Machine Teacher – Documentação

Documentação para a disciplina
"INF2102 – Projeto Final de Programação"

Pedro Lazéra Cardoso

Junho, 2020

1 Especificação do Programa

O programa é um pacote em Python que implementa o paradigma de *Machine Learning* do meu projeto de pesquisa. No projeto, estamos estudando *Machine Teaching*, em que procuramos entender como um *Teacher* pode ensinar um conceito-alvo a um *Learner*.

A finalidade do programa é possibilitar e padronizar (*i*) a interação entre *Teacher* e *Learner*, (*ii*) a criação de *Teachers*, (*iii*) a criação de *Learners* e (*iv*) a criação de relatórios que demonstrem os resultados dessa interação para um determinado conjunto de treino.

1.1 Descrição de um experimento

Esta seção descreve um experimento, realizado rodando a função *protocol*. Este é o principal caso de uso do projeto. Para a descrição, usaremos a seguinte notação:

- T: um *teacher*, uma estratégia de ensino
- L: um *learner*, com modelo de aprendizado já definido. Por exemplo, árvore de decisão com todos os atributos definidos.
- X: o dataset. Deve ser uma matriz, com cada linha correspondendo a atributos de um exemplo da população.
- y: vetor de classes. A i-ésima entrada de y é a classe da i-ésima linha de X. É o gabarito.
- $time\_limit$: número real positivo indicando o tempo limite que T e L têm para interagir.
- h: vetor de classes. A i-ésima entrada de y é a classe atribuída à i-ésima linha de X por um *learner* L. É uma hipótese sobre quais são as classes corretas.
- TS: *teaching set*. É um subconjunto de (X, y). Este subconjunto é selecionado por T e usado por L para treino.
- X_{test}, y_{test}: análogos de (X, y) para um conjunto de teste. T e L não têm acesso a (X_{test}, y_{test}) durante o aprendizado.

Cada experimento tem como *inputs* $(T, L, X, y, X_{test}, y_{test}, time\_limit)$ e como outputs estatísticas da hipótese h de L sobre X e da hipótese h_{test} de L sobre X_{test}. Exemplos de estatísticas são a acurácia e a distribuição de classes das hipóteses h e h_{test}.

Um experimento consiste numa série de interações entre T e L.

Na primeira iteração, T envia um TS a L sem conhecer nada sobre L (sem conhecer como L classifica qualquer exemplo de X).

Nas iterações seguintes, T interage de duas formas com L.

- Primeiro, enquanto T quiser, T envia conjuntos de exemplos para L classificar. T tem conhecimento de como L classifica estes exemplos. Isto pode durar várias iterações de classificação.
- Depois, T envia a L um conjunto de exemplos TS para L treinar.

1.2 Casos de uso

Esta seção descreve alguns casos de uso do programa.

Caso 1

Objetivo: criar um Teacher que pode ser usado num experimento

Requisitos: escrever uma classe que herda da classe Teacher. Esta classe deve sobrescrever os métodos *start*, $get\_first\_examples$ e $get\_new\_examples$.

Atores: usuário

Fluxo principal: usuário escreve a classe. Um script python cria um objeto dessa classe, que pode ser usado em experimentos.

Fluxo alternativo: usuário não sobrescreve os métodos *start*, $get\_first\_examples$ e $get\_new\_examples$.

Caso 2

Objetivo: criar um Learner que pode ser usado num experimento

Requisitos: escrever uma classe que herda da classe Learner. Esta classe deve sobrescrever os métodos fit e *predict*

Atores: usuário

Fluxo principal: usuário escreve a classe. Um script python cria um objeto dessa classe, que pode ser usado em experimentos.

Fluxo alternativo: usuário não sobrescreve os métodos fit e *predict*

Caso 3

Objetivo: rodar um experimento com Learner, um Teacher e um Dataset

Requisitos: instanciar objetos que herdam das classes Learner, Teacher (um objeto para cada caso). Carregar um dataset em memória.

Atores: usuário

Fluxo principal: usuário instancia objetos que herdam das classes Learner e Teacher; carrega em memória um dataset com atributos X e classes y; chama a função protocol passando como parâmetros os objetos instanciados e as configurações do experimento

Fluxo alternativo: —

Caso 4

Objetivo: rodar um experimento com Learner, um Teacher e um Dataset a partir de um arquivo de configuração

Requisitos: escrever um arquivo de configuração. Rodar um script que executa a função XPTO passando como argumento o caminho do arquivo de configuração Fluxo principal: usuário escreve arquivo de configuração em disco, configurando quatro conjuntos de parâmetros – Teacher, Learner, Dataset, Protocol (este último opcional).

Fluxo alternativo: usuário escreve arquivo de configurações com erros de configuração. O programa acusa erro. Os cenários são muito variados, pois há possibilidade de erros em qualquer dos quatro conjuntos de parâmetros.

Caso 5

Objetivo: rodar um experimento com Learner, um Teacher, um Dataset para treino e um Dataset para testes.

Requisitos: instanciar objetos que herdam das classes Learner, Teacher (um objeto para cada caso). Carregar os datasets de treino e de teste na memória.

Fluxo principal: usuário instancia objetos que herdam das classes Learner e Teacher; carrega em memória um dataset com atributos X e classes y; chama a função protocol passando como parâmetros os objetos instanciados e as configurações do experimento

Fluxo alternativo: usuário escreve arquivo de configurações com erros de configuração. O programa acusa erro. Os cenários são muito variados, pois há possibilidade de erros em qualquer dos quatro conjuntos de parâmetros.

2 Projeto do Programa

2.1 Diagrama de classes

A figura abaixo mostra um diagrama de classes simplificado do projeto.

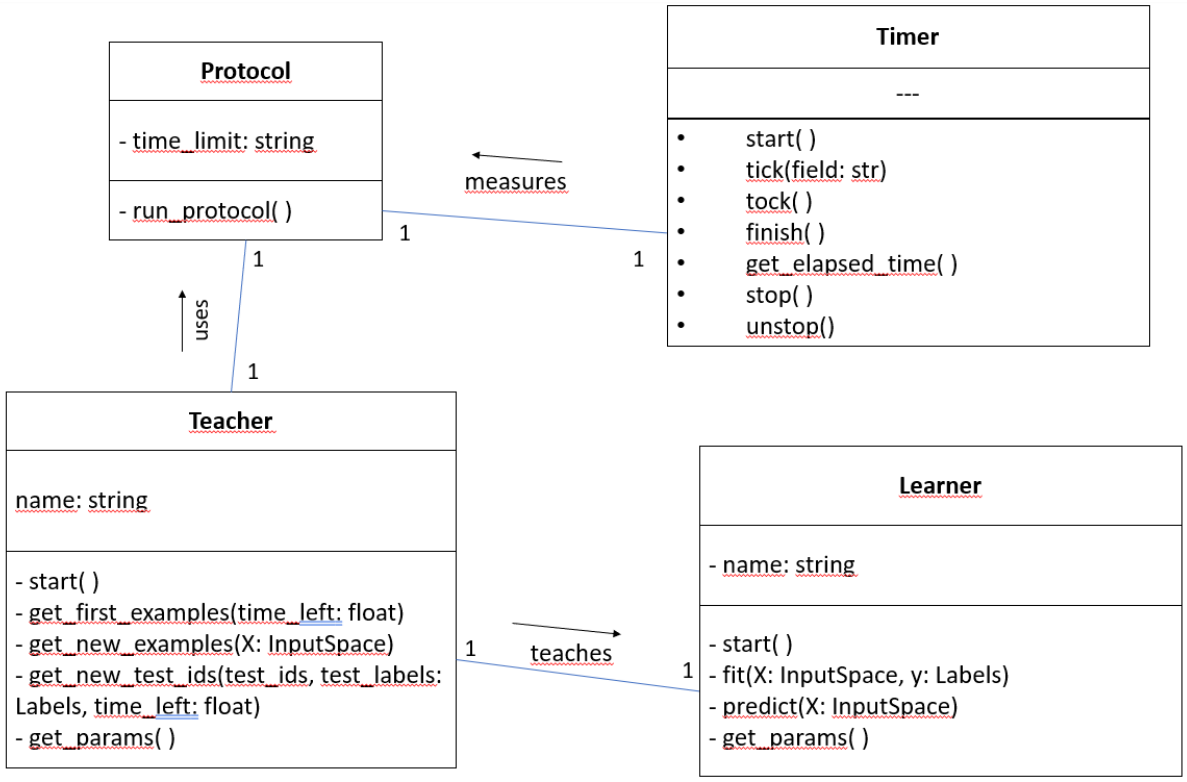

Figure 1: Diagrama de classes

3 Código-fonte cuidadosamente comentado

O código-fonte do programa se encontra no endereço `https://github.com/pedrolazera/MachineTeacher`.

4 Roteiro de Teste Efetuado

Os testes unitários foram efetuados com o pacote *unittest* da linguagem *Python*. Foram criados três módulos (arquivos) de teste: $test_double_teacher$,

$terst_teachers$ e $test_timer$. Estes módulos estão documentados e contém mais detalhes sobre os testes unitários realizados. O projeto passa em todos os testes realizados.

A seguir descrevo todos os testes feitos. O texto está em inglês, pois é constituído de trechos de comentários do código-fonte.

4.1 Módulo *test_teachers*

Este módulo testa o *teacher* $PacTeacher$, *teacher* $WTFTeacher$ — usados num projeto anterior — e também a criação de *teachers* pelo usuário.

Testes do PacTeacher

- Checks 'PachTeacher' initial conditions
- Checks 'PacTeacher' first set of training examples
- Checks 'PacTeacher' full set of training examples for a the car dataset and the RandomForest Learner

Testes do WTFTeacherTest

- Checks 'WTFTeacherTest' initial conditions
- Checks 'WTFTeacherTest' first set of training examples
- Checks 'WTFTeacher' full set of training examples for a the car dataset and the RandomForest Learner

Testes de CustomTeacher

- Checks the protocol behaviour when the teacher/learner intereactins run out of time

4.2 Módulo *test_timer*

- Test expected erros before the timewatch starts
- Checks impossible actions during tick (a field is active)
- Checks impossible actions while the wach is stopped
- Checks the watch accuracy
- Checks the watch accuracy with time stoppages
- Checks the watch accuracy measuring total time elapsed
- Checks impossible actions while the wach is finished
- Checks making a copy of the watch

4.3 Módulo *test_double_teacher*

O *test_double_teacher* é um *teacher* usado como base de outros *teachers*, então ganhou um módulo de teste exclusivo.

- Checks first examples set distribution
- Checks if the DoubleTeacher trains with every possible example when time permits. Teaching set sizes: 10, 20, 40, 80, 100

5 Documentação para o Usuário

É possível encontrar mais instruções de uso e documentação em `https://github.com/pedrolazera/MachineTeacher`.

5.1 Instalando o pacote

Ainda não é possível baixar o pacote pelo pip ou instalar o pacote. Em vez disso, faça o download do projeto e coloque a pasta raiz em algum diretório. Vou assumir que o local é */caminho/MachineTeacher*.

5.2 Dependências

As dependências do pacote estão listas em */MachineTeacher/machine_teacher/requirements.txt*. Para instalá-las, basta abrir um terminal e rodar o comando:

```
$ pip3 install -r caminho/MachineTeacher/machine_teacher/requirements.txt
```

5.3 Usando o pacote

Adicione o pacote ao seu *path* e importe o pacote

```
import sys
sys.path.append("/caminho/MachineTeacher")

import machine_teacher
```

Importe um dos teachers prontos

```
T1 = machine_teacher.Teachers.DoubleTeacher()
```

Importe um dos learners prontos ou crie um a partir de um modelos do sklearn

```
L1 = machine_teacher.Learners.SVMLinearLearner()
```

```
from sklearn.svm import LinearSVC
class MyLearner(machine_teacher.GenericLearner):
        name = "MyLearner"

        def start(self):
                self.model = LinearSVC()

L2 = MyLearner()
```

Rode um experimento sem limite de tempo

Obs.: assumo que o dataset (X, y) já foi carregado em memória

```
L1 = machine_teacher.Learners.SVMLinearLearner()
T1 = machine_teacher.Teachers.DoubleTeacher()

result = machine_teacher.protocol(T1, L1, X, y)
```

Rode um experimento com limite de tempo

Obs.: assumo que o dataset (X, y) já foi carregado em memória

```
L1 = machine_teacher.Learners.SVMLinearLearner()
T1 = machine_teacher.Teachers.DoubleTeacher()
time_limit = 10.0

result = machine_teacher.protocol(T1, L1, X, y, time_limit)
```