

Přehled nejdůležitějších metod a stručné informace k použití

Generování struktury:

-Prováděno pomocí instance třídy Generator.

-Využití metody generateSeq, případně generateFunc podle toho, v jakém API se pohybujeme.

`g = Generator()` – vytvoření instance generátoru

`g.generateSeq(x, model, string)`: x je rozsah generované struktury a model je náš model, do kterého vrstvy přidáváme; string je záznam slovníku s pravidly a definuje, která vrstva z něj má být první (v případě neuvedení se použije záznam z `globalSettings.firstLayerName`)

`g.generateFunc(x, model, string)` – obdoba pro funkcionální API

-metoda vrací model s přidanými vrstvami

Označování parametrů:

-Přímo v místech, kde má být daný parametr dosazen.

-Označení prováděno pomocí metod instance třídy Parameter - například `p.Choice([1,2,3], "rozdělení")`

- rozdělení musí být klíč slovníku metody `getDistributions` třídy `globalSettings`

Možné parametrizování:

-"rozdělení" a zaokrouhlení "a" není nutno definovat, v takovém případě se použije defaultní z třídy `globalSettings`

- `Choice(list, rozdělení)` : vrací prvek z listu
- `NumRange(x, y, rozdělení, a)`: vrací číslo z rozsahu x-y
- `StepRange(x, y, z, rozdělení, a)`: vrací číslo z rozsahu x-y se zadaným krokem zadaným
- `shuffle(index)`: vrací prvek z předem definovaných listů - před využitím metody je nutno tyto listy definovat použitím metody `setShuffleList([[list1], [list2]...])`
- `overAll(identifier, list, startIndex)`: vrací vždy (n+1)-tý prvek od posledního volání - při prvním volání si uloží záznam s klíčem „identifier“. Je možno specifikovat počáteční index listu, od kterého budeme vracet

Spuštění:

-Před spuštěním je nutno modifikovat třídu settings a upravit ji dle potřeb (tvorba modelu v metodě `getModel`, data atd.)

-V třídě settings nástroj očekává definování trénovacích a testovacích dat (`x_train`, `y_train`, `x_test`, `y_test`) - pokud nechceme data dělit sami, můžeme zde ponechat pouze proměnné s názvem "x" a "y" a nástroj si následně tato data rozdělí na množinu trénovacích a testovacích dat.

-Vytvoření instance daného optimizeru:

- `r = RandomOptimizer()`
- `g = GeneticOptimizer()`, nebo `g = GeneticOptimizer(mR = n, cR = n, kR = n, pS = n)` pokud chceme specifikovat některý parametr

-Následné spuštění pomocí:

`instanceMetody.optimize(x, y, z, str)`: počet cyklů, počet modelů, index metriky metody evaluate, „high“ nebo „low“ určuje cíl metriky (maximální/minimální hodnota)

Přidání nového rozdělení:

-Vytvoření nové třídy implementující metody podle třídy distribution.

-Přidání záznamu do metody `getDistributions` třídy `globalSettings`.

Přidání nové optimalizační metody:

-Vytvoření nové třídy implementující metodu `optimize` třídy `optimizer`. Metoda obsahuje samotnou logiku za novou optimalizační metodou.

Úpravy třídy globalSettings:

-Tyto proměnné a metodu není nutné modifikovat, pokud nechceme upravit hodnotu specifické proměnné.

-Zde obsažené proměnné slouží jako defaultní hodnoty, které jsou využity, pokud specifickou část nenastavíme při samotném používání.

Na závěrem tohoto stručného shrnutí bych ještě dodal, že v době psaní tohoto dokumentu je celý kód součástí jednoho jupyter notebooku. Do budoucna by mohlo být vhodné jej převést do podoby importovatelného package.