Agentes Probabilísticos - Redes Bayesianas

Igor Pereira dos Santos, Igor Kunrath, Leandro Roberto

Novembro de 2019

1 Introdução

Esse artigo tem como objetivo, demonstrar a pratica dos agentes probabilísticos utilizando Redes Bayesianas como base de conhecimento, também utilizando algumas bibliotecas em Python para exemplificar em prática, contudo o exemplo utilizado será da ligação entre o jogo de tênis e a situação climática durante os jogos.

2 Redes Bayesianas

As redes bayesianas foram desenvolvidas no inicio dos anos 1980, também conhecidas como redes de opinião, redes causais e gráficos de dependência probabilística.

Redes bayesianas são modelos que trabalham por meio de grafo acíclico direcionado, nos quais os arcos representam dependência condicional e os nós representam variáveis de um domínio.

3 Implementação

Utilizaremos as bibliotecas numpy, pandas, GaussianNB, LabelEncoder, train test split, acurracy score. O problema consiste em utilizar as redes bayesianas para verificar as probabilidades de acontecer um jogo de tênis ou não, de acordo com o clima do dia, ao todo serão 14 dias, utilizaremos uma planilha em csv com os dados dos jogos.

3.1 Como Utilizar

Primeiro vamos importar as bibliotecas.

Table 1: Import das bibliotecas

```
1 | import numpy as np
2 | import padas as pd
3 | from sklearn.naive_bayes import GaussianNB
4 | from sklearn.preprocessing import LabelEncoder
5 | from sklearn.model_selection import train_test_split
6 | from sklearn.metrics import accuracy_score
```

Depois carregaremos um arquivo em csv, utilizando a função da biblioteca: play_tenis = pd.read_csv("PlayTennis.csv")

Na tabela abaixo contem uma amostra dos dados que utilizaremos no teste, como perspectivas do tempo, temperatura, umidade e condições do vento.

Table 2: Jogar Tênis

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes

Neste exemplo, usamos a biblioteca Python SKLearn para criar um modelo e fazer previsões. A biblioteca SKLearn requer que os recursos sejam matrizes numéricas. Portanto, precisaremos converter as informações categóricas em nossos dados em números.
Existem várias maneiras de fazer isso, manteremos simples e usaremos um LabelEncoder para este exemplo.
Um LabelEncoder converte dados categóricos em um número que varia de 0 a n-1, onde n é o número de classes na variável.

Por exemplo, no caso do Outlook, existem 3 classes - Nublado, Chuva, Ensolarado. Estes são representados como 0,1,2 em ordem alfabética.

1	number = LabelEncoder()
2	play_tennis['Outlook'] = number.fit_transform(play_tennis['Outlook'])
3	play_tennis['Temperature'] = number.fit_transform(play_tennis['Temperature'])
4	play_tennis['Humidity'] = number.fit_transform(play_tennis['Humidity'])
5	play_tennis['Wind'] = number.fit_transform(play_tennis['Wind'])
6	play_tennis['Play Tennis'] = number.fit_transform(play_tennis['Play Tennis'])

Vamos definir os recursos e as variáveis de destino.

1	features = ["Outlook", "Temperature", "Humidity", "Wind"]
2	target = "Play Tennis"

Para validar o desempenho do nosso modelo, criamos um trem, teste de divisão. Construímos o modelo usando o conjunto de dados de trem e validaremos o modelo no conjunto de dados de teste.

Usamos train_test_split do SKLearn para fazer isso.

1	features_train, features_test, target_train, target_test = train_test_split(play_tennis[features],
2	play_tennis[target],
3	test_size = 0.33,
4	random_state = 54)

Vamos criar o modelo agora.

1	model = GaussianNB()
2	model.fit(features_train, target_train)

Agora estamos prontos para fazer previsões sobre os recursos de teste.

Também mediremos o desempenho do modelo usando a pontuação de precisão.

A pontuação de precisão mede o número de previsões corretas.

A precisão é, neste caso, de 0,80000000000000004 Agora, suponha que desejamos prever as condições,

que fornece uma previsão 1 (Sim)

1	pred = model.predict(features_test)
2	accuracy = accuracy_score(target_test, pred)

Outlook	Temperatura	Umidade	Vento
Chuva	Suave	Alto	Fraco

1	print model.predict([[1,2,0,1]])

4 Considerações Finais

As redes bayesianas fornecem uma capacidade de juntar uma quantidade grande de dados e reverter isso em estáticas para auxiliar com precisão um probabilidade que desejamos saber, com a linguagem python e as bibliotecas necessárias transforma algo complexo e algo simples de se fazer.

5 Referencias

https://pt.wikipedia.org/wiki/L%C3%B3gica_probabil%C3%ADstica
https://www.cos.ufrj.br/ ines/courses/cos740/leila/cos740/aprBayesianas.pdf
https://www.cs.usfca.edu/ galles/cs662/lecture/lecture16.pdf
https://www.organicadigital.com/blog/algoritmo-de-classificacao-naive-bayes/