Tarea 7

April 22, 2021

#
Tarea 7
Por: Miguel Angel Soto Hernandez

0.1 Fundamentos de los modelos de aprendizaje automático

Los modelos de aprendizaje automático son herramientas matemáticas que permiten descubrir representaciones sintéticas de eventos externos, con el fin de obtener una mejor comprensión y predecir el comportamiento futuro. A veces estos modelos sólo se han definidos desde un punto de vista teórico, pero los avances en la investigación nos permiten ahora aplicar conceptos de aprendizaje automático para comprender mejor el comportamiento de sistemas sistemas complejos, como las redes neuronales profundas. En este capítulo, vamos a introducir y discutir algunos elementos fundamentales. Es posible que los lectores expertos ya conozcan estos elementos, pero aquí ofrecemos varias interpretaciones y aplicaciones posibles.

0.1.1 Modelos y datos

Los modelos de aprendizaje automático trabajan con datos. Crean asociaciones, descubren relaciones, descubren patrones, generan nuevas muestras y mucho más, trabajando con conjuntos de datos bien definidos, que son colecciones homogéneas de puntos de datos (por ejemplo observaciones, imágenes o medidas) relacionados con un escenario específico (por ejemplo, la temperatura de una habitación muestreada cada 5 minutos, o los pesos de una población de individuos)

Desgraciadamente, a veces las suposiciones o condiciones impuestas a los modelos de aprendizaje automático no están claras, y un largo proceso de entrenamiento puede dar lugar a un fracaso total de la validación. Podemos pensar en un modelo como una caja gris (cierta transparencia está garantizada por la simplicidad de muchos algoritmos comunes), donde una entrada vectorial X extraída de un conjunto de datos se transforma en una salida vectorial Y

Estructura y propiedades de los conjuntos de datos £Cuál es la naturaleza de X e Y? Un problema de aprendizaje automático se centra en el aprendizaje de relaciones abstractas que permiten una generalización consistente cuando se proporcionan nuevas muestras. Más concretamente, podemos definir un proceso estocástico de generación de datos con una distribución de probabilidad conjunta asociada:

$$p_{data}(x, y) = p(x|y)p(x)$$

El proceso p_{data} representa la expresión más amplia y abstracta del problema. Por ejemplo, un clasificador que debe distinguir entre retratos masculinos y femeninos se basará en un proceso

de generación de datos que define teóricamente las probabilidades de todos los rostros posibles, con respecto al atributo binario masculino/femenino. Está claro que nunca podemos trabajar directamente con p_{data} sólo es posible encontrar una fórmula bien definida que describa p_{data} en algunos casos limitados (por ejemplo, la distribución de todas las imágenes pertenecientes a un conjunto de datos).

0.1.2 Características de un modelo de aprendizaje automático

Capacidad de aprendizaje Para simplificar, supongamos que tenemos un algoritmo selector que puede buscar una hipótesis h_i en un conjunto H. Este elemento puede interpretarse de muchas maneras según el contexto. Por ejemplo, el conjunto de hipótesis podría corresponder al conjunto de parámetros razonables de un modelo o, en otro escenario, a un conjunto finito de algoritmos ajustados para resolver problemas específicos. Como la definición es general, no tenemos que preocuparnos por su estructura.

En el otro lado de este paisaje, está el conjunto de conceptos C que queremos aprender. Un concepto es una instancia de un problema que pertenece a una clase definida. De nuevo, la estructura puede variar, pero para simplificar el lector puede asumir que un concepto está asociado a un conjunto de entrenamiento clásico que contiene un número finito de puntos de datos.

Capacidad de un modelo Si consideramos un modelo supervisado como un conjunto de funciones parametrizadas, podemos definir la capacidad de representación como la capacidad intrínseca de una determinada función genérica para mapear un número relativamente grande de distribuciones de datos. Para entender este concepto, consideremos una función $f(x)$ que admite infinitas derivadas, y reescribámosla como una expansión de Taylor alrededor de un punto de partida x_0:

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ... = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Podemos decidir tomar sólo los primeros n términos, para tener una función polinómica de n grados alrededor del punto de partida $x_0 = 0$:

$$f(x) \approx \theta_0 + \theta_1 x + ... + \theta_n x^n$$

Sesgo de un estimador Consideremos ahora un modelo parametrizado con un único parámetro vectorial. Esto no es una limitación, sólo una elección didáctica:

$$p(X; \overline{\theta}) \in C$$

El objetivo de un proceso de aprendizaje es estimar el parámetro para, por ejemplo, maximizar la precisión de sus clasificaciones. Definimos el sesgo de un estimador en relación con un parámetro $\overline{\theta}$:

$$Bias[\overline{\theta}] = E_{x|\overline{\theta}}[\overline{\theta}] - \overline{\theta} \Rightarrow (\sum_{\overline{x}} \overline{\theta} p(\overline{\theta})) - \overline{\theta}$$

Varianza de un estimador Al principio hemos definido el proceso de generación de datos pdata, y hemos supuesto que nuestro conjunto de datos X se ha extraído de esta distribución. Sin embargo, no queremos aprender las relaciones existentes limitadas a X; esperamos que nuestro modelo sea capaz de generalizar correctamente a cualquier otro subconjunto extraído de p_{data}. Una buena medida de esta capacidad la proporciona la varianza del estimador:

$$Var[\theta] = Stderr[\overline{\theta}]^2 = E[(\overline{\theta} - E[\overline{\theta}])^2]$$

0.1.3 Ejemplo: Normalización

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

from sklearn.preprocessing import Normalizer
plt.style.use('dark_background')

# Definimos una semilla random para poderlo reproducir despues
np.random.seed(1000)


numero_muestras = 200
mu = [1.0, 1.0]
matriz_covarianza = [[2.0, 0.0], [0.0, 0.8]]


if __name__ == "__main__":
  # Creamos el conjunto de datos
  X = np.random.multivariate_normal(mean=mu, cov=matriz_covarianza,
                                    size=numero_muestras)

  # Realizar la normalizacion
  normalizador = Normalizer(norm='12')
  X_nz = normalizador.fit_transform(X)

  # Mostrar los resultados
  fig, ax = plt.subplots(figsize=(10, 10))
  ax.scatter(X_nz[:, 0], X_nz[:, 1], s=50)
  ax.set_xlim([-2, 2])
  ax.set_ylim([-2, 2])
  ax.set_xlabel(r'$x_0$', fontsize=16)
  ax.set_ylabel(r'$x_1$', fontsize=16)
  ax.set_title(r'Dataset normalizado ($L_2$ norm = 1)', fontsize=16)
  plt.show()

  # Computar un ejemplo de prueba
  X_test = [
```