++14 で実装しなおした。SVM も Python3 から C++14 に書き直した時に、Clang++ による最適化 (O3), 並列化 などの要素により、目安としてサンプルデータでの交差検定が 16 倍速くなったので、C++14 にて書く価値はあると判断した。なお SVM の部分については、今回のレポートには関係がなく、また、Python3 のものとほとんど同様のインターフェースのため省略する。

1.2　プログラムの起動方法及び実行例

readme.md の項 requirements を参照のこと

2　外部仕様

プロッティングとデータ抽出には Python3 を、計算のコア部分には C++14 を用いている。以下、そのプログラムの CUI について述べる。

2.1　py3/plotdata.py

このコードは dat ファイル (形式は実験のデータ通り) を読み込み、そのデータをプロットすることに特化したコードである。–1d は一次元データを折れ線グラフでプロットし、–2d は SVM 用の二次元データを、–3d は二次元データを三次元散布図でプロットする。–save オプションをつけると、結果の画像ファイルを保存できる。ファイルを 2 つ指定して、2 つのデータを重ねて表示して差異を確認することもできる。実際の使用例は、readme.md の項 example を参照のこと。

2.2 実行可能ファイル svr

このファイルは dat ファイルを読み込み、SVR を作成する。–plot オプションをつけると、結果を dat ファイル形式で出力し、–cross オプションをつけると、交差検定を行う。–p, –c, –eps オプションを指定して、値を指定することができる。gauss, polynomial, linear などを引数に入れておくことでカーネル指定することができる。mean_abs,mean_square,coefficient などを引数に入れておくことで、交差検定の指標を指定することができる。実際の使用例は、readme.md の項 example を参照のこと。

3 内部仕様

全体的にコードが多いので、各ファイルの概要のみを述べる。

3.1 py3/plotdata.py

docopt を用いてコマンドライン引数を解析し、numpy を用いてデータをプロットできる形式に変換し、matplotlib を用いて実際にデータをプロットしている。

3.2 cpp/util.*

この SVR のプロジェクトファイル全体で使用されうるファイルである。標準ライブラリのインクルード、parse_args 関数、最頻の標準関数の using、どうしても使いたいマクロ 3 種 (REP,FOR,ALL) の定義を行う。マクロの使用は本来絶対やめるべきだが、個人的コード群であり、REP.FOR,ALL マクロの有用性は非常に高いため、使用することにした。(マクロの使用については、例えば quadProg++ 自体に,`#define solve lu_solve` などが書いてあり、(しかも undef していない)solve という名前が文脈に関係なく一生使えないようになっていたりしたので、私は quadProg++ の作成者に強い憤りを感じた。)

3.2.1 parse_args 関数

この関数はコマンドライン引数のパースを行う。この関数は、SVM を作成する際の main 関数からも参照されうるため、範囲の広いところで定義をした。コマンドライン引数に、所望するキーワードが入っているか、入っていればそのパラメータは何かを解析する。

3.3 cpp/kernel.*

このファイルは、クラス Kernel を定義する。Kernel クラスは、内積 (linear)、ガウスカーネル (gauss)、多項式カーネル (polynomial) のカーネルを表現するクラスである。`enum Kernel::kind` を内包し、それに応じたカーネルを提供する。コンストラクタにて Kernel::kind 及び パラメータ (例えばガウスカーネルなら σ) をとり、カーネルを確定させる。自身のカーネルの文字列型との変換関数 (to_string / strings2kernel_kind) も提供する。また、データ全体に適用する関数群についてもこのクラスの static 関数としており、例えば、ファイルからデータを読み込む read_data 関数、読み込んだデータを 0 から 1 の範囲に正規化する normalize 関数などを提供している。

3.4 cpp/plotable.h

このクラスは、プロットできる関数に対する基底クラスである。プロットできる関数として、ここでは SVM や SVR などが該当する。実際に SVR クラス、SVM クラスはこのクラスを継承する。このクラスは抽象メソッドとして `virtual double func(const vector<double> &x) const` をもつクラスが継承できる。この func メソッドを利用して、指定したファイルに、1 次元又は 2 次元のデータをグリッド状にプロットする plot_data 関数を実装しているため、このクラスを継承している SVR や SVM クラスでは plot_data 関数を利用することができる。

3.5 cpp/main.cpp

このファイルは、メイン関数を提供する。コマンドライン引数の解析を行い、print_usage 関数を使用して使用方法をアラートしたり、実際にデータを正規化したのち SVR に投げたり、交差検定を行ったり、プロットしたりする。

3.6 cpp/svr.*

このファイル群にて SVR クラスを実装している。

3.6.1 SVR クラス

このクラスは、SVR を表現する。コンストラクタにて、サンプルデータ点 x, およびその値 y, パラメータが確定したカーネル、C、eps という SVR を形成に必要十分なデータを取り、実際に solve 関数にて SVR を解く。quadProg++ を用いて SVR を解き、サポートベクターとなっている点から θ を計算し、評価器を得る。サポートベクターの数だけ微小に異なる θ が存在するが、中央値の θ をとることにした。評価器を得た後は、そのインスタンスは func 関数、print_func 関数、test 関数を提供できる。func 関数は、実際の評価器を表現し (内部で関数 kernel_dot_to_w を使用する)、print_func 関数は、その評価機を出力する。test 関数は、その評価器の精度の目安として、その評価器を使用した値と実際の値の差が eps 内にあるサンプル点の割合をプリントする。

3.6.2 SVR クラスの交差検定

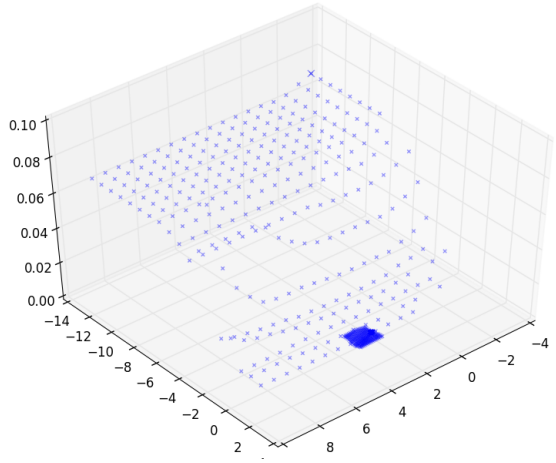
SVR クラスは static 関数として、交差検定を行う cross_validation 関数、及びそれを利用してパラメータや C の値を探索する search_parameter 関数を提供する。cross_validation 関数については SVM の時に実装したものとほとんど同じである。但し、指標として正解数ではなく誤差の総和を持ちる点が異なる。誤差を計算するために、`enum cross_validation_type` を提供していて、平均二乗誤差、平均絶対誤差、決定係数などを用いて calc_diff 関数にて誤差を計算する事ができる。search_parameter 関数についても SVM のときのものとほとんど同じであるが、パラメータに加えて C の値も探索する必要があるため、二次元グリッド上を探索する必要がある点が異なる。計算量が増えるため、SVM ほど詳細に探索できないことが難点である。ちなみに、計算過程においては高速化を図るため、std::thread を用いて並列化して計算している。

4　考察

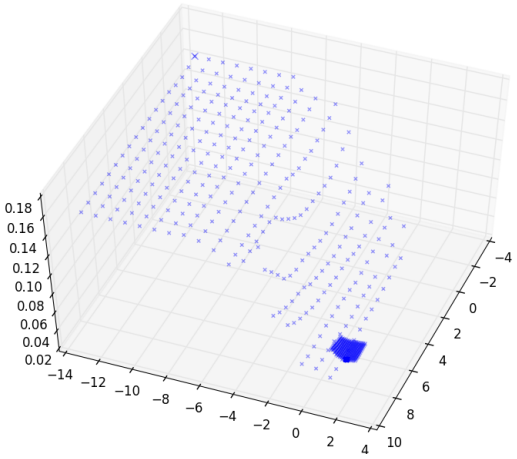
以下では実際に上記のプログラムを用いて、予測精度がカーネル及びパラメータによりどのように変化するかの考察を行う。以下では、特に断りが無い限り eps=0.01 としている。

4.1　探索過程の可視化

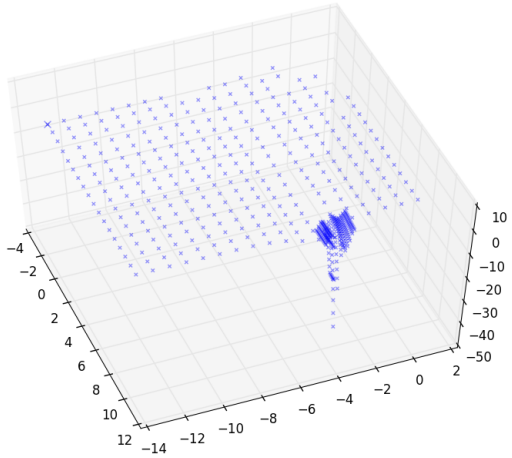
誤差の総和が少ないほどよりよいものとして探索する。最初は荒く、徐々に狭くして探索している。サンプルデータとして、二次元データである sample20.dat を用いた。



(a) 平均二乗誤差 , 探索解 $c = 6.4663, p = 1.72429$



(b) 平均絶対誤差 , 探索解 $c = 186.412, p = 2.36526$

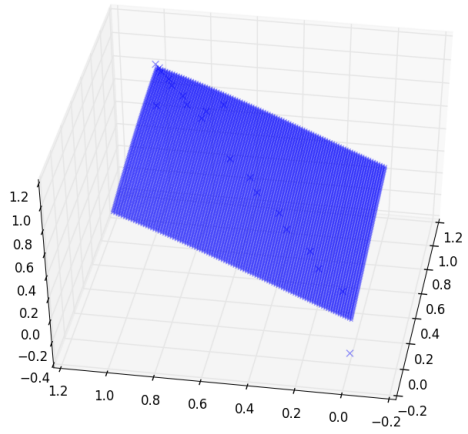


(c) 決定係数 , 探索解 $c = 512, p = 0.1152$

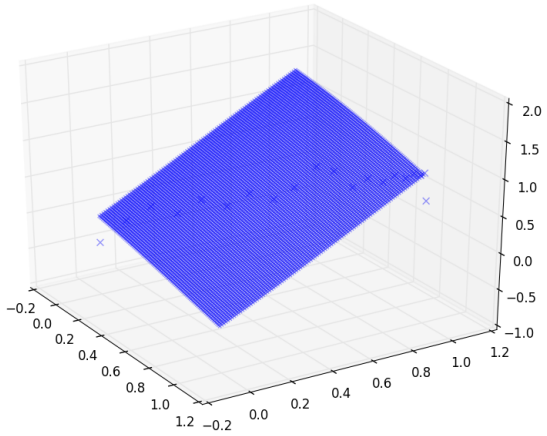
図 1: ガウスカーネルのパラメータの探索について (グリッドの数値 x に対して実際の値は 2 の x 乗を表す)

4.2　探索結果の可視化

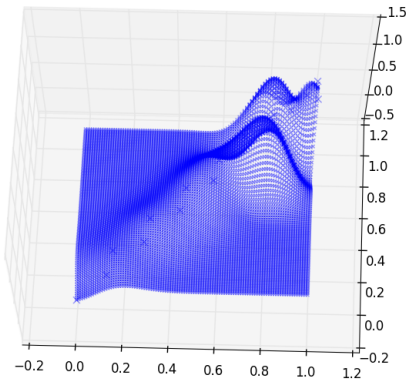
二次元データに対してどのように変化するのか容易に確認できるのは Python の利点である。



(a) 平均二乗誤差の探索解 $c = 6.4663, p = 1.72429$



(b) 平均絶対誤差の探索解 $c = 186.412, p = 2.36526$

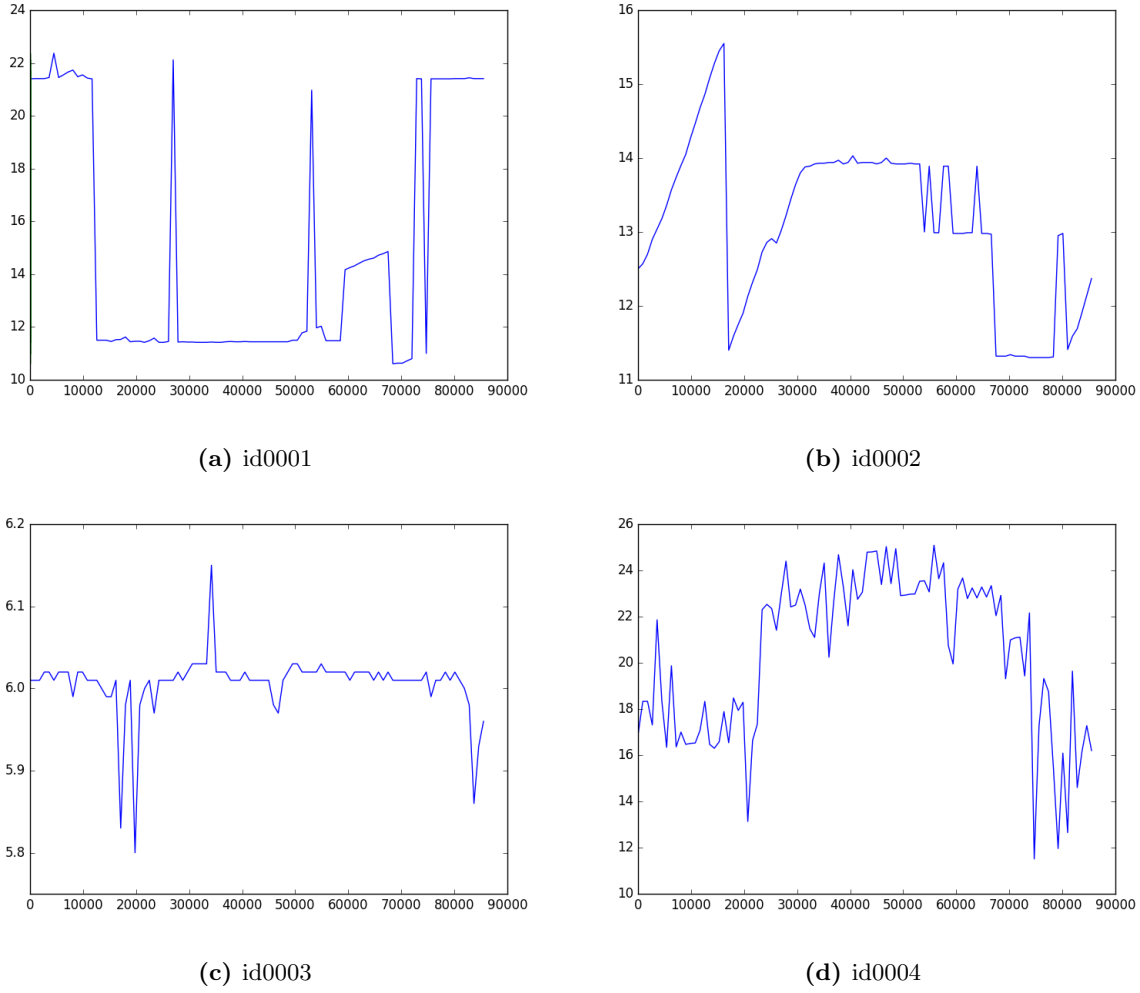


(c) 決定係数の探索解 $c = 512, p = 0.1152$

図 2: ガウスカーネルのパラメータの探索について (グリッドの数値 x に対して実際の値は 2 の x 乗を表す)

4.3　オークションデータの予測について

課題として与えられたオークションデータを用いて、カーネルのパラメータとその予測値の変化を可視化する。実験のページにあるオークションデータは、単純に処理するとサンプルデータ数が 10000 になり、とてもではないが計算が間に合わないため、データを別の形式に変換する必要がある。今回は、日程ごとの差異はなく、時間ごとの差異しかないと仮定して、同じ時間のデータごとにその中央値のデータを代表点として取り出し、サンプルデータを 96 個として予測するものとした。その処理に関しては py3/step3_csv2dat.py にて実装しているのでそれを参照のこと。また、オークションのサンプルデータは 5 つ与えられていたが、今回の考察では id0002.csv を使用することにした。



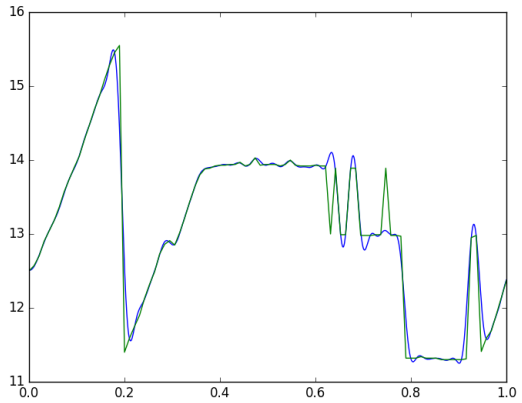
(a) id0001　　**(b)** id0002

(c) id0003　　**(d)** id0004

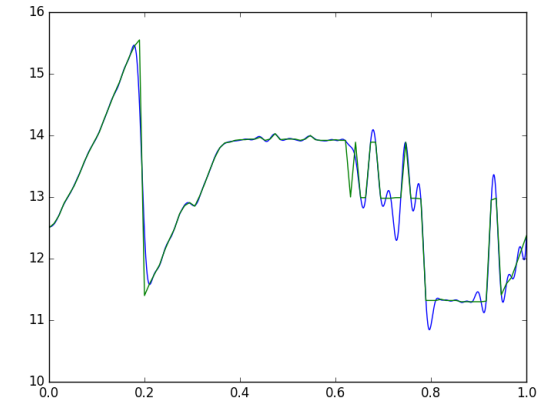
図 3: 与えられたオークションデータ

4.4　id0002 のオークションデータのガウスカーネルでの予測について

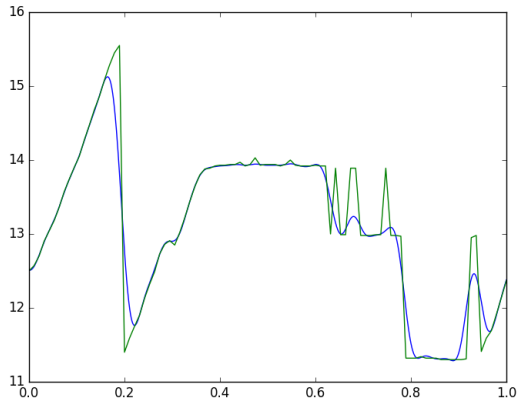
実際にパラメータを変化させてどのように予測精度が変化するかを示す。以下の図で示すように、C が大きいほど教師データに依存し、C が小さいほどモデルは単純になることが分かる。また、ガウスカーネルの場合、σが大きいほど緩やかに近似し、σが小さいほど激しく近似することが分かる。



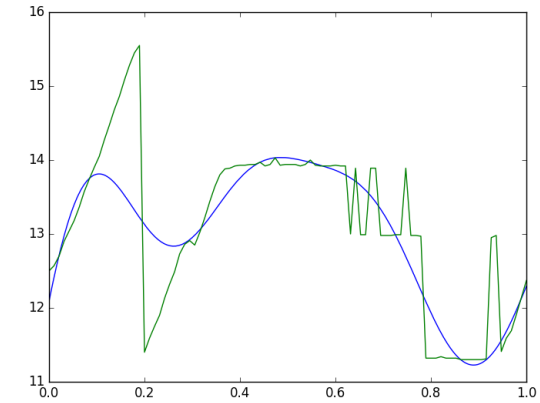
(a) 交差検定による結果 $c = 81.4532, p0.0207861$



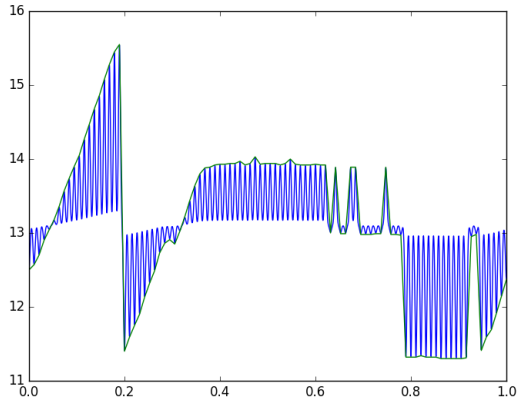
(b) C を 100 倍してみたもの $c = 8145.32, p0.0207861$



(c) C を 0.01 倍してみたもの $c = 0.814532, p0.0207861$



(d) σを 10 倍してみたもの $c = 81.4532, p0.207861$



(e) σを 0.1 倍してみたもの $c = 81.4532, p0.00207861$

図 4: id0002 のガウスカーネルでの変化について