REDES NEURONALES

Carlos A. Espinoza Garcia
Universidad Autónoma de Baja California
Instituto de Ingeniería
Carlos.espinoza7@uabc.edu.mx

ABSTRACT

Minería de datos es un conjunto de técnicas para encontrar información o manipularla así, logrando o exponiendo un entendimiento tanto como el que analiza o el que visualiza.
A medida que pasa el tiempo la información a nuestro alrededor va incrementando a pasos agigantados por lo cual es necesario utilizar nuevas técnicas o algoritmos más específicos según se requiera el problema.
Análisis discrimínate lineal es una popular técnico estadística para el reconocimiento de patrones, sin embargo este presenta problemas cuando los datos tienen una dimensionalidad alta, cada clase de este es representado por una matriz de covarianza y falla si las densidades se encuentran en su estado general. También se debe mencionar las diferencias que existen con el discriminante lineal cuadrático.

1. INTRODUCCION

El Análisis Discriminante Lineal (LDA) es un método de clasificación supervisado donde intervienen variables cualitativas, donde dos o más grupos son conocidos a priori, y además, en uno de ellos, se clasifican nuevas observaciones en función de sus características; mediante el uso del Teorema de Bayes. En este análisis, se estima la probabilidad de que una observación, a partir del valor de los predictores, sea parte de cada una de las clases de la variable cualitativa, P(Y=k|X=x). Asignando la observación a la clase k, para la que la probabilidad predicha es mayor.
Presenta las siguientes ventajas en contraposición con la Regresión logística: Cuando las clases están muy separadas, en la Regresión logística los parámetros estimados se vuelven inestables, mientras que esto no sucede en LDA.
Lo mismo sucede cuando si el número de observaciones es bajo y la distribución de los predictores es aproximadamente normal en cada clase, en LDA hay mayor estabilidad. De acuerdo al Teorema de Bayes:

$$f_k(X) = P(Y = k|X = x) = \frac{1}{\sqrt{2\pi\sigma_k}}\ exp(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2)$$

Esta fórmula se puede simplificar mediante transformación logarítmica:

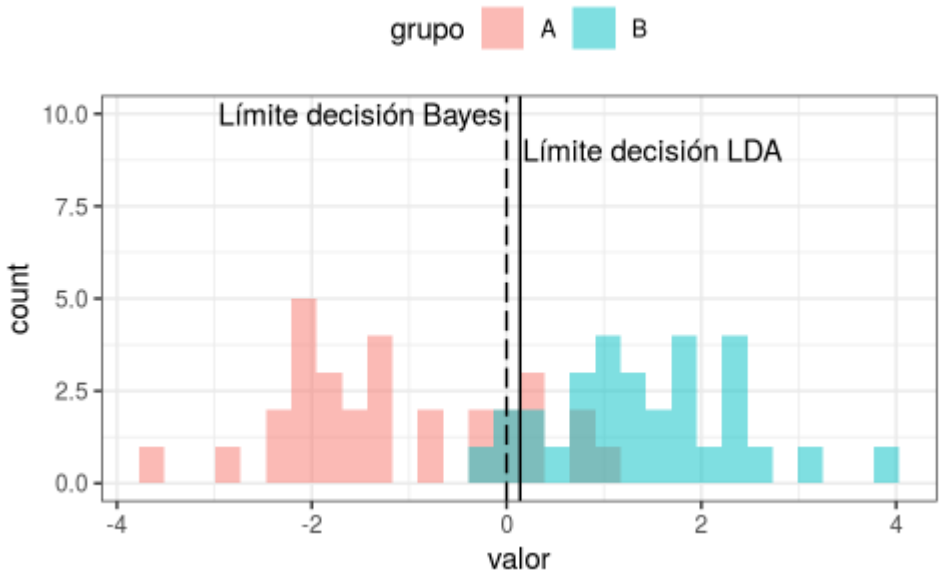
$$\hat{\delta}_k(x) = log(P(Y = k|X = x)) = x\,\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

Las condiciones que se deben cumplir para que un Análisis Discriminante Lineal sea válido son:

- Cada predictor que forma parte del modelo se distribuye de forma normal en cada una de las clases de la variable respuesta. En el caso de múltiples predictores, las observaciones siguen una distribución normal multivariante en todas las clases.
- La varianza del predictor es igual en todas las clases de la variable respuesta. En el caso de múltiples predictores, la matriz de covarianza es igual en todas las clases. Si esto no se cumple se recurre a Análisis Discriminante Cuadrático (*QDA*).

Cuando la condición de normalidad no se cumple, el LDA pierde precisión, pero aun así puede llegar a clasificaciones relativamente buenas. *Using discriminant analysis for multi-class classification: an experimental investigation (Tao Li, Shenghuo Zhu, Mitsunori Ogihara).*

Ejemplo:

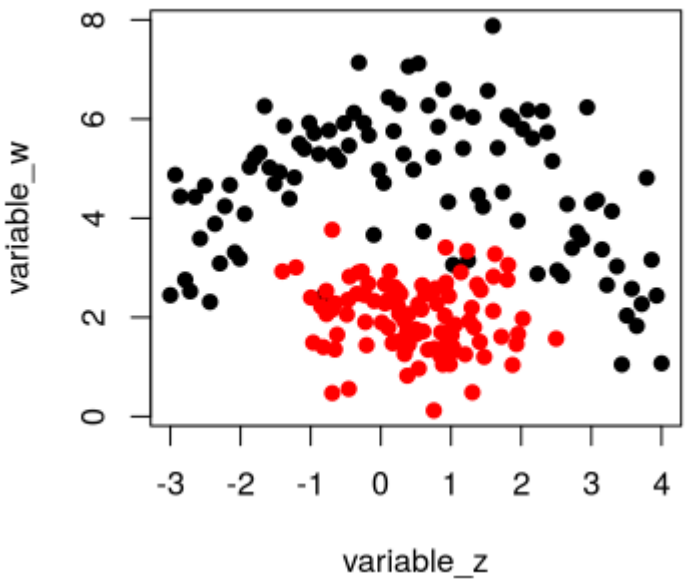


El clasificador cuadrático o *Quadratic Discriminat Analysis QDA* se asemeja en gran medida al *LDA*, con la única diferencia de que el *QDA* considera que cada clase *k* tiene su propia matriz de covarianza (∑k)(∑k) y, como consecuencia, la función discriminante toma forma cuadrática:

$$log(P(Y = k|X = x)) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log(\pi_k)$$

Para poder calcular la *posterior probability* a partir de esta ecuación discriminante es necesario estimar, para cada clase, (∑k)(∑k), μkμk y πkπk a partir de la muestra. Cada nueva observación se clasifica en aquella clase para la que el valor de la *posterior probability* sea mayor.
QDA genera límites de decisión curvos por lo que puede aplicarse a situaciones en las que la separación entre grupos no es lineal.

Ejemplo:



2. DESARROLLO

- Se tienen como datos de entrada un archivo el cual se carga para el estudio de las técnicas QDA como LDA.

C_train - Arreglo de NumPy

	0	1	2	3	4
0	17.99	10.38	122.8	1001	0.1184
1	20.57	17.77	132.9	1326	0.08474
2	19.69	21.25	130	1203	0.1096
3	11.42	20.38	77.58	386.1	0.1425
4	20.29	14.34	135.1	1297	0.1003
5	12.45	15.7	82.57	477.1	0.1278
6	18.25	19.98	119.6	1040	0.09463
7	13.71	20.83	90.2	577.9	0.1189
8	13	21.82	87.5	519.8	0.1273

- Se utiliza un segmento de datos como training y otro test el cual corresponde con el 80 y 20 porciento.\
- Separamos los datos en clases con sus respectivas etiquetas.
- Obtenemos las matrices de covarianza de los datos de entrenamiento

cov_xc - Arreglo de NumPy

	0	1	2	3	4	5
0	12.2579	5.67246	84.2596	1204.24	0.00863685	0.0958798
1	5.67246	16.6689	39.7749	549.594	0.00362642	0.0683228
2	84.2596	39.7749	581.868	8285.31	0.0722255	0.729335
3	1204.24	549.594	8285.31	120956	0.895496	9.35217
4	0.00863685	0.00362642	0.0722255	0.895496	0.000187741	0.000482744
5	0.0958798	0.0683228	0.729335	9.35217	0.000482744	0.00290169
6	0.185644	0.118371	1.36001	18.6585	0.000608676	0.00388588
7	0.111823	0.0589486	0.798942	11.1082	0.000308928	0.00176747
8	0.0140898	0.01307	0.122017	1.39704	0.000214199	0.0009158
9	0.001	0.001	0.001	0.001	5.37778e-05	0.000212793

- Saneamos la muestra anterior el cual es evaluada por la siguiente función:

```
def sanear(datos):
    D, V = LA.eig(datos)
    datos[datos<0] =.001
    tras = np.transpose(V)
    op = (D*V)
    ops = np.dot(op,tras)
    return ops
```

- Posteriormente se requiere hacer los cálculos de los datos con las técnicas estadísticas LDA y QDA, tomando en cuenta la longitud del test. Para este apartado es necesario calcular inversas, medias y clases a priori.

```
QDC1 = []
QDC2 = []
LDC1 = []
LDC2 = []

for i in range(len(C_test)):

    Test = C_test[i, :]
    sumaT1 = QDC(Test,inv_san1,med_clas1,san1,priori1)
    QDC1.append(sumaT1)
    sumaT2 = QDC(Test,inv_san2,med_clas2,san2,priori2)
    QDC2.append(sumaT2)
    sumaL1 = LDC(Test,med_clas1,mios_t,priori1)
    LDC1.append(sumaL1)
    sumaL2 = LDC(Test,med_clas2,mios_t,priori2)
    LDC2.append(sumaL2)
```

- Estos valores son evaluados por en dos vectores los cuales tendrán que ser comparados por casos test. Antes existe la regla de compararlos entre si para asignarles valores correspondientes a la clase.

```
def mod_class(mod1,mod2):
    mod = []
    for j in range(len(mod1)):
        if mod1[j]> mod2[j]:
            mod.append(1)
        else:
            mod.append(2)

    return mod
```

EL SIGUIENTE PASO ES CALCULAR LA TASA DE ACEPTACIÓN.

QDC tasa de reconocimiento

85.96491228070175

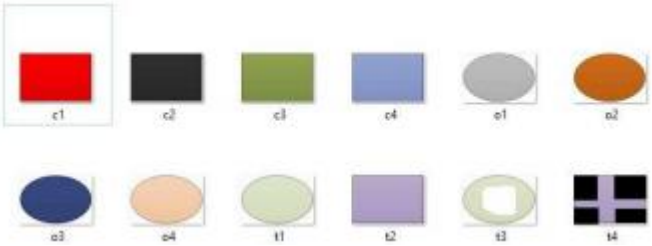
LDC tasa de reconocimiento

97.36842105263158

DESARROLLO 2

Se tiene como fuente de datos 8 imágenes de training y 4 de test, el cual tiene que reconocerse los patrones para adivinar que figura esta entre los casos de estudio.

Trainig



Test

Los datos antes mencionados tienen que se extraidos en forma de matrices antes de ser analizados por lo cual es necesario transformar su escala de colores, existen varias formas, RGB, escala de grises u opacos, para este estudio se realizo con blanco y negro ya que a simple vista los datos tienen contornos y los brillos pueden provocar dificultad en el modelo propuesto.

La siguiente funcionalidad nos crea el escenario descrito:

```
newl = []
newt = []
clases = [0,0,0,0,1,1,1,1]
for name in glob.glob('circulos_cuadrados/*.jpg'):
    print(name)
    im = Image.open(name)
    thresh = 200
    fn = lambda x : 1 if x > thresh else 0
    im = im.convert('L').point(fn)

    tama = np.size(im)
    dimen = tama[0]*tama[1]
    data = np.asarray(im)
    newv = np.matrix.flatten(data,'C')
    newl.append(newv)


        #im.show()


test = 'circulos_cuadrados/test/*.jpg'
v =3
for name in glob.glob(test):
    print(name)
    im = Image.open(name)
    thresh = 200
    fn = lambda x : 1 if x > thresh else 0
    im = im.convert('L').point(fn)

    tama = np.size(im)
    dimen = tama[0]*tama[1]
    data = np.asarray(im)
    newv = np.matrix.flatten(data,'C')

    newt.append(newv)
```

- es necesario tomar las características principales de las matrices resultantes

newt - Arreglo de NumPy

	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0	0

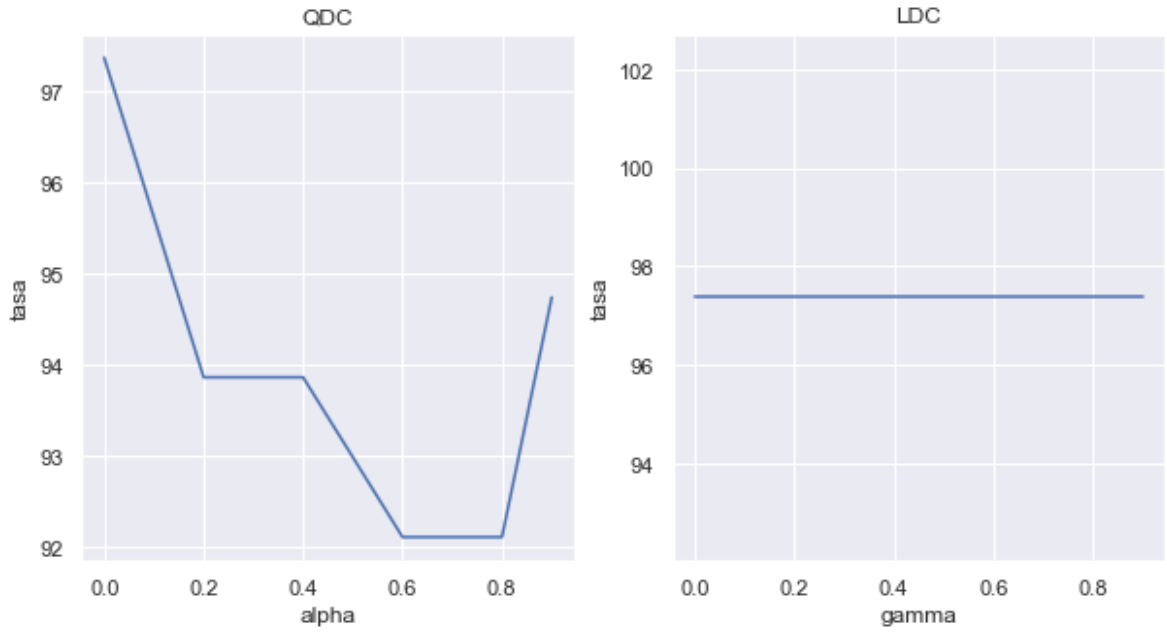
```
pca = PCA(n_components=8)
pca = pca.fit(newl)
cosa =pca.components_
```

- se tienen que calcular las variables para ingresar a LDA:

```
m1 = medias(datos_c1)
m2 = medias(datos_c2)


LDC1 = []
LDC2 = []

sigma = np.cov(new1)
saneado = sanear(sigma)
saneado = np.linalg.inv(saneado)
c, r = class1.shape
c2, r2 = class2.shape
p_clas1 = r/len(clases)
p_clas2 = r2/len(clases)


for i in range(mult2.shape[1]):

    Test = mult2[:, i]
    sumaL1 = LDC(Test,m1,saneado,p_clas1)
    LDC1.append(sumaL1)
    sumaL2 = LDC(Test,m2,saneado,p_clas2)
    LDC2.append(sumaL2)

LDC_T = mod_class(LDC1,LDC2)
tasa_LDC = tasas(LDC_T,clases)*v
```

RESULTADOS:

Para la primera parte tenemos que QDA y LDC :



Y LAS FIGURAS SE TIENE EL SIGUIENTE PORCENTAJE:

circulos_cuadrados\c1.jpg
circulos_cuadrados\c2.jpg
circulos_cuadrados\c3.jpg
circulos_cuadrados\c4.jpg
circulos_cuadrados\o1.jpg
circulos_cuadrados\o2.jpg
circulos_cuadrados\o3.jpg
circulos_cuadrados\o4.jpg
circulos_cuadrados/test\t1.jpg
circulos_cuadrados/test\t2.jpg
circulos_cuadrados/test\t3.jpg
circulos_cuadrados/test\t4.jpg

Tasa de reconocimiento
75.0

CONCLUSIÓN

Existen diversas técnicas estadísticas para encontrar patrones dentro del mundo de la minería de datos, lo esencial es entender el problema y ejecutarlo con las herramientas que se tienen. Es importante entender los niveles de tratamiento de datos, ya que estos serán de mucha ayuda al estar operándolos. En nuestro caso de estudio fue necesario tomar otras herramientas, tales como el saneamiento de matrices. PCA nos apoyó para la dimensión de datos.

QDA y LDA son poderosos para la discriminación de datos, pero integrarlo con otras su técnicas pueden dar resultados favorables.