平成 28 年度 3 回生後期実験 (エージェント)
課題 4 入札エージェントの作成

村田 叡

2016/11/18

1　プログラム概要

今回のレポートではオークションで予測値を元に入札するエージェントを作成した。

1.1　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

エージェントは Python3 で実装した。以下、その Python3 の外部インターフェイスについて述べる

2.1　py3/multiagent.py

このコードでは、多エージェントによるオークションシミュレートを実装している。オークションデータの csv ファイル、エージェントの種類を引数にとり、オークションを行う。
エージェントは以下の 4 種類を用意している。
1. 一日目の購入価格の中央値で購入し続ける単純エージェント @simple
2. 一日目の購入価格の中央値の 5 倍の価格で購入し続ける貪欲エージェント @greedy
3. 一つ前の購入価格を次の価格の予測値として利用して戦略的に購入するエージェント @sorena
4. SVR に基づいて次の価格を予測し、戦略的に購入するエージェント @svr

例えば、id0001.csv を用いて、 貪欲エージェントと SVR エージェントを戦わせるには
`python3 py3/multi_agent.py sample_data/id0001.csv @greedy @svr` のようにすればよい。
`--show-process` を引数に加えると、購入過程を逐一表示する。この購入過程は、例えば `0 145.16 13.138` については、左から 購入者 id(0), 購入された時間 (145.16)、落札価格 (13.138) を表す。購入者 id は、csv のデータの人が買えば -1 を, 0 以上であれば、それぞれエージェントを順に表す。例えば、
`python3 py3/multi_agent.py sample_data/id0001.csv @greedy @svr @simple --show-process`
によって実行すれば、@greedy は 0 番、 @svr は 1 番、@simple は 2 版である。SVR の予測値については、ガウスカーネルで、一日目のデータを元に求めた C, σ の値を元にして、最近のデータから毎度 SVR を作成

して予測している (教師データは 100 個)。

3 内部仕様

3.1 py3/visualize.py

このコードでは、データをプロットして可視化するために必要な関数を実装している。CUI インターフェースは無い。3D データを表示する関数、SVR の予測がどのようなものなのかを検証する関数を提供する。

3.2 py3/auction.py

このコードでは、オークションデータを取り扱う Auction クラスを実装している。CUI インターフェースは無い。Auction クラスは、コンストラクタに CSV ファイルをとってデータを読み込み、可視化する関数や、価格の配列を取り出す関数などを提供している。

3.3 py3/multiagent.py

このコードでは、内部的には購入に携わる Buyer クラス及び購入戦略に基づいて購入する Agent クラスを実装している。

3.3.1 Buyer クラス

最初に所持金額を与え、buy メソッドにより購入を行うことができるような抽象化をしている。

3.3.2 Agent クラス

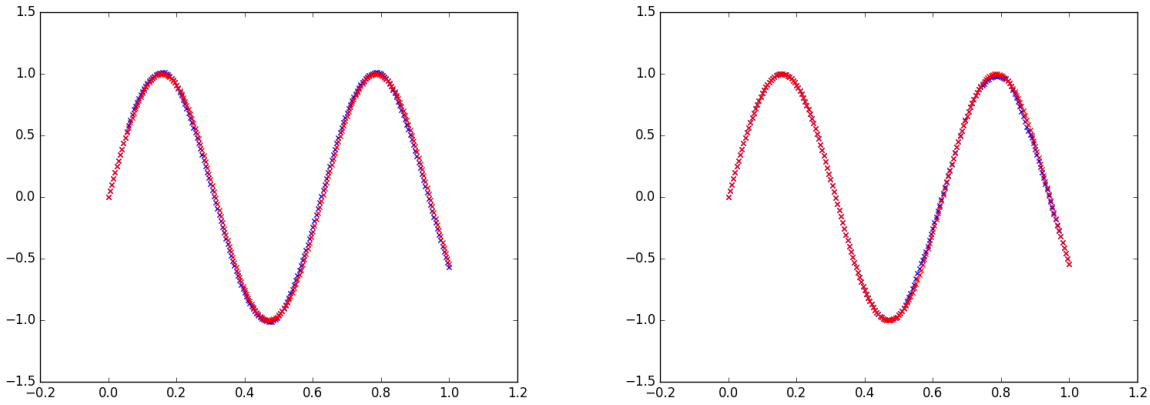
一日目のデータ、一日目の購入価格の中央値、Buyer クラスインスタンスを元にして作成する。buy_から始まる関数では、各エージェントがその関数が表す戦略に基づいた購入価格を出力する。do_multi_auction 関数によって、実際に多エージェントによるオークションをシュミレートする。

4 考察

4.1 SVR の予測精度について

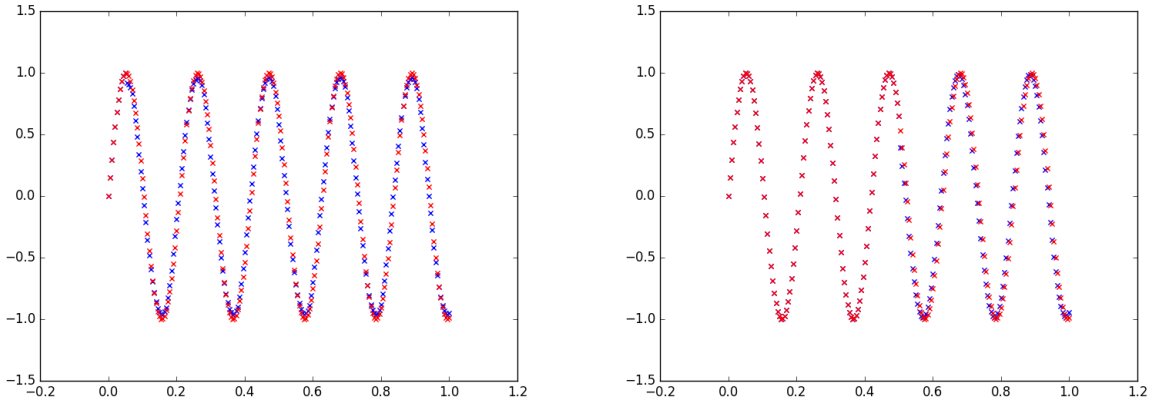
SVR 自体の能力を過信せず、一度検証してみることが大事である。予測精度を可視化するために、ある関数 $f(x)$ を定義し ($x \in [0,1]$)、与えるデータの範囲を $[0,b](0 < b < 1)$ とすることで予測がどのように変化するかを考察する。なお、この考察において教師データ数は b * 200 である。

4.1.1 sin 波



(a) $b = 0.05$ **(b)** $b = 0.5$

図 **1:** $f(x) = sin(10x)$ による予測値 (赤:実値, 青:予測値)

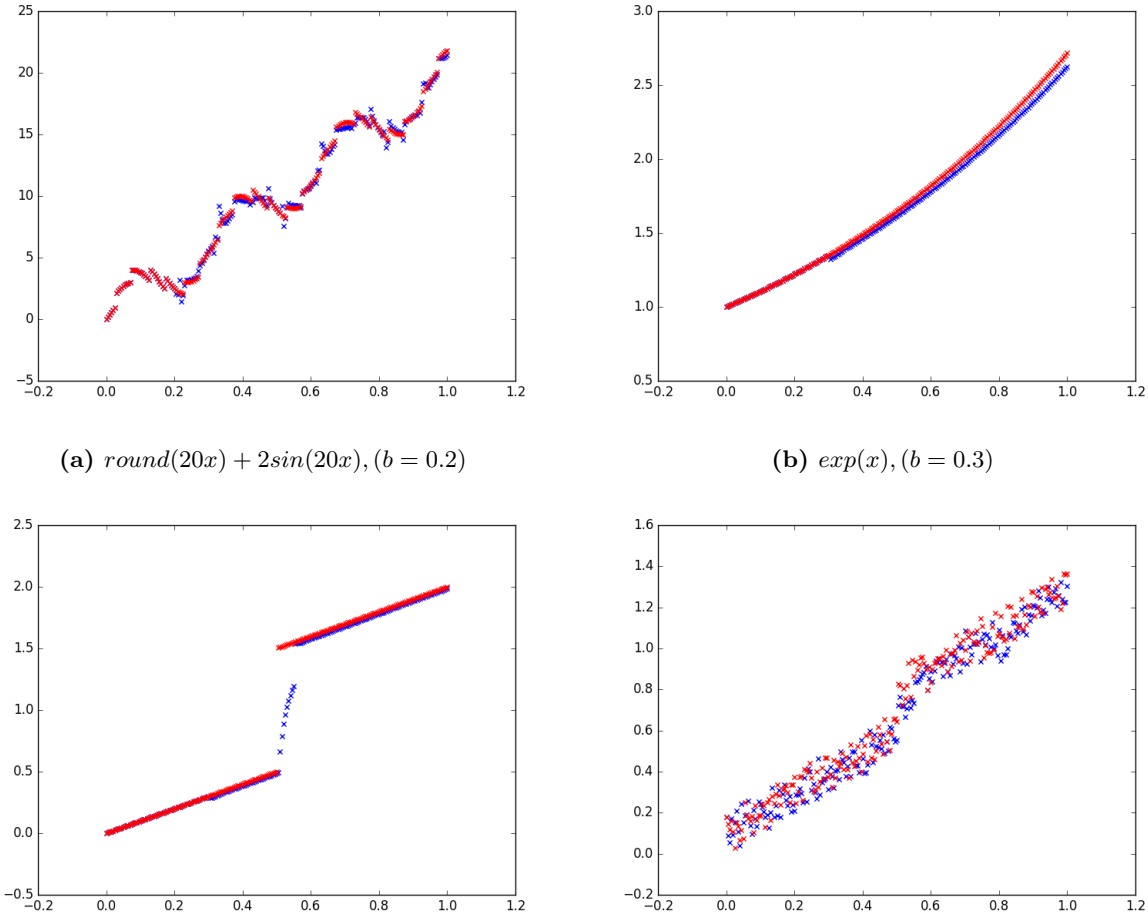


(a) $b = 0.05$ **(b)** $b = 0.5$

図 **2:** $f(x) = sin(30x)$ による予測値 (赤:実値, 青:予測値)

sin 波は、規則性が強く、とても少ないサンプル数でもほとんど正しい予測ができることがわかる。

4.1.2 その他の 100 件 ($b = 0.5$) 以下の教師データによる関数の予測



(a) $round(20x) + 2sin(20x), (b = 0.2)$

(b) $exp(x), (b = 0.3)$

(c) $x + round(x), (b = 0.3)$

(d) $x + 0.2 * (round(x) + random), (b = 0.3)$

図 3: 赤:実値, 青:予測値 (random は、0 1 の一様乱数を、round は四捨五入を表す関数)

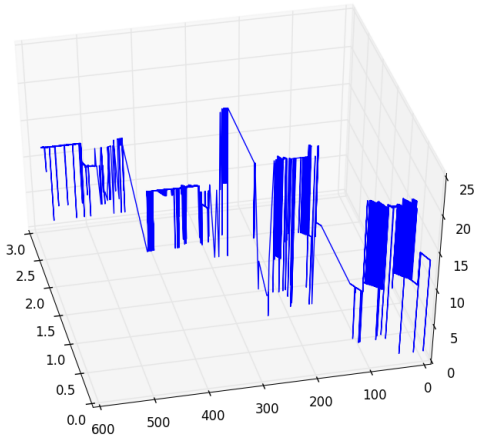
ある程度の規則性があれば、ほとんど正しい予測ができていることが分かる。特に図 (c) のように、急激に変化した場合も、ある程度は急激に変化した値を外れ値とみなして急な追随をしていないところが、SVR の予測精度の高さをよく示している。図 (d) のように、ノイズの乗ったものに関しても、ノイズの範囲で正しく予測できている事がわかる。

4.2　オークションについて

以下、課題にあるオークションについて考察する。

4.3　オークションのデータについて

まず、時系列データの妥当性について考える。例えば、毎日のデータについて朝 7 時は安い、などのように、時刻に関連して値段が変わりやすい場合、日にちに関する情報を与えたほうがよくなる。しかし、以下の図のように、同時刻でも値段は全く異なり、日にちには依存がないように見えるため、時系列データとして扱う方法は妥当であるといえる。



(a) 奥行きが日程を表す

4.4　エージェントについて

エージェントは前述の通り以下の 4 種類を用意している。

1. 一日目の購入価格の中央値で購入し続ける単純エージェント @simple
2. 一日目の購入価格の中央値の 5 倍の価格で購入し続ける貪欲エージェント @greedy
3. 一つ前の購入価格を次の価格の予測値として利用して戦略的に購入するエージェント @sorena
4. SVR に基づいて次の価格を予測し、戦略的に購入するエージェント @svr

特に、@sorena は @svr と比べて、予測精度の高さによって購入数がどう変化するかを表すのによく適している。

4.5　課題 4-1, 落札額データに従う入札者との勝負

まず、予測値を元に行う戦略について考える。予測精度が高ければ高いほど購入が最適になる戦略が望ましい。購入の期待値 exp を 残金 ÷ 本日の残り出品回数 と定義する。exp どおりに買うことができれば、最大限大きな金額を提示しながらぴったりオークションの回数で購入することができる。更に、この値を参考にすることで、残りオークション回数が少ないのに残金が余っていれば、多少高くても買うことができる。実際にはこの通りに購入しようとしても、全て購入することは難しいので、以下の戦略を取る。すなわち、予測値が exp 以下だと、 exp で購入することを宣言し、予測値が $exp * 3$ 以下なら 予測値の 1.1 倍、$exp * 3$ 以上だと、それ以上はお金を出せないので $exp * 3$ を宣言する というものである。なお、この 3 はヒューリスティックであるが、この値は数倍程度でははあまり性能に寄与しないことを追記しておく。

予測値の 1.1 倍を使う部分について、予測値を使わずに exp の定数倍を宣言する方法もあるが、その方法の場合、 安く買える時に安く購入することができなくなる割合が増えるため、結果的に損をしてしまう。よって予測値を使うことにしている。また、予測値どおりだと、購入する確率は半分になりお金を余らせがちなので、1.1 倍している。

4.5.1 購入結果

初期所持金額は 10000 としている。課題では 5 種類の商品時系列データが与えられているが、id0003,id0004 に関してはオークション回数が少ないため省略する。

表 **1:** id0001 理論上最大で 1457 中 628 個買うことができる
(括弧内は参考として残金を最大価格 22.93 で使い切った場合の購入数を示す)

エージェント名	購入数	残金	最大購入値	最小購入値
@simple	308(607)	6860.49	12.48	0.05
@greedy	543	1.32	22.93	0.05
@sorena	482(548)	1519.09	21.97	0.05
@svr	620(627)	173.14	21.73	0.05

表 **2:** id0002 理論上最大で 1709 中 820 個買うことができる
(括弧内は参考として残金を最大価格 25.0 で使い切った場合の購入数を示す)

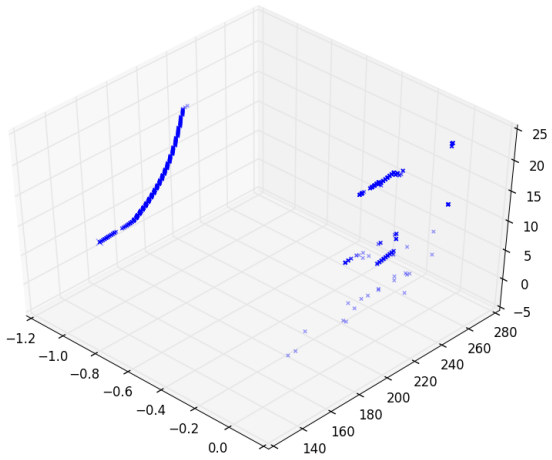
エージェント名	購入数	残金	最大購入値	最小購入値
@simple	541(696)	3886.92	11.39	0.05
@greedy	543	1.32	22.93	0.05
@sorena	722(776)	1373.71	13.9	11.26
@svr	772	6.20	14.06	11.26

表 **3:** id0005 理論上最大で 2193 中 554 個買うことができる
(括弧内は参考として残金を最大価格 25.0 で使い切った場合の購入数を示す)

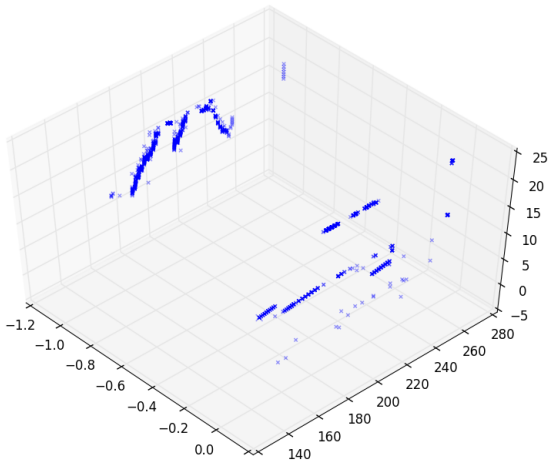
エージェント名	購入数	残金	最大購入値	最小購入値
@simple	539	2.12	22.06	0.05
@greedy	436	0.58	25.0	0.05
@sorena	506(522)	418.33	20.37	0.05
@svr	545	0.80	20.54	0.05

id0002 では、@svr は () 内の値では@sorena に負けている。理由としては、突然現れる最小購入値を見逃してしまったことが考えられる。そのほかのものに関しては@svr が一番購入できている。@simple はしきい値以下のものしか購入しないため、しきい値以下のものが全然出ないパターンに関しては非常に弱くなる。@greedy はお金が尽きるまで購入し続けるため、はじめに高いものが重なるパターンに関しては非常に弱くなる。@sorena はある程度の予測はできるが、常に一歩遅れた値を予測するため、買い逃す機会が増えてしまう。

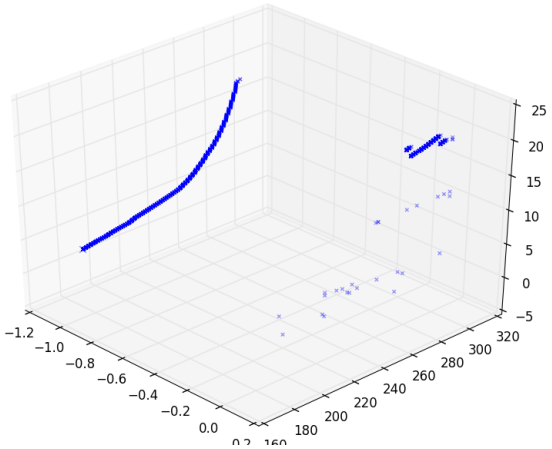
4.5.2 購入過程



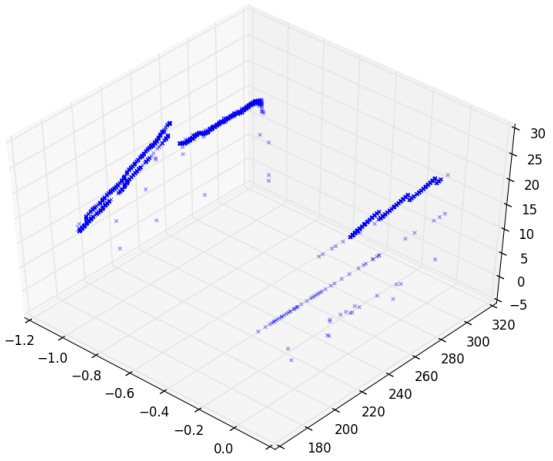
(a) @sorena の id0001 購入過程



(b) @svr の id0001 購入過程



(c) @sorena の id0005 購入過程



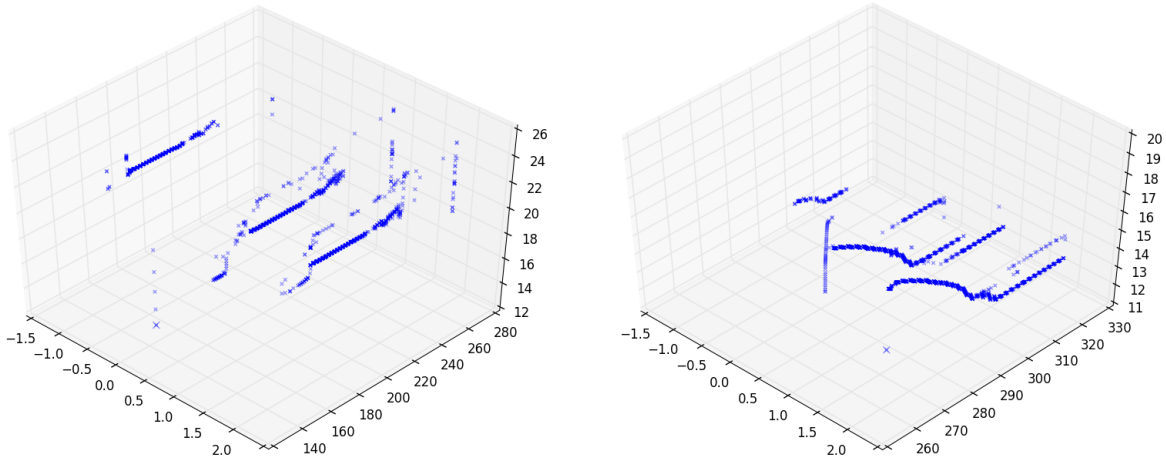
(d) @svr の id0005 購入過程

図 5: 0 25 の軸は購入金額を、0.0 -1.0 の軸は、手前 0.0 の列がエージェントを、奥-1.0 の列が csv データによる購入者を表し、140 280 の軸は、購入時刻を表す。

@simple,@greedy に関しては、しきい値以下のもののみ買う、はじめから全て買うなどわかりきっているので省略する。@sorena の購入過程については、初めに安く設定しすぎてデータによる購入者がとても安い価格で落札してしまい、更にその落札額を予測値として利用するために、はじめはほとんど買えないことが分かる。一方、SVR では、前半でも安いものをちょくちょくいい感じに購入し、後半は余らさないように購入頻度を増やしていることが見て取れる。

4.6　課題 4-2, 多人数の勝負

他人の広告掲示価値は非常にわかりにくいため、4-1 のモデルをそのまま適応したものについて考察する。すなわち、データに基づく購入者のほかに、 @simple,@sorena,@svr の 3 名がいる場合の動向について考察する。@sorena と @svr を組み合わせた場合、両者が依存しあい、結果的にほぼおなじ購入曲線を描くことが



(a) id0001 購入過程　　**(b)** id0002 購入過程

図 6: 奥から順に データによる購入者,@simple,@sorena,@svr を表す

分かる。また、 @simple では、購入金額が一日目と大きく変わり、ほとんど購入できていないことが分かる。