dynpricepy Documentation

Release

Author

Dec 28, 2017

Contents:

1 dynpricepy package **1**
- 1.1 Submodules . . . 1
- 1.2 dynpricepy.agent module . . . 1
- 1.3 dynpricepy.price_path module . . . 2
- 1.4 dynpricepy.pricing_step module . . . 4
- 1.5 Module contents . . . 5

2 Indices and tables **7**

Python Module Index **9**

CHAPTER 1

dynpricepy package

1.1 Submodules

1.2 dynpricepy.agent module

class dynpricepy.agent.**AgentBehaviour**(*probability_model*, ***kwargs*)

Bases: object

object which contains the modelisation of the market we want to price Future : maybe the modelisation of the behaviour of each individual composing the market

```
>>> def probability_model(t, price, date):
...     return np.exp(-price)
>>> agent_behaviour = AgentBehaviour(probability_model=probability_model)
>>> agent_behaviour.compute_selling_probability(t=2, price=np.array(1,2),␣
↪date=None)
```

Parameters

- **probability_model** (*python callable object*) – a python callable object with at least t, price and date as argument, which should always return results between 0 and 1
- **kwargs** (*function parameters*) – additional parameters directly passed to the probability_model:

compute_selling_probability(*t*, *price*, *date=None*)

Compute the probability to sell an item at time t, price price, date date, and other parameters if specified during class init Not used by user

Parameters

- **t** (*int, > 0*) – the step from the begining of the price path
- **price** (*float or numpy array*) – the price we want to estimate the probability to book

- **date** (*datetime or date*) – the date of the price step

Returns probability to sell one good at price price, computed from the model probability_model passed in init

Return type float or numpy array, with element between 0 and 1

1.3 dynpricepy.price_path module

class dynpricepy.price_path.**PricePath**(*agent_behavior*, *min_price_start=0*, *max_price_start=1*, *max_price_search=0.3*, *min_price_search=0.3*, *price_step=0.01*)

Bases: object

A pricing path is a sequence of prices to apply at each time step. The optimal pricing path is optimal in the following sense: We have n days to sell an item, we know the demand price elasticity $\mathbb{P}[\text{sell}; t, p_t]$, so $p_0, ..., p_n$ are defined such that

$$p_0, ..., p_n = argmax_{p_0,...,p_n} \sum_{t=0}^{n} p_t \mathbb{P}[\text{sell}; t, p_t] \prod_{i>t} (1 - \mathbb{P}[\text{sell}; i, p_i])$$

for an inventory of size 1, or more generally,

...

The order of the prices step are always ordered descending along the date : t=0 correspond the perish date of the items we are selling.

Example : we want to sell an item which perish on 2018-01-20, starting on 2018-01-01. Thus we have to price for 2018-01-01, 2018-01-02, ..., 2018-01-20. The price path is therefore a list where for each date, t is the time remaining : 20, 19, ..., 0, and the price is the optimal price to apply. The index 0 correspond to the last day available to sell

Dev information : - the list is ordered is reverse order : (t=0, t=1, t=2, ..., t=20)

Parameters

- **min_price_start** (*float*) – the lower bound for the price for t=0
- **max_price_start** (*float*) – the upper bound for price for t=0
- **min_price_search** (*float*) – the maximum difference between the previous optimal price and the new searched
- **max_price_search** (*float*) – the maximum difference between the previous optimal price and the new searched
- **price_step** (*float*) – the discretisation step for price grid search
- **agent_behavior** (AgentBehaviour) – AgentBehaviour instance, which provide demand and offer or directly probability

compute_price_path(*t*, *start_date=datetime.date(2017, 12, 28)*, *date_step=datetime.timedelta(1)*, *n=1*)

Compute and store the optimal price path as defined before. This method handles multiple items, whereas the price is independant of the remaining items. Can be deprecated later if not useful.

$$V(i, j) = P(i, j) =$$

Parameters

- **t** (*int*) – number of step to compute
- **start_date** (*date or datetime*) – the starting date (default today)
- **date_step** (*timedelta*) – the timedelta between two step (default one day)
- **n** (*int*) – number of items we have to sell (default 1)

Raises **ValueError** – if the number of step provided is < 0

compute_price_path_one(*t*, *start_date=datetime.date(2017, 12, 28)*, *date_step=datetime.timedelta(1)*)

Compute and store the optimal price path as defined before. This method handles only 1 item to sell. The price path is computed as followed :

$$p_0 = argmax_p p\mathbb{P}[\text{sell}; p, 0] + \text{future income}(1 - \mathbb{P}[\text{sell}; p, 0])$$
$$\text{future income} = p_0\mathbb{P}[\text{sell}; p_0, 0] + \text{future income}(1 - \mathbb{P}[\text{sell}; p_0, 0])$$
$$\forall i > 0, p_i = argmax_p p\mathbb{P}[\text{sell}; p, 0] + \text{future income}(1 - \mathbb{P}[\text{sell}; p, 0])$$
$$\forall i > 0, \text{future income} = p_i\mathbb{P}[\text{sell}; p_i, i] + \text{future income}(1 - \mathbb{P}[\text{sell}; p_i, i])$$

Parameters

- **t** (*int*) – number of step to compute
- **start_date** (*date or datetime*) – the starting date (default today)
- **date_step** (*timedelta*) – the timedelta between two step (default one day)

Raises **ValueError** – if the number of step provided is < 0

```
>>> price_path.compute_price_path_one(t=4)
>>> print(price_path)
Time 0, date 2017-12-27, price 0.990, selling probability 0.372, sold rate 0.
↪696, sell today probability 0.180
Time 1, date 2017-12-28, price 1.290, selling probability 0.275, sold rate 0.
↪517, sell today probability 0.184
Time 2, date 2017-12-29, price 1.590, selling probability 0.204, sold rate 0.
↪333, sell today probability 0.171
Time 3, date 2017-12-30, price 1.820, selling probability 0.162, sold rate 0.
↪162, sell today probability 0.162
```

compute_sell_today_probability()

Compute and store the probability to effectively selling the item exactly at time t for each time t in each price step instance

$$\mathbb{P}[\text{sell at t; not sell before t}]$$

compute_sold_rate()

Compute and store the probability to have sold the item at time t in each price step instance

$$\mathbb{P}[\text{sell at t or t+1 or ... or n}]$$

expected_revenue()

The expected revenue of selling one item when following the optimal price price computed:

$$\mathbb{E}[\text{revenue}] = \sum_{t=0}^{n} p_t\mathbb{P}[\text{sell}; p_t, t]$$

Returns the expected revenue

Return type float

Raises **ValueError** – when the price path has not been computed yet

to_dataframe()

The dataframe representation of the price path.

Returns the dataframe representation of the price path

Return type pandas dataframe

Raises **ValueError** – if the price path has not been computed yet

1.4 dynpricepy.pricing_step module

class dynpricepy.pricing_step.**PricingStep**(*t*, *price*, *selling_probability*, *date=None*)

Bases: object

Object representing at a specific time the price to apply, and some related information about the probability to sell an item

```
>>> step = PricingStep(t=0, price=0.4, selling_probability=0.1, date=datetime.
↪date.today())
```

Parameters

- **t** (*int, > 0*) – represents the place of the pricing step in the whole price path, where t=0 represents the perish date of the item
- **price** (*float*) – the optimal price to set in order to satifsy the dynamic pricing problem
- **selling_probability** (*float, between 0 and 1*) – the probability to sell an item at price price, provided by the agent model
- **date** (*datetime or date*) – the associated date to the price step (None if not provided)

Returns PricingStep instance

Return type *PricingStep*

Raises

- **ValueError** – if the provided probability is not between 0 and 1
- **ValueError** – if t < 0

expected_revenue()

The expected income from the date associated to the pricing step. Computed by:

$$p_t P[\text{sell at t; not sold at } t' > t]$$

```
>>> step.expected_revenue()
```

Returns the expected income

Return type float

1.5 Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

dynpricepy, 5
dynpricepy.agent, 1
dynpricepy.price_path, 2
dynpricepy.pricing_step, 4

Index

A

AgentBehaviour (class in dynpricepy.agent), 1

C

compute_price_path() (dynpricepy.price_path.PricePath method), 2
compute_price_path_one() (dynpricepy.price_path.PricePath method), 3
compute_sell_today_probability() (dynpricepy.price_path.PricePath method), 3
compute_selling_probability() (dynpricepy.agent.AgentBehaviour method), 1
compute_sold_rate() (dynpricepy.price_path.PricePath method), 3

D

dynpricepy (module), 5
dynpricepy.agent (module), 1
dynpricepy.price_path (module), 2
dynpricepy.pricing_step (module), 4

E

expected_revenue() (dynpricepy.price_path.PricePath method), 3
expected_revenue() (dynpricepy.pricing_step.PricingStep method), 4

P

PricePath (class in dynpricepy.price_path), 2
PricingStep (class in dynpricepy.pricing_step), 4

T

to_dataframe() (dynpricepy.price_path.PricePath method), 4