AAI.EX.20200201

Ejercicios elaborados con fines educativos, inspirados en los contenidos evaluados en el exámen del 01/02/2022 (convocatoria Feb-2020) de Aprendizaje Automático 1 del MUICD de la UNED.

Este documento no es una copia ni una transcripción del examen oficial, sino una redacción propia de ejercicios conceptualmente equivalentes.

Test

Pregunta correcta: +0.5 puntos
Pregunta incorrecta: -0.1 puntos

AAI.EX.20200201.T.1

Enunciado AAI.EX.20200201.T.1
¿En qué situación no suele ser la mejor opción aplicar aprendizaje automático?

a) Automatizar decisiones basadas en reglas muy complejas y difíciles de mantener.
b) Facilitar el descubrimiento de patrones que ayuden a las personas a aprender.
c) Resolver problemas donde ya existe un algoritmo determinista eficiente y conocido.
d) Crear sistemas capaces de adaptarse a cambios en el entorno.

Solución AAI.EX.20200201.T.1
Respuesta correcta: c)

Justificación:

El aprendizaje automático es especialmente útil cuando no existe una solución algorítmica clara o cuando el entorno cambia. Si ya hay un algoritmo exacto y eficiente, usar ML añade complejidad innecesaria.

AAI.EX.20200201.T.2

Enunciado AAI.EX.20200201.T.2
¿Qué tipo de algoritmo de Aprendizaje Automático no sería nada conveniente para que un robot aprendiera a desplazarse por terrenos desconocidos?

a) Aprendizaje supervisado.
b) Aprendizaje no-supervisado.
c) Aprendizaje por refuerzo.
d) Aprendizaje semi-supervisado.

Solución AAI.EX.20200201.T.2
Respuesta correcta: a)

Justificación:

El aprendizaje supervisado requiere datos etiquetados de antemano, algo poco realista en un entorno desconocido. El aprendizaje por refuerzo es el enfoque natural para aprender mediante interacción con el entorno.

AAI.EX.20200201.T.3

Enunciado AAI.EX.20200201.T.3
¿Qué problema puede surgir si se usan los datos de prueba para ajustar hiperparámetros?

a) El modelo puede ajustarse en exceso a los datos de prueba.
b) La estimación del error de generalización será demasiado optimista.
c) El modelo puede quedar infraajustado respecto al conjunto de prueba.
d) Las opciones a) y b) son correctas.

Solución AAI.EX.20200201.T.3
Respuesta correcta: d)

Justificación:

El conjunto de test debe usarse solo una vez para evaluar el rendimiento final. Ajustar hiperparámetros con él provoca fuga de información y una estimación optimista del rendimiento.

AAI.EX.20200201.T.4

Enunciado AAI.EX.20200201.T.4
¿Qué variante de PCA es más adecuada cuando los datos siguen una estructura no lineal?

a) PCA estándar.
b) PCA con kernel.
c) PCA incremental.
d) PCA aleatorizado.

Solución AAI.EX.20200201.T.4
Respuesta correcta: b)

Justificación:

Kernel PCA permite proyectar los datos a un espacio de mayor dimensión mediante un kernel, capturando relaciones no lineales que el PCA clásico no puede modelar.

AAI.EX.20200201.T.5

Enunciado AAI.EX.20200201.T.5
En un problema de clasificación con SVM lineal y un conjunto de datos muy grande (millones de ejemplos y muchas características), ¿qué formulación es más eficiente?

a) Las SVM lineales solo usan la forma primal.
b) Las SVM lineales solo usan la forma dual.
c) Resolver el problema en su forma primal es más rápido.
d) Resolver el problema en su forma dual es más rápido.

Solución AAI.EX.20200201.T.5
Respuesta correcta: c)

Justificación:

Las SVM lineales existen tanto la forma primal como la dual, aunque su rendimiento sea distinto.

En SVM lineales, la forma primal escala mejor cuando el número de instancias es muy grande. La forma dual depende del número de ejemplos y se vuelve ineficiente en estos casos.

AAI.EX.20200201.T.6

Enunciado AAI.EX.20200201.T.6
Si entrenar un árbol de decisión con 1 millón de ejemplos lleva 1 hora, aproximadamente ¿cuánto tiempo llevará entrenarlo con 10 millones, manteniendo el resto constante?

a) 11,7 horas.
b) 5,6 horas.
c) 12,4 horas.
d) 15,1 horas.

Solución AAI.EX.20200201.T.6
Respuesta correcta: a)

Justificación:

El coste de entrenamiento de un árbol de decisión crece aproximadamente como $O(n \log n)$. Con estos costes, multiplicar n por 10 multiplica el tiempo por un poco más de 10, porque el logaritmo crece muy lentamente. Intuitivamente y sin hacer cálculos, se puede intuir que el valor correcto será el mayor y más próximo a 10 entre los disponibles.

AAI.EX.20200201.T.7

Enunciado AAI.EX.20200201.T.7
¿Cuál de los siguientes problemas no corresponde al uso típico de algoritmos de clustering?

a) Agrupar clientes según su comportamiento.
b) Mejorar la organización interna de resultados en buscadores.
c) Identificar valores atípicos o comportamientos anómalos.
d) Asignar etiquetas conocidas a datos previamente etiquetados.

Solución AAI.EX.20200201.T.7
Respuesta correcta: d)

Justificación:

El clustering es aprendizaje no supervisado. La clasificación de datos con etiquetas conocidas es un problema de aprendizaje supervisado, no de agrupamiento.

AAI.EX.20200201.T.8

Enunciado AAI.EX.20200201.T.8
¿Qué técnicas se utilizan para estimar un buen valor de k en K-means?

a) Analizar el punto donde la inercia deja de disminuir bruscamente (método del codo).
b) Buscar el máximo de la puntuación de silueta.
c) Ambas opciones son correctas.
d) No existen métodos prácticos para estimar k.

Solución AAI.EX.20200201.T.8
Respuesta correcta: c)

Justificación:

Tanto el método del codo como la puntuación de silueta son heurísticas comunes para seleccionar un número razonable de clusters.

AAI.EX.20200201.T.9

Enunciado AAI.EX.20200201.T.9
En una red neuronal con 10 entradas, una capa oculta de 50 neuronas y una capa de salida de 3 neuronas, ¿qué dimensiones tiene la matriz de entrada X?

a) 10×50.
b) M $\times$ 10, siendo M el número de ejemplos del lote.
c) $10 \times$ M, siendo M el número de ejemplos del lote.
d) 50×10.

Solución AAI.EX.20200201.T.9
Respuesta correcta: b)

Justificación:

Cada fila de X representa una instancia y cada columna una característica, por lo que su forma es (número de ejemplos, número de características) = $M \times 10$

AAI.EX.20200201.T.10

Enunciado AAI.EX.20200201.T.10
¿Qué función de activación es la más adecuada en la capa de salida de un perceptrón multicapa para clasificar imágenes con dígitos del 0 al 9?

a) ReLU.
b) Sigmoide.
c) Softmax.
d) Ninguna función de activación.

Solución AAI.EX.20200201.T.10
Respuesta correcta: c)

Justificación:

En un problema de clasificación multiclase mutuamente excluyente como el reconocimiento de dígitos del 0 al 9, la función softmax es la más adecuada en la capa de salida de un perceptrón multicapa, ya que transforma los valores de activación en una distribución de probabilidad normalizada sobre las 10 clases, cumpliendo que $\sum_{k=1}^{1} 0p_k = 1$. Esto modela correctamente que cada imagen pertenece a una única clase y permite el uso de la entropía cruzada como función de pérdida, lo que da lugar a un entrenamiento estable y a una interpretación probabilística coherente de las salidas.

ReLU y sigmoide no son adecuadas en la capa de salida para este problema porque ambas actúan de forma independiente sobre cada neurona y no imponen ninguna normalización ni competencia entre clases. ReLU produce salidas no acotadas y no interpretables como probabilidades, mientras que la sigmoide genera valores en (0,1) pero permite que varias clases tengan alta activación simultáneamente, lo cual es incompatible con un problema de clasificación multiclase mutuamente excluyente como el reconocimiento de dígitos, donde cada imagen pertenece a una única clase.

AAI.EX.20200201.D

AAI.EX.20200201.D.1

Puntuación máxima: 5 puntos
Extensión máxima orientativa: 2 caras

Enunciado AAI.EX.20200201.D.1
Redacta una explicación sobre el algoritmo de descenso por gradiente usado para entrenar modelos de aprendizaje automático. Tu texto debe incluir:

- Una descripción clara del procedimiento (qué intenta hacer y cómo actualiza los parámetros).
- Qué papel juega la función de coste (loss) y cómo se utiliza para guiar el ajuste.
- Los hiperparámetros más relevantes y cómo afectan al comportamiento del algoritmo.
- Problemas habituales que pueden aparecer durante el entrenamiento (por ejemplo, relacionados con la convergencia o la estabilidad numérica).
- Principales variantes de implementación del descenso por gradiente y una comparación razonada entre ellas.

Solución AAI.EX.20200201.D.1

El descenso por gradiente (*Gradient Descent*) es un algoritmo iterativo de optimización utilizado para ajustar los parámetros de un modelo de aprendizaje automático minimizando una función de coste. En cada iteración, los parámetros se actualizan en la dirección opuesta al gradiente de la función de coste, ya que esta es la dirección de máximo descenso.

La regla de actualización general es:

$$\theta \leftarrow \theta - \eta \nabla_\theta J(\theta)$$

donde:

- θ representa el vector de parámetros del modelo,
- $J(\theta)$ es la función de coste,
- $\nabla_\theta J(\theta)$ es el gradiente de dicha función,
- η es la tasa de aprendizaje (*learning rate*).

El proceso se repite hasta que el algoritmo converge o se alcanza un criterio de parada.

Función de coste
La función de coste cuantifica el error cometido por el modelo al realizar predicciones. Su objetivo es proporcionar una medida que pueda minimizarse mediante el ajuste de los parámetros. Para que el descenso por gradiente sea aplicable, la función de coste debe ser continua y derivable (o al menos sub-derivable).

Ejemplos comunes de funciones de coste son:

- **Error cuadrático medio (MSE)** en regresión lineal.
- **Log-loss** en clasificación logística.

En muchos casos se añade un término de regularización a la función de coste para evitar el sobreajuste, penalizando valores grandes de los parámetros.

Hiperparámetros Los principales hiperparámetros del descenso por gradiente son:

- **Tasa de aprendizaje** (η): controla el tamaño del paso en cada actualización. Valores grandes pueden provocar divergencia, mientras que valores pequeños ralentizan la convergencia.
- **Número de iteraciones / épocas**: determina cuántas veces se actualizan los parámetros.
- **Tamaño del lote (*batch size*)**: número de instancias utilizadas para calcular el gradiente en cada paso.
- **Parámetro de regularización**: influye en la complejidad del modelo y en la estabilidad del entrenamiento.
- **Criterio de parada**: basado en un número máximo de iteraciones o en la mejora mínima de la función de coste.

Problemas habituales Durante la aplicación del descenso por gradiente pueden aparecer varios problemas:

- **Mala escala de las características**: provoca una convergencia lenta y trayectorias en zig-zag.
- **Elección inadecuada de la tasa de aprendizaje**: puede causar divergencia o un entrenamiento excesivamente lento.
- **Mínimos locales y puntos de silla**: especialmente en funciones de coste no convexas.
- **Gradientes inestables**: pueden ser demasiado grandes o demasiado pequeños.
- **Sensibilidad al ruido y a valores atípicos**: depende de la función de coste utilizada.

Estos problemas suelen mitigarse mediante el escalado de variables, una buena inicialización, regularización y ajustes adecuados de hiperparámetros.

Existen distintas **variantes del descenso** por gradiente según la cantidad de datos utilizada para calcular el gradiente:

Descenso por gradiente por lote (*Batch Gradient Descent*): Utiliza todo el conjunto de entrenamiento para calcular cada gradiente.

- Ventajas: convergencia estable y precisa.
- Inconvenientes: alto coste computacional en conjuntos de datos grandes.

Descenso por gradiente estocástico (*Stochastic Gradient Descent*) Actualiza los parámetros utilizando una sola instancia en cada iteración.

- Ventajas: rápido y escalable a grandes volúmenes de datos.
- Inconvenientes: trayectoria ruidosa y menor estabilidad.

Descenso por gradiente por mini-lotes (*Mini-batch Gradient Descent*) Emplea pequeños subconjuntos de datos para cada actualización.

- Ventajas: equilibrio entre eficiencia y estabilidad.
- Inconvenientes: requiere elegir adecuadamente el tamaño del lote.

Comparación de implementaciones En la práctica, el descenso por gradiente por mini-lotes es el más utilizado, ya que combina la estabilidad del método por lote con la eficiencia computacional del método estocástico. El método por lote se reserva para problemas pequeños, mientras que el método estocástico es habitual en escenarios de datos masivos o aprendizaje en línea.