Lista de ejercicios 4

Curso: Tópicos de Investigación : Machine Learning CM-072

Lecturas Importantes

1. Introducción al Machine Learning usando R:
   `www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/`.

---

1. Supóngase que se nos da la tarea de construir un sistema para distinguir correo basura. ¿Cómo se puede detectar en el equipo correo no deseado a través de un análisis sintáctico?. ¿Qué nos gustaría que el equipo haga si detecta un correo electrónico no deseado, que lo borre de forma automática, lo mueva a un archivo diferente, o simplemente lo resalte en la pantalla ?.

2. Digamos que se nos da la tarea de construir un taxi automático. Define las restricciones. ¿Cuáles son las entradas? ¿Cuál es la salida? ¿Cómo podemos comunicarnos con el pasajero? ¿Es necesario comunicarse con los otros taxis automáticos, es decir, es necesario un `lenguaje`?.

3. Si una imagen facial es una imagen de 100×100 , es decir un vector de dimensión 10000. Si desplazamos la imagen un píxel a la derecha, tenemos un vector muy diferente en el espacio de 10000 dimensiones. ¿Cómo podemos construir reconocedores faciales a tales distorsiones?.

4. En un diario, encontramos cinco muestras de informes de noticias para cada categoría como `política`, `deportes` y la `artes`. Si queremos encontrar palabras sobre estas muestras que se utilizan con frecuencia para cada categoría, las cuales pueden ayudarnos a discriminar entre las diferentes categorías. Por ejemplo, en un informe de noticias sobre política es probable que se incluyen palabras como `gobierno`, `recesión`, `Congreso`, y así sucesivamente, mientras que un informe de noticias sobre `artes` puede incluir `disco`, `lienzo`, o `teatro` . también hay palabras tales como `objetivo` que son ambiguas. ¿Cómo proceder en esos casos?.

5. Una moneda es lanzada n veces, mostrando caras H_n veces y sellos T_n veces. Sea $S_n = H_n - T_n$. Muestra que

   $$\mathbb{P}(S_n > an)^{1/n} \to \frac{1}{\sqrt{(1+a)^{1+a}(1-a)^{1-a}}} \text{ si } 0 < a < 1.$$

   ¿ Qué ocurre si $a \geq 1$?.

6. Sean X y Y variables aleatorias independientes con función densidad común $f(x) = 1$ si $0 < x < 1$, $f(x) = 0$ en otros casos.

   Sea $U = I_{\{Y \leq g(X)\}}$, la función indicador del evento que $Y \leq g(X)$ y sea $V = g(X)$, $W(X) = \frac{1}{2}\{g(X) + g(1-X)\}$.

   Muestra que $\mathbb{E}(U) = \mathbb{E}(V) = \mathbb{E}(W) = J$ y que $\mathbb{V}(W) \leq \mathbb{V}(V) \leq \mathbb{V}(U)$. Prueba que W es el estimador más eficiente de J.

7. Supongamos que tenemos una matriz cuadrada $M \in \mathbb{R}^{n \times n}$ es escrita en forma de bloques

   $$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

   donde $A \in \mathbb{R}^{k \times k}$.

- Muestra que podemos descomponer M como el producto

$$M = \begin{pmatrix} I & O \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & O \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ O & I \end{pmatrix}$$

donde I es la matriz identidad de tamaño apropiado.

- Supongamos que descomponemos $A = L_1U_1$ y $D - CA^{-1}B = L_2U_2$. Muestra como construir una factorización LU de M dados esas matrices adicionales.
- Usa esta estructura para definir un algoritmo recursivo para la factorización LU, se puede asumir que $n = 2^l$ para algún $l > 0$.

8. Una versión general de la descomposición de Cholesky, que no requiere el cálculo de las raices es la `descomposición LLT`

   - Supongamos que $A \in \mathbb{R}^{n\times n}$ es simétrica y admite una factorización LU (sin pivot). Muestra que A puede ser factorizada como $A = LDL^T$, donde D es diagonal y L es una matriz triangular.
   - Modifica la construcción de la descomposición de Cholesky para mostrar que una matriz simétrica, definida positiva A puede ser factorizada como $A = LDL^T$ sin usar alguna operación de raices cuadradas.

9. Explica y corrige el siguiente código que implementa el `perceptron`

```
import numpy as np

class Perceptron(object):

   def __init__(self, eta=0.01, n_iter=10):
      self.eta = eta
      self.n_iter = n_iter

   def fit(self, X, y):
      self.w_ = np.zeros(1 + X.shape[1])
      self.errors_ = []

      for _ in range(self.n_iter):
         errors = 0
     for xi, target in zip(X, y):
     update = self.eta * (target - self.predict(xi))
            self.w_[1:] += update * xi
            self.w_[0] += update
            errors += int(update != 0.0)
         self.errors_.append(errors)
       return self

   def net_input(self, X):
      return np.dot(X, self.w_[1:]) + self.w_[0]

   def predict(self, X):
      return np.where(self.net_input(X) >= 0.0, 1, -1)
```