CSI, examen de aprendizaje automático

21 de diciembre del 2021. Tiempo: 1 hora 50 minutos

Las preguntas 1 a 6 valen 1 punto. Las preguntas 7 a 10 valen 2 puntos (el examen es sobre 14 puntos).

Pregunta 1 En el contexto de un problema de agrupamiento (clustering) no supervisado,

1. Indica al menos tres criterios de determinar la distancia entre dos grupos (clusters). Sólo es necesario mencionarlos.
2. Indica cuál de ellos es más generoso y cuál es más conservador al determinar el parecido entre los dos grupos.

Respuesta:

1. *Single linkage*, *average linkage* y *complete linkage*.
2. El primero es el más generoso y el último es el más conservador.

Pregunta 2 Explica como se aplica un modelo *k-nn* (k nearest neighbours) para realizar una predicción sobre un ejemplar concreto.

1. En un problema de clasificación.
2. En un problema de regresión.

Respuesta: Un modelo *k-nn* calcula dentro del conjunto de entrenamiento el subconjunto de los k ejemplares más cercanos al ejemplar sobre el que queremos realizar la predicción. Por ejemplo,

1. En un problema de clasificación, se devuelve la clase más frecuente dentro de dicho subconjunto.
2. En un problema de regresión, se puede devolver el promedio de los valores objetivo dentro de dicho subconjunto de ejemplares.

Pregunta 3 Hemos visto que los diferentes métodos de regresión se basan en optimizar una función para obtener los mejores parámetros durante el entrenamiento. ¿Cuál es la diferencia principal entre la regresión lineal tradicional (ordinaria) y la regresión logística a nivel de la función que optimizan?

Respuesta: En la primera se buscan los parámetros que minimiza el error cuadrático mientras que en la segunda se buscan los parámetros que maximizan el logaritmo de la verosimilitud (log-likelihood).

Pregunta 4 Si observamos que el error que comete nuestro modelo con el conjunto de datos de entrenamiento es menor que el error que comete con el conjunto de datos de validación,

1. ¿Qué tipo de problema tenemos según la terminología de aprendizaje automático?
2. ¿Por qué sucede esto?

Respuesta:

1. Se trataría de un problema de sobreajuste (overfitting).
2. Sucede porque el modelo aprende propiedades de los datos especificas del conjunto de entrenamiento pero no válidas para otras muestras del mismo conjunto de datos original.

Pregunta 5

1. ¿Qué tienen en común PCA y t-SNE?
2. ¿En qué se diferencian?

Respuesta:

1. Ambos son métodos de reducción de dimensionalidad.
2. En el nuevo espacio reducido, PCA intenta respetar la estructura global de los datos mientras que t-SNE intenta respetar la estructura local. t-SNE funciona en general mucho mejor que PCA (PCA es un método lineal de reducción mientras que t-SNE es uno no lineal).

Pregunta 6 A lo largo del curso hemos visto diferentes técnicas de reescalado o normalización de los datos.

1. ¿Por qué son necesarias estas técnicas?
2. A nivel de `scikit learn`, ¿qué diferencia hay entre `MinMaxScaler` y `StandardScaler`? Indica la fórmula que se usaría en cada caso para transformar un valor v de una columna X^i de una matriz de datos X.

Respuesta:

1. Para garantizar que las variables (atributos) puedan contribuir *a priori* de forma equitativa al modelo. Se pretende evitar que una variable contribuya más en un modelo simplemente porque su intervalo de variación sea mayor que el resto pero no porque realmente sea más importante.

2. En un `MinMaxScaler`, v se convierte en

$$v' = \frac{v - X^i_{min}}{X^i_{max} - X^i_{min}}, \tag{1}$$

donde X^i_{min} y X^i_{max} son, respectivamente, el valor mínimo y máximo de los valores de X^i. En un `StandardScaler`, v se convierte en

$$v' = \frac{v - \mu^i}{\sigma^i}, \tag{2}$$

donde μ^i y σ^i son, respectivamente, el valor promedio y la desviación estándar de los valores de X^i.

Pregunta 7 En el código siguiente

```
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler

X_normalized = StandardScaler().fit_transform(iris.data)
X_pca = PCA(n_components=2).fit_transform(X_normalized)
X_tsne = TSNE(n_components=2, perplexity=10).fit_transform(X_normalized)

print(X_pca.shape, X_tsne.shape)
```

```
(150, 2) (150, 2)
```

1. ¿Para qué sirve el parámetro `n_components` de las llamadas?

2. ¿Para qué sirve el parámetro `perplexity` de la llamada a t-SNE?

3. ¿Qué sucede al aumentar el valor de `perplexity`?

Respuesta:

1. Para determinar el numero de dimensiones sobre las cuales queremos proyectar el conjunto de datos, típicamente dos.

2. Para determinar cuantos vecinos de un punto son relevantes para la proyección en un espacio de dimensión menor (es una medida efectiva del número de puntos hacia los cuales un punto dado es atraído).

3. Al aumentar su valor a partir de 1, los grupos en el espacio reducido tienden a ser más claros (cada grupo tienden a ser más denso en el nuevo espacio y más separado del resto de grupos).

Pregunta 8 Queremos predecir el valor de una variable y a partir de una variable x. Supongamos que nuestros datos tienden a seguir en realidad un polinomio de segundo grado, es decir,

$$y = ax^2 + bx + c + \epsilon \tag{3}$$

donde ϵ es un cierto ruido aleatorio que no depende de x. Teniendo en cuenta que en aprendizaje automático el error generalización se descompone en sesgo (bias) y variancia, indica como seria el sesgo o la variancia en los siguientes modelos:

1. *a*) Polinomio de segundo grado.
   b) Recta.
   c) Polinomio de grado elevado (grado mucho mayor que dos).
2. ¿En qué modelos se produciría sobreajuste (overfitting) y en qué modelos (underfitting)?

Respuesta:

1. *a*) Sesgo y variancia bajos.
   b) Sesgo alto.
   c) Variancia alta.
2. Infraajuste con la recta y sobreajuste con el polinomio de grado elevado.

Pregunta 9 El algoritmo k-means nos permite encontrar una aproximación al mejor particionamiento de un conjunto de datos en k grupos (clusters). Para un conjunto de datos, queremos encontrar el valor de k óptimo, es decir el valor de k más representativo de la estructura de los datos. Para ello ejecutamos `KMeans` de `scikit learn` con diferentes valores de k. Sería conveniente ejecutar `KMeans` varias veces para un mismo valor de k? Por qué sí o por qué no?

Respuesta: Sí, seria conveniente porque el resultado de una ejecución de `KMeans` puede depender de los prototipos elegidos inicialmente. Al hacerlo así, podemos calcular un valor promedio de la calidad de los grupos para un k fijo. El inconveniente es la pérdida de eficiencia temporal. A cambio, la elección del k óptimo es más fiable.

Nota: los estudiantes que responden que sí pero justificando su respuesta en base a poder aplicar métodos de validación cruzada (k-fold cross validation) o simplemente dividir los datos en entrenamiento y validación deben preguntarse para qué es necesario dividir entre entramiento y test en un método de aprendizaje no supervisado; también deben preguntarse si al ajustar el modelo a un

subconjunto de los datos uno se arriesga a infraestimar el valor de real de k porque una de las classes no esté suficientemente representada en el subconjunto de datos seleccionado.

Pregunta 10 En el código siguiente,

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features.
Y = iris.target

logreg = LogisticRegression(C=1e5, solver='lbfgs', multi_class='multinomial')

# Create an instance of Logistic Regression Classifier and fit the data.
logreg.fit(X, Y)
```

```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='multinomial', n_jobs=None, penalty='l2',
          random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
          warm_start=False)
```

1. ¿Para qué sirve el parámetro `C` de la llamada a `LogisticRegression`?
2. ¿Es un parámetro o un hiperparámetro? Justifica tu respuesta.
3. ¿Qué sucede al aumentar su valor?
4. ¿Qué significa el valor del parámetro `multi_class`?

Respuesta:

1. `C` es un parámetro de regularización (para evitar el sobreajuste).
2. Es un hiperparámetro porque se fija antes de buscar los parámetros que maximizan la verosimilitud.
3. Al aumentar su valor, menor la regularización.
4. Indica que no se trata de un problema de clasificación binaria.