TICS-411 Minería de Datos

Clase 11: Naive Bayes

Alfonso Tobar-Arancibia
alfonso.tobar.a@edu.uai.cl

Naive Bayes: Preliminares

> También conocido como `Clasificador Inexperto de Bayes`, es uno de los clasificadores más conocidos y sencillos.

Se hizo particularmente conocido como uno de los primeros algoritmos en funcionar como Clasificador de Spam de manera efectiva.

- Es un modelo netamente probabilístico basado en el `Teorema de Bayes`.
  - Aprende una distribucional de Probabilidad Condicional.
  - Dado un punto x_i, el modelo retorna la "probabilidad" de que x_i pertenezca a una clase específica.

Definiciones

Probabilidad Condicional

$$P(X|C) = \frac{P(X \cap C)}{P(C)}$$

Se lee como la Probabilidad de que Ocurra X dado que tenemos C.

Teorema de Bayes

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

La `probabilidad a posteriori` (LHS), depende de el ***Likelihood***, la ***probabilidad a priori*** y la ***evidencia*** (RHS).

Independencia Condicional

$$P(X_1, X_2, \ldots, X_k|C) = \prod_{i=1}^{k} P(X_i|C)$$

Si asumimos ***independencia***, entonces la probabilidad conjunta de k eventos condicionados, se calcula como la ***productoria de las probabilidades condicionales independientes***.

Ejemplo básico

Supongamos que:

- Sabemos que la Meningitis produce Tortícolis el 50% de las veces.
- La probabilidad de tener meningitis es: $1/50000$.
- La probabilidad de tener Tortícolis: $1/20$.

Si su paciente tiene tortícolis, ¿Cuál es la probabilidad de que tenga Meningitis?

$$P(M|T) = \frac{P(T|M)P(M)}{P(T)} = \frac{0.5 \cdot 1/50000}{1/20} = 0.0002$$

Modelo Naive Bayes: Aprendizaje

$$P(y = C_j|X_1, X_2, \ldots, X_k) = \frac{P(X_1, X_2, \ldots, X_k|y = C_j)P(y = C_j)}{P(X_1, X_2, \ldots, X_k)}$$

- $P(y = C_j|X)$ sería la probabilidad de que la predicción del modelo sea C_j dado que lo alimentamos con las variables X.
- Luego $P(y = C_j)$ es la probabilidad *a priori* de que la clase sea C_j.
- $P(X|y = C_j)$ es el likelihood (verosimilitud). Corresponde a la distribución de probabilidad de las variables X cuando la clase es C_j.
- $P(X)$ es la evidencia, y normalmente es muy complejo de calcular.

Por simplicidad reduciremos $X_1, X_2, \ldots, X_k$ a X.

$P(X)$ tiene como única función la de normalizar la probabilidad para que vaya en un rango entre 0 y 1.

Modelo Naive Bayes: Predicción

$$\hat{y}_i = \underset{C_j}{argmax}\ P(y = C_j|X)$$

donde,

$$P(y = C_j|X) \propto \prod_{i=1}^{k} P(X|y = C_j)P(y = C_j)$$

La predicción de `Naive Bayes` corresponde a la clase que entrega ***[un estimado de]*** la `Probabilidad a Posteriori` más grande.

Ejemplo

Clima	Temperatura	Humedad	Viento	Juego Tenis?
Soleado	Calor	Alta	Débil	No
Soleado	Calor	Alta	Fuerte	No
Cubierto	Calor	Alta	Débil	Sí
Lluvia	Media	Alta	Débil	Sí
Lluvia	Fría	Normal	Débil	Sí
Lluvia	Fría	Normal	Fuerte	No
Cubierto	Fría	Normal	Fuerte	Sí
Soleado	Media	Alta	Débil	No
Soleado	Fría	Normal	Débil	Sí
Lluvia	Media	Normal	Débil	Sí
Soleado	Media	Normal	Fuerte	Sí
Cubierto	Media	Alta	Fuerte	Sí
Cubierto	Calor	Normal	Débil	Sí
Lluvia	Media	Alta	Fuerte	No

¿Cómo clasificamos el siguiente punto?

$$X = [C = Soleado, T = Media, H = Alta, V = Débil]$$

Probabilidad de Sí

$$P(y = Sí|X) = P(X|y = Sí)P(y = Sí)$$

Probabilidad de No

$$P(y = No|X) = P(X|y = No)P(y = No)$$

Ejemplo

$$P(y = Sí|X) = P(C = Soleado|y = Sí)P(T = Media|y = Sí)P(H = Alta|y = Sí)P(V = Débil|y = Sí)P(y = Sí)$$

$$P(y = No|X) = P(C = Soleado|y = No)P(T = Media|y = No)P(H = Alta|y = No)P(V = Débil|y = No)P(y = No)$$

Probabilidad Condicional para clase Sí

$$P(C = Soleado|y = Sí) = 2/9$$

$$P(T = Media|y = Sí) = 4/9$$

$$P(H = Alta|y = Sí) = 3/9$$

$$P(V = Débil|y = Sí) = 6/9$$

Probabilidad Condicional para clase No

$$P(C = Soleado|y = No) = 3/5$$

$$P(T = Media|y = No) = 2/5$$

$$P(H = Alta|y = No) = 4/5$$

$$P(V = Débil|y = No) = 2/5$$

Probabilidad a priori

$$P(y = Sí) = \frac{9}{14} = 0.64$$

$$P(y = No) = \frac{5}{14} = 0.3$$

Predicción

$$P(y = Sí|X) = P(C = Soleado|y = Sí)P(T = Media|y = Sí)P(H = Alta|y = Sí)P(V = Débil|y = Sí)P(y = Sí)$$

$$P(y = Sí|X) = \frac{2}{9} \cdot \frac{4}{9} \cdot \frac{3}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = 0.0141$$

$$P(y = No|X) = P(C = Soleado|y = No)P(T = Media|y = No)P(H = Alta|y = No)P(V = Débil|y = No)P(y = No)$$

Smoothing

Supongamos otro dataset más pequeño:

Clima	Juego Tenis?
Soleado	No
Soleado	No
Cubierto	No
Lluvia	Sí
Lluvia	Sí
Lluvia	No
Cubierto	Sí
Soleado	No
Soleado	No
Lluvia	Sí
Soleado	No
Cubierto	Sí
Cubierto	Sí
Lluvia	No

> ⚠ Dado que `Naive Bayes` se calcula como una Productoria, al tener probabilidades 0 inmediatamente la ***Probabilidad a Posteriori es 0***.

- : Es un Hiperparámetro. Si se le llama Laplace Smoothing, si entonces se le llama Lidstone Smoothing.
- **M**: Corresponde al número de posibles valores que puede tomar
- : Corresponde a la cantidad de registros que toman el valor de la variable solicitado en la clase .
- : Corresponde a la cantidad de registros totales que tienen la clase .

Laplace Smoothing

Sin Laplace

C	Sí	No
Soleado	0/6	5/8
Cubierto	3/6	1/8
Lluvia	3/6	2/8
Total	1	1

Con Laplace

C	Sí	No
Soleado	1/9	6/11
Cubierto	4/9	2/11
Lluvia	4/9	3/11
Total	1	1

En este caso y ya que Clima puede tomar 3 valores: Soleado, Cubierto y Lluvia.

Variables Continuas

Humedad	Juego Tenis?
85	No
90	No
86	Sí
96	Sí
80	Sí
65	Sí
70	Sí
70	No
95	No
80	Sí
91	No
70	Sí
90	Sí
75	Sí

Podemos calcular el Likelihood como una PDF ***(Probability Density Function)***. La más común: Distribución Normal ***(Gaussian Naive Bayes)***.

C	Sí	No
Mean	79.1	86.2
Std	10.2	9.7

Variables Continuas: Predicción

Luego la predicción es Sí.

Detalles Técnicos

Fortalezas

- Fácil de Implementar
- A menudo tiene un rendimiento decente a pesar de que las variables pueden no ser independientes.
- Puede aprender de forma incremental.
- Valores faltantes son ignorados en el proceso de Aprendizaje.
- Modelo robusto frente a datos atípicos y/o irrelevantes.

Debilidades

- Asumir clases condicionadas produce probabilidades sesgadas.
- Dependencias entre las variables no pueden ser modeladas.

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Implementación en Scikit-Learn

Multinomial Naive Bayes (Normal)

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha = 1)
nb.fit(X_train, y_train)

y_pred = nb.predict(X_test)
y_proba = nb.predict_proba(X_test)
```

Gaussian Naive Bayes

```python
from sklearn.naive_bayes import GaussianNB

gb = GaussianNB()
gb.fit(X_train, y_train)

y_pred = gb.predict(X_test)
y_proba = gb.predict_proba(X_test)
```

Adeus