Lista de ejercicios 2

Curso: Tópicos de Investigación : Machine Learning CM-072

Lecturas Importantes

1. Tutoriales para ciencia de datos y machine learning según kdnuggets: http://www.kdnuggets.com/2016/04/top-10-ipython-nb-tutorials.html
2. http://www.r2d3.us/visual-intro-to-machine-learning-part-1/ es un página que muestra de manera visual, los conceptos y técnicas más importantes del machine learning.

Python-Scipy-matplotlib y scikit-learn

1. Escribe una función que retorne la distancia entre dos puntos a y b, donde a y b son tuplas (x, y). Por ejemplo, si $a = (3, 0)$ y $b = (0, 4)$, la función debe retornar 5.
2. Crea una lista de los cubos de x para $xin[0, 10]$ usando
   - un bucle `for`.
   - una lista de comprensión.
   - una función `map`.
3. Encuentra el error en esta función que toma una lista de números y retorne una lista de números normalizados.

```
def f(ln):
    """Retorna una lista normalizada sumando  1."""
    s = 0
    for n in ln:
        s += n
    return [n/s for n in ln]
```

4. Explica el código sobre manipulación de matrices

```
import scipy

A = np.reshape(np.arange(1, 5), (2,2))
b = np.array([36, 88])
ans = scipy.linalg.solve(A, b)
P, L, U = scipy.linalg.lu(A)
Q, R = scipy.linalg.qr(A)
D, V = scipy.linalg.eig(A)
print ('ans =\n', ans, '\n')
print ('L =\n', L, '\n'
print ("U =\n", U, '\n'
print ("P = \nPermutacion Matrix\n", P, '\n'
print ('Q =\n', Q, '\n')
```

```
print ("R =\n", R, '\n')
print ('V =\n', V, '\n')
print ("D =\nDiagonal matrix\n", np.diag(abs(D)), '\n')
```

5. Reescribimos el bucle anidado como una lista de comprensión

```
ans = []
for i in range(3):
    for j in range(4):
        ans.append((i, j))
print (ans)
```

6. Escribe un decorador `hola` que hace que cada función imprima `Python`. Por ejemplo

```
@hola
def eje(x):
    return x*x
```

7. Reescribe lo siguiente como una lista por comprensión

```
lc = map(lambda x: x*x, filter(lambda x: x%2 == 0, range(5)))
print (lc)
```

8. Explica los siguientes códigos

```
def m_l(obj):
    def lg(data):
        with open(obj, 'a') as f:
            f.write(data + '\n')
    return lg
```

```
def f(x = []):
    x.append(1)
    return x

print (f())
print (f())
print (f())
print (f(x = [9,9,9]))
print (f())
print (f())
```

```
from functools import partial

sum_ = partial(reduce, op.add)
prod_ = partial(reduce, op.mul)
print (sum_([1,2,3,4]))
print (prod_([1,2,3,4]))
```

```
import scipy.stats as stats

def compare(x, y, func):
     return func(x, y)[1]
x, y = np.random.normal(0, 1, (100,2)).T

print ("p valores asumiendo igual varianza    =%.8f" % compare(x, y, stats.ttest_ind))
prueba = partial(stats.ttest_ind, equal_var=False)
print ("p valores sin asumir  igual varianza=%.8f" % compare(x, y, prueba))



def quick_sort1(xs):
    """quicksort -1"""

    # caso inicial
    if xs == []:
        return xs
    else:
        pivot = xs[0]
        menos_que = [x for x in xs[1:] if x <= pivot]
        mas_que = [x for x in xs[1:] if x > pivot]
        return quick_sort1(menos_que) + [pivot] + quick_sort1(mas_que)



def c_d(n):
    for i in range(n, 0, -1):
        yield i

c = c_d(10)
print (next(c))
print (next(c))
for c1 in c:
    print (c1)



def func_timer(func):
    """Tiempo que toma una funcion ."""

    def f(*args, **kwargs):
        import time
        i = time.time()
        r = func(*args, **kwargs)
        print ("Ha transcurrido: %.2fs" % (time.time() - i))
        return r

    return f

@func_timer
def s(msg, d=1.0):
    """se retrasa mientras responde"""
    import time
    time.sleep(d)
    print (msg)

s("Hola Python", 1.5)
```

9. Explica el siguiente código

```
from itertools import cycle, groupby, islice, permutations, combinations

print (list(islice(cycle('abcd'), 0, 10)))
lenguajes = sorted(['R', 'Python', 'scala', 'F++',
                    'javascript', 'Java', 'smalltalk', 'awk', 'perl'], key=len)
for k, g in groupby(lenguajes, key=len):
    print (k, list(g))
print ([''.join(p) for p in permutations('abc')])
print ([list(c) for c in combinations([1,2,3,4], r=2)])
```

10. Comparando complejidad. Explica lo siguiente

```
def f1(n, k):
    return k*n*n

def f2(n, k):
    return k*n*np.log(n)

n = np.arange(0, 20001)

plt.plot(n, f1(n, 1), c='blue')
plt.plot(n, f2(n, 1000), c='red')
plt.xlabel('Entrada(n)', fontsize=16)
plt.ylabel('Numero de operaciones ', fontsize=16)
plt.legend(['$\mathcal{O}(n^2)$', '$\mathcal{O}(n \log n)$'], loc='best', fontsize=20);
```

11. Determina si el siguiente sistema de ecuaciones, no tiene solución, infinitas soluciones o única solución `sin resolver el sistema de ecuaciones`

```
A = np.array([[1,2,-1,1,2],[3,-4,0,2,3],[0,2,1,0,4],[2,2,-3,2,0],[-2,6,-1,-1,-1]])
```

12. Calcula la descomposición LU de la siguiente matriz, de manera manual y usando Python

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -4 & 6 \\ 3 & -9 & -3 \end{pmatrix}$$

13. Calcula la descomposición de Choleski de la siguiente matriz, de manera usual y usando Python

$$\begin{pmatrix} 4 & 5 & 3 \\ 2 & 4 & 6 \\ 3 & 5 & 8 \end{pmatrix}$$

14. Escribe una función para resolver el sistema de ecuaciones $Ax = b$ usando la descomposición `SVD`. La función debe tomar como entrada `A` y `b` y retornar x.

15. Escribe una función para la covarianza

$$Cov(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n-1}.$$

Prueba la función construida tomando los valores `X = np.random.random(10)` y `Y = np.random.random(10)` y compara tus valores y el tiempo de ejecución con alguna función incorporada en Python.

16. Explica el siguiente código

```
mu = [0,0]
sigma = [[0.6,0.2],[0.2,0.2]]
n = 1000
x = np.random.multivariate_normal(mu, sigma, n).T

A = np.cov(x)
m = np.array([[1,2,3],[6,5,4]])
ms = m - m.mean(1).reshape(2,1)
np.dot(ms, ms.T)/2

e, v = np.linalg.eig(A)

plt.scatter(x[0,:], x[1,:], alpha=0.2)
for e_, v_ in zip(e, v.T):
    plt.plot([0, 3*e_*v_[0]], [0, 3*e_*v_[1]], 'r-', lw=2)
plt.axis([-3,3,-3,3])
plt.title('Escribe ....');
```

17. Encuentra los autovalores y autovectores de la matriz covarianza del conjunto de datos, usando la descomposición espectral.

18. Encuentra los autovalores y autovectores de la matriz covarianza del conjunto de datos, usando SVD y verifica si son equivalentes a los encontrados usando la descomposición espectral.

19. Usa la función `decomposition.PCA()` desde `sklearn` para llevar a cabo la descomposición. Representa el conjunto de puntos de forma que sea muestre dos filas de 3 gráficos de dispersión cada uno, donde las columnas muestren las proyecciones $(0,1), (0,2), (1,2)$.

20. Explica el siguiente código

```
----------------------------------
Primer paso

def f(x):
    return x**3 + 4*x**2 -6

x = np.linspace(-3.1, 0, 100)
plt.plot(x, x**3 + 4*x**2 -6)

a = -3.0
b = -0.5
c = 0.5*(a+b)

plt.text(a,-1,"a")
plt.text(b,-1,"b")
plt.text(c,-1,"c")

plt.scatter([a,b,c], [f(a), f(b),f(c)], s=50, facecolors='none')
plt.scatter([a,b,c], [0,0,0], s=50, c='red')

xaxis = plt.axhline(0);

-----------------------------
Siguiente paso
```

```
x = np.linspace(-3.1, 0, 100)
plt.plot(x, x**3 + 4*x**2 -6)

d = 0.5*(b+c)

plt.text(d,-1,"d")
plt.text(b,-1,"b")
plt.text(c,-1,"c")

plt.scatter([d,b,c], [f(d), f(b),f(c)], s=50, facecolors='none')
plt.scatter([d,b,c], [0,0,0], s=50, c='red')

xaxis = plt.axhline(0);
```

21. Revisar los siguientes códigos que usa la librería scikit-learn

```
from sklearn import metrics
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X, y)
print(model)

# hacer predicciones

esperado = y
predice = model.predict(X)

# resumen del modelo fijado

print(metrics.classification_report(esperado, predice))
print(metrics.confusion_matrix(esperado, predice))
```

```
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
model.fit(X, y)
print(model)

# hacer predicciones

esperado = y
predice = model.predict(X)

# resumen del modelo fijado

print(metrics.classification_report(esperado, predice))
print(metrics.confusion_matrix(esperado, predice))
```

```
from sklearn import metrics
```

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier()
model.fit(X, y)
print(model)

# hacer predicciones

esperado = y
predice = model.predict(X)

# resumen del modelo fijado

print(metrics.classification_report(esperado, predie))
print(metrics.confusion_matrix(esperado, predice))



from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X, y)
print(model)

# hacer predicciones

esperado = y
predice = model.predict(X)

# resumen del modelo fijado

print(metrics.classification_report(esperado, predie))
print(metrics.confusion_matrix(esperado, predice))



from sklearn import metrics
from sklearn.svm  import SVC

model = SVC()
model.fit(X, y)
print(model)

# hacer predicciones

esperado = y
predice = model.predict(X)

# resumen del modelo fijado

print(metrics.classification_report(esperado, predie))
print(metrics.confusion_matrix(esperado, predice))
```