Práctica 2. Clasificador Bayesiano

Andrés González Flores, Facultad de Ingeniería, UNAM.

—————————— ◆ ——————————

# 1 Objetivos

Clasificar imágenes con 2, 3 o 4 regiones utilizando el clasificador de Bayes

# 2 Introducción

El clasificador Bayesiano es un método de clasificación supervisada que consiste en asignar a un objeto descrito por un conjunto de atributos o características, , a una de *m* clases posibles, , tal que la probabilidad de la clase dados los atributos se maximiza:



La formulación de este clasificador se basa en utilizar la regla de Bayes para calcular la probabilidad a posteriori de la clase dados los atributos

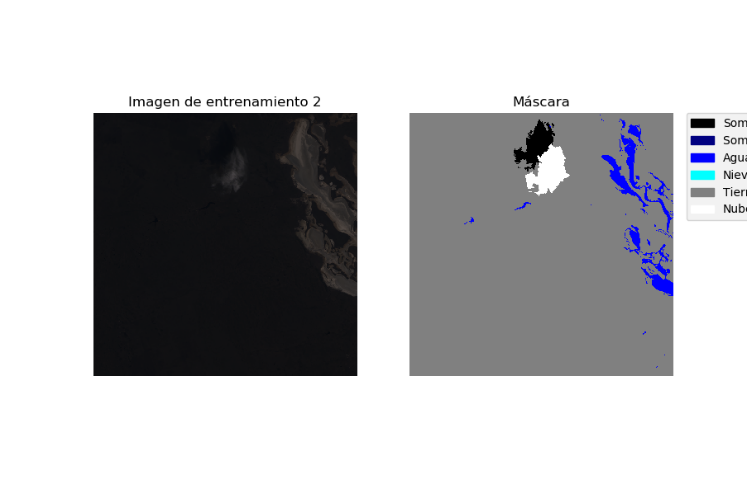
Dado que el denominador no varía para las diferentes clases, se puede considerar como una constante si lo que interesa es maximizar la probabilidad de la clase.



El clasificador Bayesiano ingenuo (naive Bayes) se basa en la suposición de que todos los atributos son independientes dada la clase, esto es, cada atributo  es condicionalmente independiente de los demás atributos dada la clase

Bajo estas consideraciones, la ecuación 2 se puede escribir como:

Donde . se puede considerar como una constante de normalización.



# 3 Desarrollo

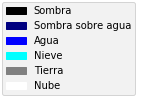
1. Seleccionar imágenes con zonas aledañas a clasificar.

Escogí trabajar con los datos de validación de SPARCS de M. Joseph Huges de la Universidad del Estado de Oregon. Estos datos se usaron para la detección automática de nubes usando redes neuronales.

Los datos consisten en 80 subconjuntos de escenas del

Landsat 8 de 1000x1000px en formato .tiff. Cada escena tiene 10 bandas. Estos datos ya venían con sus respectivas máscaras de segmentación. Las máscaras venían segmentadas en 6 clases que se interpretan como sigue:

Valor / Interpretación

0 Sombra

1 Sombra Sobre Agua

2 Agua

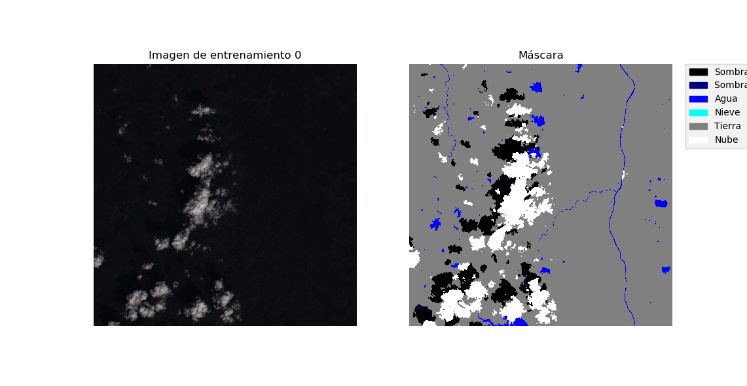
3 Nieve

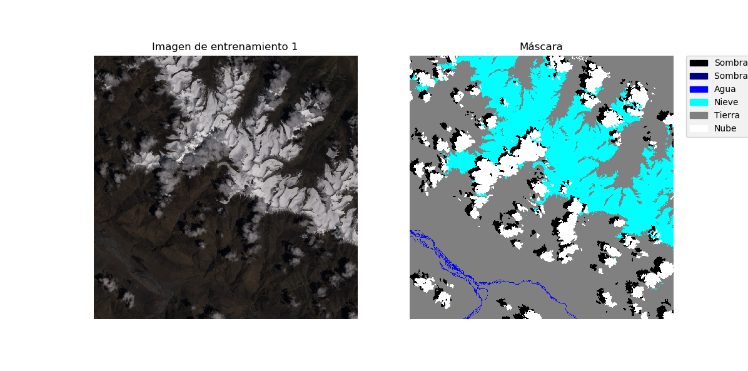
4 Tierra

5 Nube

6 Inundado

Para que la carga de cómputo no resultara muy pesada, seleccioné sólo 5 bandas (de la 1 a la 5) de 3 escenas. Esto resultó en datos de tamaño 3x1000x1000x5 (un total de 15,000,000‬ de datos simples).

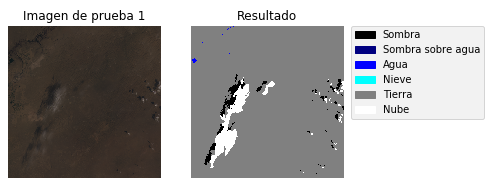
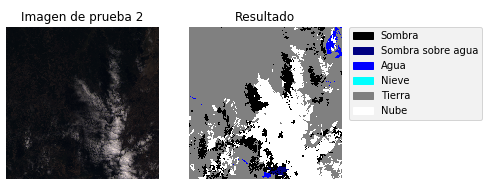
Las imágenes RGB se encuentran en las bandas 2-3, siendo B la banda 2, G la banda 3 y R la banda 4. A continuación, muestro las imágenes de entrenamiento.



Con los datos de las 5 bandas de cada escena, calculé las probabilidades a priori de cada una de las regiones. Para estas imágenes, sólo tomé en cuenta las clases 0 a 5.



Luego, calculé los vectores de medias de cada clase.



Luego, se calcularon las matrices de covarianzas Σk con la siguiente fórmula.





Con esto, tenemos todos los datos necesarios calculados para presentarle al clasificador, escenas de Landsat nuevas.

La función discriminante de Bayes está definida como:

Esto lo calculamos para cada dato *x* en cada clase. El valor máximo indicará la clase a la que pertenece ese vector.

# 4 Resultados

# 5 Código

funciones.py

1. **import** numpy as np
3. **def** obtenerProbAPriori(datos\_y, n\_clases : int):
4. data\_shape = datos\_y.shape
5. total\_datos = np.multiply.reduce(data\_shape)
6. hist = np.histogram(datos\_y.reshape(1, total\_datos),
7. bins=n\_clases,
8. range=[0,n\_clases-1])
9. prob = hist[0].astype(np.long) / total\_datos
10. **return** prob
12. **def** obtenerMatricesMedias(datos\_x, datos\_y, clases, prob\_clases):
13. datos\_shape = datos\_x.shape
14. total\_datos = np.multiply.reduce(datos\_shape)
15. medias = np.zeros(shape=(len(clases), datos\_x.shape[3]))
16. **for** i\_img **in** range(datos\_shape[0]):
17. **for** x **in** range(datos\_shape[1]):
18. **for** y **in** range(datos\_shape[2]):
19. k = datos\_y[i\_img, x, y]
20. medias[k]+=datos\_x[i\_img, x, y]
21. **for** k, clase **in** enumerate(clases):
22. medias[k]/=(prob\_clases[k]\*total\_datos)
23. **return** medias
25. **def** obtenerMatricesCovarianzas(datos\_x, datos\_y, medias,clases, prob\_clases):
26. datos\_shape = datos\_x.shape
27. total\_datos = np.multiply.reduce(datos\_shape)
28. covarianzas = np.zeros(shape=(len(clases), datos\_shape[3], datos\_shape[3]))
29. **for** i\_img **in** range(datos\_shape[0]):
30. **for** x **in** range(datos\_shape[1]):
31. **for** y **in** range(datos\_shape[2]):
32. k = datos\_y[i\_img, x, y]
33. m\_temp = np.matrix(datos\_x[i\_img, x, y]-medias[k])
34. covarianzas[k]+=np.matmul(m\_temp.T, m\_temp)
35. **for** k, clase **in** enumerate(clases):
36. covarianzas[k]/=(prob\_clases[k]\*total\_datos)
37. **return** covarianzas
39. **def** disc\_bayes(x, m, SI, detS, Pk):
40. temp = np.matrix(x-m)
41. SI = np.matrix(SI)
42. temp2 = np.matmul(temp, SI)
43. temp3 = np.matmul(temp2, temp.T).item()
44. disc = -(1.0/2.0)\*temp3-(1.0/2.0)\*np.log(detS)+np.log(Pk)
45. **return** disc.item()
47. **def** predecir(datos\_x, modelo):
48. datos\_shape = datos\_x.shape
49. prediccion\_y = np.empty\_like(datos\_x[:,:,0]).astype(np.uint8)
50. discr = np.empty(len(modelo['clases'])).astype(np.float64)
51. **for** i **in** range(datos\_shape[0]):
52. **for** j **in** range(datos\_shape[1]):
53. **for** k **in** modelo['clases']:
54. x=datos\_x[i,j]
55. m=modelo['medias'][k]
56. S=modelo['covarianzas'][k]
57. SI=modelo['covs\_inv'][k]
58. detS=modelo['det\_covs'][k]
59. pk=modelo['prob\_a\_priori'][k]
60. discr[k] = disc\_bayes(x, m, SI, detS, pk)
61. prediccion\_y[i,j] = discr.argmax()
62. **return** prediccion\_y

entrenamiento.py

1. # -\*- coding: utf-
2. **import** rasterio
3. **import** numpy as np
4. **import** glob
5. **import** scipy.io
6. **from** funciones **import** obtenerProbAPriori, obtenerMatricesMedias, obtenerMatricesCovarianzas
7. # https://www.usgs.gov/land-resources/nli/landsat/spatial-procedures-automated-removal-cloud-and-shadow-sparcs-validation
8. path\_datos\_x = glob.glob('./Dataset/Train/\*data.tif')
9. datos\_x  = np.array([np.dstack(np.array(rasterio.open(path).read())) **for** path **in** path\_datos\_x])
10. datos\_x = datos\_x[:,:,:,0:5]/ 65536.0
11. path\_datos\_y = glob.glob('./Dataset/Train/\*mask.mat')
12. datos\_y = np.array([scipy.io.loadmat(path)['img'] **for** path **in** in path\_datos\_y])
13. clases = {
14. 0 : 'Sombra',
15. 1 : 'Sombra sobre agua',
16. 2 : 'Agua',
17. 3 : 'Nieve',
18. 4 : 'Tierra',
19. 5 : 'Nube',
20. #6 : 'Inundado',
21. }
22. n\_clases = len(clases)
23. probabilidades = obtenerProbAPriori(datos\_y,
24. medias = obtenerMatricesMedias(datos\_x, datos\_y, clases, probabilidades)
25. covarianzas = obtenerMatricesCovarianzas(datos\_x, datos\_y, medias, clases, probabilidades)
26. det\_covs = np.array([np.linalg.det(covarianzas[k]) **for** k **in** range(n\_clases)])
27. covs\_inv = np.zeros\_like(covarianzas)
28. **for** k **in** range(n\_clases):
29. covs\_inv[k] = np.linalg.inv(np.matrix(covarianzas[k]))
30. modelo = {
31. 'clases' : clases,
32. 'prob\_a\_priori' : probabilidades,
33. 'medias' : medias,
34. 'covarianzas' : covarianzas,
35. 'det\_covs' : det\_covs,
36. 'covs\_inv' : covs\_inv
37. }
39. np.save('./modelo', modelo)

prueba.py

1. **import** rasterio
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. **from** matplotlib.colors **import** ListedColormap
5. **import** matplotlib.patches as mpatches
6. **import** glob
7. **from** funciones **import** predecir
9. path\_datos\_prueba = glob.glob('./Dataset/Test/\*data.tif')
10. datos\_prueba = np.array([np.dstack(np.array(rasterio.open(path).read())) **for** path **in** path\_datos\_prueba])
11. datos\_prueba = datos\_prueba[:,:,:,0:5]/65536.0
12. imgs = np.flip(datos\_prueba[:,:,:,1:4], 3)
13. modelo = np.load('./modelo.npy', allow\_pickle=True).
14. pred\_y = np.zeros\_like(datos\_prueba[:,:,:,0]).astype(np.uint8)
15. **for** i, datos\_x **in** enumerate(datos\_prueba):
16. pred\_y[i] = predecir(datos\_x, modelo)
18. np.save('./Dataset/test/predicciones', pred\_y)
19. clases = modelo['clases']
20. n\_clases = len(clases)
21. sombra = np.array([0., 0., 0., 1.])
22. sombra\_sa = np.array([0., 0., 0.5, 1.])
23. agua = np.array([0., 0., 1., 1.])
24. nieve = np.array([0., 1., 1., 1.])
25. tierra = np.array([0.5, 0.5, 0.5, 1.])
26. nube = np.array([1., 1., 1., 1.])
27. nuevos\_colores = np.zeros(shape=(n\_clases, 4))
28. nuevos\_colores[0]=sombra
29. nuevos\_colores[1]=sombra\_sa
30. nuevos\_colores[2]=agua
31. nuevos\_colores[3]=nieve
32. nuevos\_colores[4]=tierra
33. nuevos\_colores[5]=nube
34. custom\_cm = ListedColormap(nuevos\_colores)
36. **for** i, img **in** enumerate(imgs):
37. fig, axs = plt.subplots(1,2)
38. axs[0].imshow(imgs[i])
39. imsh = axs[1].imshow(pred\_y[i], cmap=custom\_cm, vmin=0, vmax=6)
40. axs[0].axis('off')
41. axs[1].axis('off')
42. axs[0].set\_title('Imagen %d' %i)
43. axs[1].set\_title('Resultados')
44. patches = [ mpatches.Patch(color=nuevos\_colores[i], label=clases[i]) **for** i **in** range(len(clases)) ]
45. plt.legend(handles=patches, bbox\_to\_anchor=(1.05, 1),
46. loc=2, borderaxespad=0.,
47. facecolor = '010001')
48. plt.show()
50. **for** i, path **in** enumerate(path\_datos\_prueba):
51. plt.imsave(path[:-8]+'pred.png', pred\_y[i], cmap=custom\_cm, vmin=0, vmax=6)

# 6 Conclusiones

Con el desarrollo de esta práctica pude observar de manera práctica el comportamiento del clasificador Bayesiano y su utilidad en el área de reconocimiento de patrones.

Los resultados que obtuve no fueron tan precisos, en especial para distinguir cuerpos de agua. Es posible que al ajustar los datos de entrada para tomar un vector de características diferente, pueda tener mejores resultados.

**Referencias**

1. U.S. Geological Survey, 2016. L8 SPARCS Cloud Validation Masks. U.S. Geological Survey data release. doi:10.5066/F7FB5146.
2. Hughes M.J. & Hayes, D.J. (2014). Automated detection of cloud and cloud shadow in single-date Landsat imagery using neural networks and spatial post-processing. Remote Sensing, 6(6), 4907–4926. doi:10.3390/rs6064907.W.-K. Chen, *Linear Networks and Systems.* Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)
3. Sucar, Luis Enrique. *Clasificadores Bayesianos: de Datos a Conceptos.*
4. Colaboradores de Wikipedia. (2019, 26 septiembre). *Clasificador bayesiano ingenuo - Wikipedia, la enciclopedia libre*. Recuperado 30 septiembre, 2019, de https://es.wikipedia.org/wiki/Clasificador\_bayesiano\_ingenuo