Aprendizagem Automática

Licenciatura em Engenharia Informática

Trabalho Prático 2024/2025

-- KNN e Naïve Bayes (v1.3, 2024.11.07) --

1. Objetivo

Implementar os algoritmos **K-Nearest Neighbors** e **Naïve de Bayes** para problemas de classificação com **atributos numéricos**.

2. Descrição do trabalho

Pretende-se implementar os algoritmos KNN e Naïve de Bayes para problemas de **classificação** que permitam a integração com o ambiente scikit-learn. As **classes a desenvolver** devem permitir criar o modelo, aplicar o modelo e calcular o desempenho. A criação dos modelos dependem da escolha de:

- KNN: nº de vizinhos e métrica de Minkowski;
- Naïve de Bayes: valor de suavização (valor adicionado à variância dos atributos para estabilidade do cálculo) e peso dos exemplos.

Pretende-se igualmente testar a implementação através da sua aplicação a diversos conjuntos de dados, avaliando o desempenho dos modelos para diferentes hiperparâmetros.

3. Implementação

As classes a implementar, **KNNeighborsUE** e **NBayesUE**, deverão ser o mais compatível possível com o ambiente scikit-learn, ou seja, os parâmetros de entrada e saída dos métodos deverão permitir a substituição destes algoritmos por outros implementados no scikit-learn. Assim, para cada classe, deverá implementar os seguintes métodos:

- **inicialização** do objeto: definição dos parâmetros do algoritmo
- método **fit(X,y)**[1]: constrói (e guarda numa estrutura de dados adequada) o modelo a partir do conjunto de dados fornecido:
  - X: array com forma (nexemplos, natributos). Dados de treino;
  - y: array com forma (nexemplos). Etiquetas;
  - devolve a classe
- método **predict(X)**: aplica o modelo a cada exemplo de X, e devolve as etiquetas previstas:
  - X: array com forma (nexemplos, natributos). Dados de teste;
  - devolve array com forma (nexemplos). Previsão;
- método **score(X,y)**: aplica o modelo a cada exemplo de X e compara a previsão com a etiqueta real y, devolvendo a **exatidão**:
  - X: array com forma (nexemplos, natributos). Dados (treino ou teste);
  - y: array com forma (nexemplos). Etiquetas reais (treino ou teste);
  - devolve valor da exatidão

[1] Os arrays X e y devem ser compatíveis com numpy.ndarray.

As classes poderão ter outros métodos que considerar relevantes para implementar as funcionalidades requeridas.

3.1. KNNeigborsUE

Os parâmetros para a inicialização do objeto são:

- **k**: int; valor por omissão é 3
  Nº de vizinhos mais próximos a considerar.

- **p**: float, valor por omissão é 2.0
  Potência para a métrica Minkowski; quando p=1 é equivalente à distância de Manhattan; com p=2 temos a distância Euclidiana.

Métrica de Minkowski

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

3.2. NBayesUE

Os parâmetros para a inicialização do objeto são:

- **suave**: float, valor por omissão é 1e-9
  Valor de suavização (valor adicionado à variância dos atributos para estabilidade do cálculo).

Assuma que a verossimilhança, $P(x_i|y)$, segue uma distribuição Gaussiana para cada x_i em y, ou seja,

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right)$$

Implementação adicional. Não sendo obrigatória, será valorizada a possibilidade de considerar pesos distintos para os exemplos. Neste caso, os métodos **fit()** e **score()** terão um parâmetro adicional:

- **peso_amostra**: array de floats com forma (nexemplos); valor por omissão é *None*
  Pesos a serem aplicados aos exemplos individuais.

3.3 Programa

Para além da implementação das classes, será necessário implementar um programa que, dado um conjunto de dados, cria um modelo e avalia-o.

4. Dados

Estão disponíveis no moodle vários conjuntos de dados para testar os algoritmos. Assuma que o input é um **ficheiro csv** onde:

- a 1ª linha do ficheiro identifica os atributos
- o atributo a prever está na última coluna

Recomenda-se a consulta e análise dos ficheiros para verificar a pertinência de cada atributo para o problema.

A utilização da biblioteca **Pandas** (pandas.pydata.org) permite a leitura do ficheiro csv para uma estrutura de dados compatível com o sklearn. A função **read_csv()** lê o ficheiro para um **DataFrame**, uma estrutura de dados semelhante a uma folha de cálculo, que permite identificar colunas (e linhas) a partir do nome. Num *DataFrame* também é possível identificar colunas (e linhas) através de índices.

O ficheiro **read_split_write.py** exemplifica a utilização de DataFrame (usando identificadores dos atributos e índices) com o conjunto de dados "contact lenses".

5. Relatório

O relatório deve incluir a:

1. explicação da implementação escolhida, nomeadamente as **estruturas de dados** que permitem guardar o modelo;

2. pesquisa e **escolha de um conjunto de dados** disponível na web (ou criação de um novo conjunto). O conjunto de dados deve ser identificado com a indicação adicional da sua localização original;

3. apresentação do desempenho dos modelos obtidos por cada um dos algoritmos com os seguintes hiperparâmetros:

    a. KNN: k={1, 5, 9}, p={1, 2}
    b. Naïve Bayes: smooth={1e-9, 1e-5}

    O desempenho deve ser calculado para **4 conjuntos de dados** distintos: "iris", "rice", "wdbc" (disponibilizados no moodle) e o conjunto de dados selecionado (ver ponto anterior). Deve ser utilizado um **método de avaliação de desempenho à escolha** (por exemplo, divisão treino-teste ou validação cruzada).

    Os resultados de desempenho devem ser analisados à luz da capacidade de generalização dos modelos.

6. Condições gerais

O trabalho deverá ser desenvolvido em **grupos de 2 alunos** e submetido no moodle através de um ficheiro **.tgz** ou **.zip** até à data indicada no moodle; o nome do ficheiro deverá ser os números dos alunos por ordem crescente (e.g. "33333_44444.zip").

Os trabalhos submetidos após o prazo terão uma penalização de 2 valores por dia (máximo 2 dias).

O conteúdo do ficheiro submetido deve incluir o código fonte do trabalho, adequadamente comentado e o relatório em formato **PDF**.

A apresentação do trabalho é obrigatória e será realizada em dia e horário a combinar.