



UNIVERSIDAD DE BUENOS AIRES

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

## Reconocimiento de Patrones

### Trabajo Práctico 1

2do. cuatrimestre 2015

Integrante	LU	Correo electrónico
Rey, Martín	483/12	marto.rey2006@gmail.com
Garassino, Agustín	394/12	ajgarassino@gmail.com

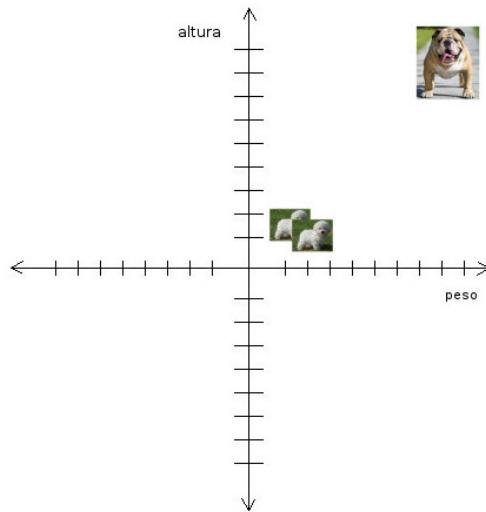
# Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Clasificación: features y featurespace	1
1.2. Método simple para clasificación	1
1.3. Gaussianas bivaluadas y regiones	2
<b>2. Clasificación de imágenes sintéticas</b>	<b>2</b>
2.1. Verdad terrestre	2
2.2. Creación de imágenes sintéticas	3
2.3. Modelo	3
2.4. Resultados	3
2.4.1. Imágenes sintéticas vs imágenes clasificadas	3
2.4.2. Matrices de confusión	4
2.5. Conclusión	5
<b>3. Clasificación de imágenes reales</b>	<b>6</b>
3.1. Estimando distribuciones	6
3.2. Clasificación de la imagen	6
3.3. Matrices de confusión	6
3.4. Conclusión	7

# 1. Introducción

## 1.1. Clasificación: features y featurespace

Un problema interesante en las ciencias de la computación es la clasificación automatizada de datos. Por ejemplo, teniendo mediciones de alturas y pesos de distintos perros y sabiendo a qué raza pertenecen, es posible entrenar a un programa de computadora que determine a qué raza pertenece un perro que no se encuentre en la población utilizada para que el programa aprenda. En este caso estamos abstrayendo a un objeto de la vida real, un perro, en dos características o *features* que nos interesan para diferenciarlo de los demás: su peso y su altura. Estas características cuantificables, pueden ser representadas como puntos en un espacio o *featurespace* bidimensional. Un perro pesado y alto entonces estará lejos de nuestro centro de coordenadas, mientras que uno delgado y bajo estará más cerca. De la misma manera, para cualquier conjunto de objetos, podemos seleccionar características deseables y medirlas, modelando a nuestra población como un conjunto de puntos en un espacio.



Una población de 3 perros representada en un *featurespace*. En este caso conocemos la raza de los perros de antemano, por lo tanto podemos usarlo para entrenar un programa. Luego conociendo sólo la altura y el peso para un perro, podríamos determinar con cierta confianza a cuál de las dos razas pertenece.

## 1.2. Método simple para clasificación

Si consideramos que conocemos la forma en la que las características se distribuyen en nuestra población para cada clase, entonces podemos realizar métodos simples para la clasificación de los mismos. Supongamos por ejemplo que conocemos que los caniches típicamente son delgados y de baja estatura. Entonces podemos decir que si un perro pertenece a esa raza, tiene una probabilidad muy alta de estar cerca del origen de coordenadas. Podemos asumir que el peso y la altura de los caniche siguen una distribución normal bidimensional, centrada cerca del origen y con poca varianza. Podemos hacer un razonamiento análogo para los bulldog. Una vez que escogimos nuestras distribuciones, sin tener siquiera la necesidad de entrenar a nuestro programa, podemos comenzar a clasificar perros. Lo único que tenemos que hacer es utilizar el teorema de Bayes:

$$P(\text{caniche} \mid \text{medición de altura y peso}) = \frac{P(\text{medición de altura y peso} \mid \text{caniche}) P(\text{caniche})}{P(\text{medición de altura y peso})}$$

$$P(\text{bulldog} \mid \text{medición de altura y peso}) = \frac{P(\text{medición de altura y peso} \mid \text{bulldog}) P(\text{bulldog})}{P(\text{medición de altura y peso})}$$

Para determinar si es un bulldog o un caniche, podemos tomar aquél que sea más probable de ambos. Podemos notar que el término en el denominador no es necesario computarlo, dado que aparece en ambas expresiones. Además podemos ver que aparecen los términos  $P(\text{caniche})$  y  $P(\text{bulldog})$ , estas probabilidades son denominadas *a priori*, y tienen que ver con la probabilidad de que una muestra aleatoria de la población sea

de la raza esa, sin tener ningún dato adicional. Cada punto en nuestro *featurespace* entonces será clasificado de manera determinística en una de las dos clases, separandolo en dos regiones posibles. Para el caso en donde tenemos  $k$  clases, el espacio quedará dividido en  $k$  regiones distintas.

### 1.3. Gaussianas bivaluadas y regiones

Muchas de las características de los fenómenos que encontramos en la naturaleza tienen una distribución normal. Se puede demostrar que cuando se poseen dos clases distribuidas normalmente, la frontera que separa la clasificación entre ambas puede ser descripta como una cónica. A continuación algunos gráficos de gaussianas bivaluadas y las fronteras que generan:

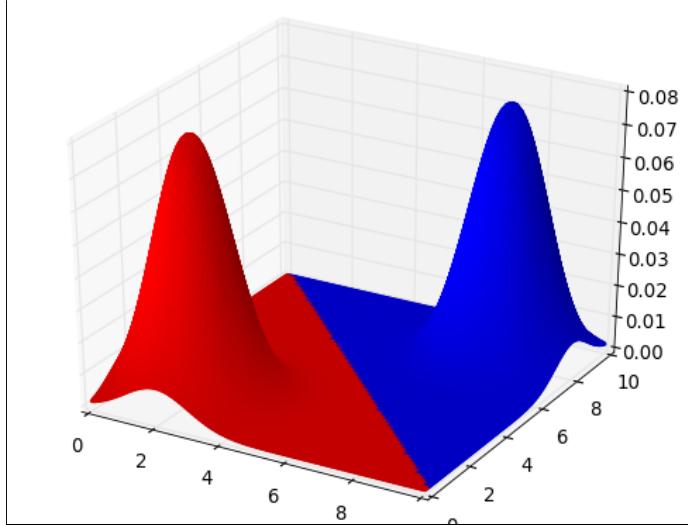


Figura 1: Ejemplo de frontera lineal.

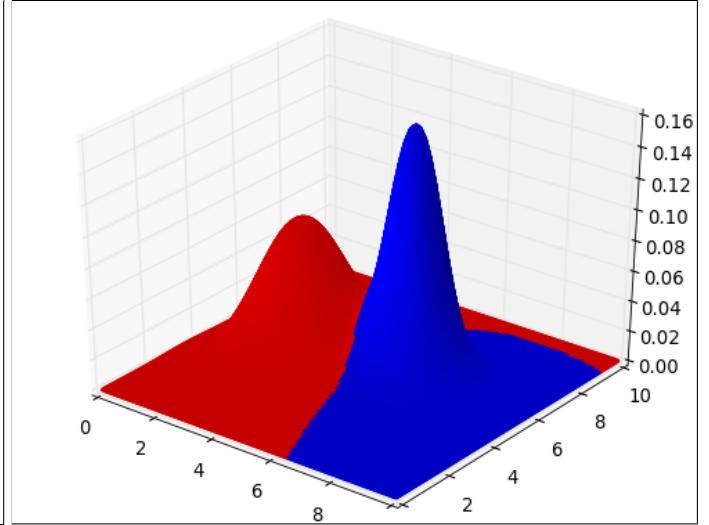


Figura 2: Ejemplo de frontera elíptica.

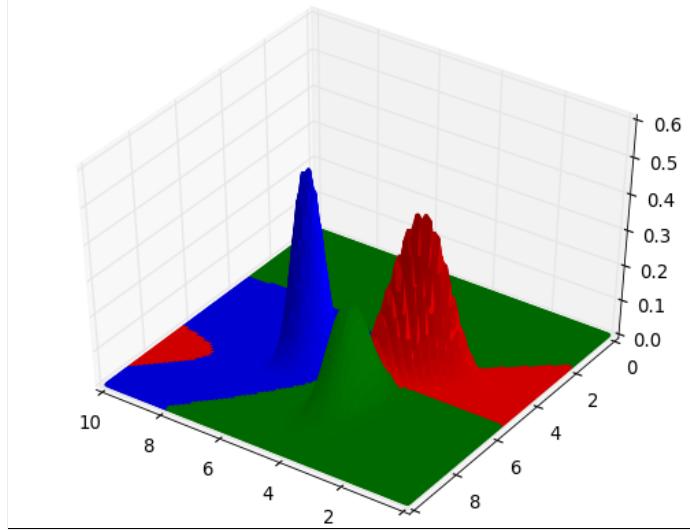


Figura 3: Ejemplo de frontera estilo parabólica. En este caso tenemos 3 tipos de clase.

## 2. Clasificación de imágenes sintéticas

### 2.1. Verdad terrestre

Supongamos que estamos analizando un terreno a través de imágenes satelitales. Cada porción de terreno recibe un color en base a la clase que se le es asignada. Las clases reales de las distintas partes de terreno son

llamadas la **verdad terrestre** y típicamente son el dato que no poseemos y queremos hallar. En este caso conocemos la verdad terrestre pero, para realizar la experimentación, asumiremos que no la conocemos y luego compararemos el resultado de nuestro algoritmo de clasificación con la misma.



*Verdad terrestre: cada porción de terreno está coloreada según la clase que le corresponde en la realidad.*

## 2.2. Creación de imágenes sintéticas

Típicamente poseemos imágenes satelitales ruidosas, donde no es tan fácil discernir a qué clase corresponde una sección del terreno. Nuestro objetivo ahora es generar imágenes sintéticas que se asemejen a las que poseeríamos en una situación más realista durante un problema de clasificación. Para hacerlo partiremos de la imagen que posee la **verdad terrestre** y haremos que los colores en nuestra imagen sintética se desvien ligeramente de los originales, generando ruido. Por ejemplo si en la imagen original una porción de terreno era puramente roja, en nuestra imagen sintetizada tendremos secciones con tonos rojizos, pero con ruido que lo desvía de ese rojo originalmente homogéneo. Las matrices de covarianza utilizadas para agregar ruido sobre los colores de la imagen original, serán aumentadas en complejidad progresivamente de manera que el ruido generado por las mismas sea mayor. Nuestra hipótesis es que esto hará que el algoritmo de clasificación se comporte peor en estos últimos casos.

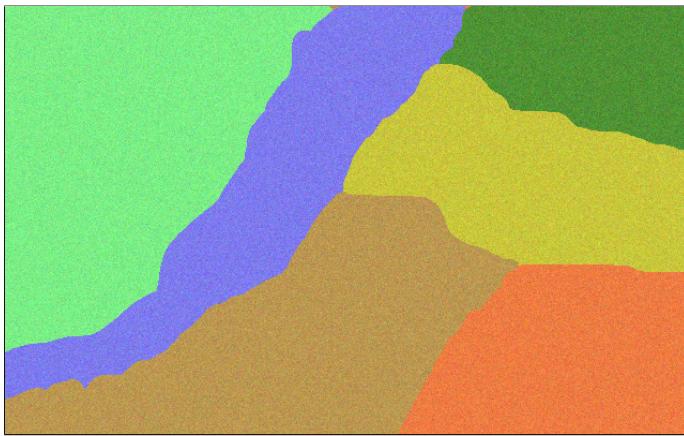
## 2.3. Modelo

El **featurespace** en este caso es tridimensional, una coordenada por cada una de los colores que conforman el modelo **RGB**. Las probabilidades a priori de cada una de las 6 clases serán tomadas como equiprobables. Las distribuciones asumidas para el algoritmo de clasificación serán en todos los casos gaussianas con  $\mu = (\text{color en verdad terrestre})$  y  $\Sigma = 3 * I_{(3,3)}$ .

## 2.4. Resultados

### 2.4.1. Imágenes sintéticas vs imágenes clasificadas

A continuación presentamos los resultados de la experimentación. Tenemos 6 clases, cada una de ellas tiene un color asociado que la representa. En la columna izquierda tenemos las imágenes sintéticas, generadas con ruido a partir de los colores representativos de cada clase. En la columna derecha tenemos el resultado de clasificar cada uno de esos puntos, en estas imágenes cada pixel es pintado con el color representativo de la clase que se le asignó. Es decir que mientras que en las imágenes sintéticas tenemos una cantidad arbitraria de colores distintos, en la derecha sólo tenemos 6 colores. Notar que la verdad terrestre también tiene sólo 6 colores: en definitiva esta imagen es equivalente a la clasificación perfecta de cada punto.



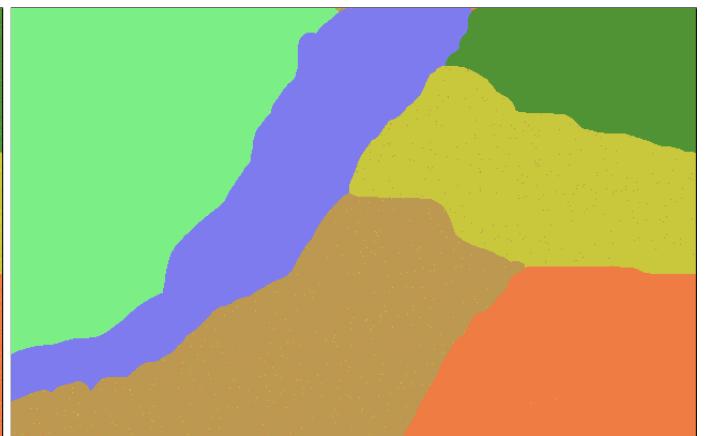
Ruido con matrices de covarianza isotrópica e iguales entre todas las clases.



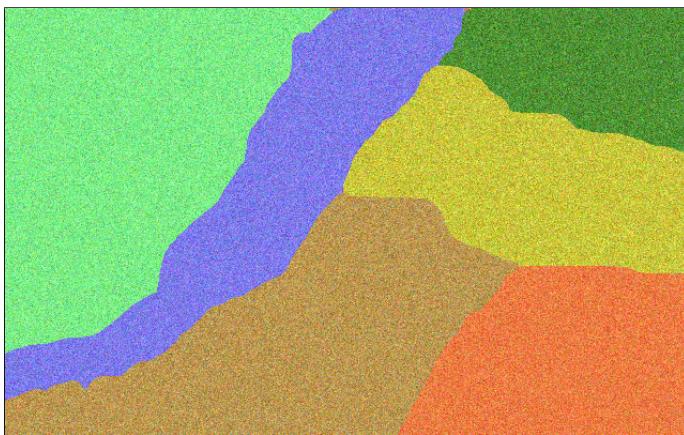
Resultado de clasificación, bastante similar a la verdad terrestre.



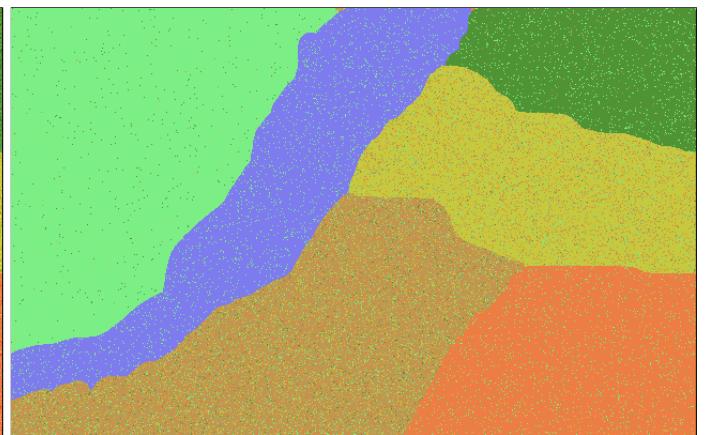
Ruido con matrices de covarianza diagonales y distintas entre todas las clases.



Resultado de clasificación, bastante similar a la verdad terrestre.



Ruido con matrices de covarianza no diagonales y distintas entre todas las clases.



Resultado de clasificación, con bastantes más errores que en el primer caso.

#### 2.4.2. Matrices de confusión

Si bien se puede ver a simple vista que en el tercer experimento la clasificación fue bastante peor que en el primero, todavía no utilizamos una métrica concreta que refleje esto. Las matrices de confusión nos indican cuál es la cantidad de puntos que clasificamos erróneamente, y su dimensión es de  $k \times k$  donde  $k$  es la cantidad de clases distintas. Para cada sujeto analizado se elige una fila y una columna y se lo posiciona en la matriz de confusión. Para seleccionar la fila se utiliza la clase real a la que pertenece el sujeto (en este caso un pixel

de terreno). Para seleccionar la columna se utiliza la clase predicha. Entonces idealmente nuestra matriz sería puramente diagonal, indicando que cada sujeto fue etiquetado en la clase de la que era realmente.

- Poco ruido en la imagen, generado con matrices de covarianza isotrópicas e iguales entre si:

	$K_1$	$K_2$	$K_3$	$K_4$	$K_5$	$K_6$
$K_1$	63288	0	0	0	1	0
$K_2$	0	53809	388	0	867	0
$K_3$	0	259	35095	0	4	0
$K_4$	0	0	0	22829	0	0
$K_5$	0	581	4	0	38134	0
$K_6$	0	0	0	0	0	40741

La cantidad total de pixeles mal clasificados es: 2014.

- Poco ruido en la imagen, generado con matrices de covarianza diagonales y distintas entre si:

	$K_1$	$K_2$	$K_3$	$K_4$	$K_5$	$K_6$
$K_1$	63289	0	0	0	0	0
$K_2$	0	54534	189	0	341	0
$K_3$	0	3	35353	0	2	0
$K_4$	0	0	0	22829	0	0
$K_5$	0	264	0	0	38455	0
$K_6$	0	0	0	0	0	40741

La cantidad total de pixeles mal clasificados es: 799.

- Ruido alto en la imagen, generado con matrices de covarianza no diagonales y distintas entre si:

	$K_1$	$K_2$	$K_3$	$K_4$	$K_5$	$K_6$
$K_1$	62669	41	0	429	150	0
$K_2$	2633	42312	4413	977	4729	0
$K_3$	1115	2538	31180	0	525	0
$K_4$	1101	19	0	21521	17	0
$K_5$	2017	3886	558	135	32123	0
$K_6$	2133	0	0	0	0	38608

La cantidad total de pixeles mal clasificados es: 27416.

## 2.5. Conclusión

Podemos ver que efectivamente el peor de los casos, aquel en donde clasificamos erróneamente la mayor cantidad de puntos, es donde hemos generado más ruido. El segundo caso dió mejor que el primero, esto nos indica que el ruido generado por matrices de covarianza diagonales distintas entre si, en este caso no influyó negativamente en las clasificaciones con respecto a las matrices isotrópicas. La generación de las matrices fue en parte aleatoria, y la repetición de este experimento podría resultar en matrices de confusión distintas. Un método más robusto sería repetir el experimento de montecarlo reiteradas veces y agregar los valores. Para más información sobre la forma en que se generaron las matrices, se puede acudir al código en matlab adjuntado.

La moraleja es que cuando dos clases, si bien distintas, poseen una distribución con alta varianza y además sus esperanzas están cerca, el clasificador no realizará un buen trabajo. En este caso por ejemplo, si tenemos dos colores similares como el amarillo y el marrón, y la variación es muy alta, el clasificador terminará confundiéndose bastante el marrón con el amarillo y viceversa. Una solución para esto podría ser utilizar otro método. Otra solución podría ser considerar aún más **features** de nuestros objetos, aumentando la dimensionalidad del problema y quizás haciendo que los datos se dispersen más. Por ejemplo si además de medir el color del suelo, midieramos la temperatura. En este caso podría suceder que el suelo marrón sea muy frío y el amarillo sea caliente, luego con esta dimensión nueva sería fácil distinguir una porción de la clase amarilla de otra de la clase marrón.

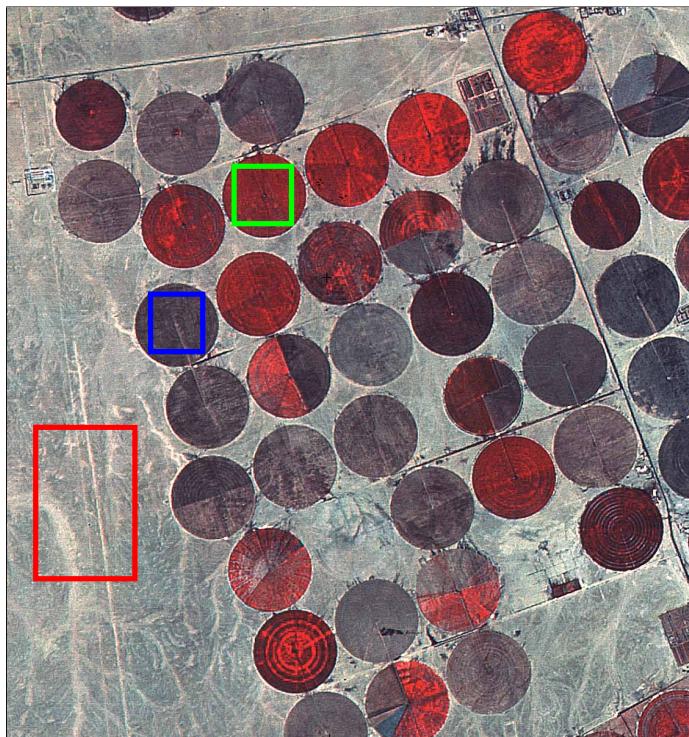
### 3. Clasificación de imágenes reales

#### 3.1. Estimando distribuciones

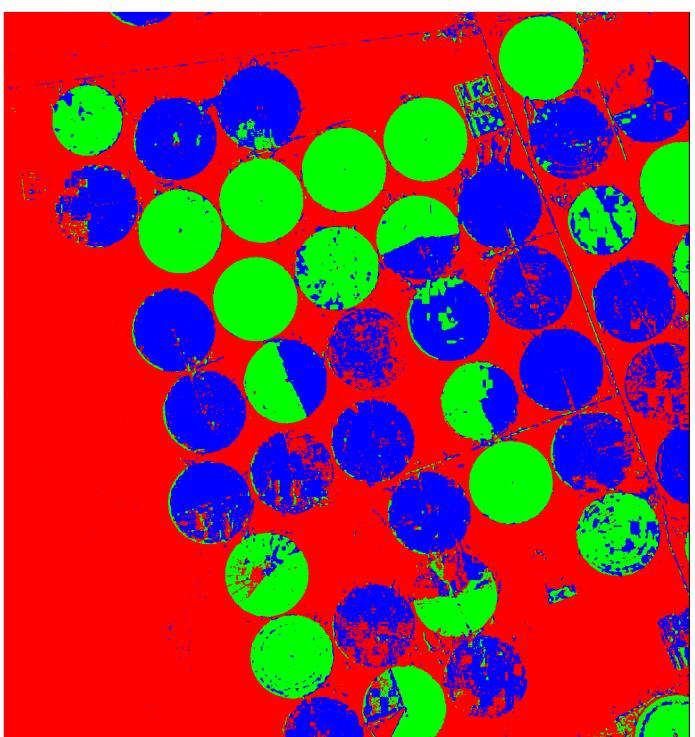
Ahora tenemos una imagen real tomada desde un satélite. Queremos realizar lo mismo que en el caso de las imágenes sintéticas, pero tenemos un problema: no conocemos la distribución de nuestras clases. Podemos empezar asumiendo que la distribución es, como en el caso anterior, una normal gaussiana. Pero todavía seguimos sin conocer los parámetros de la misma, es decir, nuestros  $\mu$  y  $\Sigma$  para cada caso. Para obtenerlos elegiremos “a ojo” distintas secciones de la imagen, asumiremos que estas están íntegramente compuestas por porciones de terrenos que corresponden a la misma clase. Partiendo de estas secciones, utilizaremos cada uno de sus puntos para calcular estimaciones de nuestros parámetros faltantes. Utilizaremos el promedio muestral y la varianza muestral, que son estimadores insesgados.

#### 3.2. Clasificación de la imagen

Utilizando el mismo algoritmo presentado en la primera sección, podemos proceder a tomar la misma imagen y ahora clasificar todos sus píxeles. Muchos de ellos ni siquiera fueron tenidos en cuenta a la hora de definir nuestras distribuciones. Para presentar el resultado, mostraremos cada píxel de la imagen con el color de la región de entrenamiento de la clase que le fue asignada:



Cada región utilizada para la estimación de los parámetros de cada clase está recuadrada.



Resultado de la clasificación.

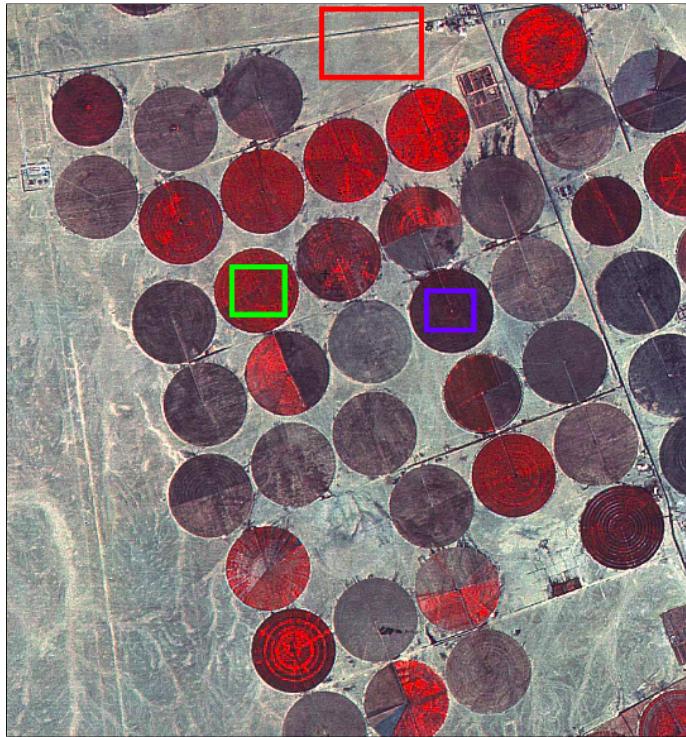
#### 3.3. Matrices de confusión

Al igual que en la sección de las imágenes sintéticas, ahora podemos calcular las matrices de confusión asociadas para cuantificar el desempeño de nuestro algoritmo. Esta vez no conocemos la verdad terrestre, así que definiremos a ojo porciones de terreno y consideraremos que todos los píxeles en ella pertenecen a la clase que intuitivamente debería pertenecer. Luego contrastaremos qué tan eficaz es nuestro algoritmo para encasillar en la clase apropiada cada pixel de esa región. Además, como utilizaremos regiones con distinta cantidad de muestras para generar las matrices, esta vez pondremos el porcentaje de acertados.

- Matriz de confusión considerando la verdad terrestre como las regiones de entrenamiento:

	$K_1$	$K_2$	$K_3$	% acierto
$K_1$	59979	0	21	$\sim 100\%$
$K_2$	74	14295	31	$\sim 99\%$
$K_3$	246	57	12897	$\sim 98\%$

- Ahora la verdad terrestre son otras regiones seleccionadas al azar, pero que tranquilamente podrían haber sido utilizadas para entrenar en un principio por su homogeneidad. Las regiones de entrenamiento se dejaron fijas. Podemos observar que el algoritmo hizo un buen trabajo de todas formas:



	$K_1$	$K_2$	$K_3$	% acierto
$K_1$	27782	1	217	$\sim 99\%$
$K_2$	74	11000	0	$\sim 99\%$
$K_3$	7	1141	7352	$\sim 86\%$

Regiones alternativas.

### 3.4. Conclusión

En definitiva si uno conoce las distribuciones subyacentes en los datos y además elige cuidadosamente las regiones de entrenamiento, puede implementar este algoritmo que tiene las ventajas de ser muy simple y eficiente, y obtener unos resultados bastante decentes. Sobre todo si se eligen correctamente las *features* a capturar de los objetos de estudio, de manera tal que las clases sean fácilmente separables.