Aprendizaje Supervisado: Clasificación

Juan F. Pérez

Departamento MACC
Matemáticas Aplicadas y Ciencias de la Computación
Universidad del Rosario

juanferna.perez@urosario.edu.co

2018

Contenidos

Introducción

Introducción

- Búsqueda de patrones
- Estudio de fenómenos físicos
- Reconocimiento de patrones
- Descubrimiento automático de regularidades
- Algoritmos computacionales

Ejemplo

Reconocimiento de dígitos

Ejemplo Reconocimiento de Dígitos

- Cada dígito es una imagen de 28x28 pixeles.
- Vector x de 784 números reales que representan la intensidad en cada pixel (0 blanco, 1 negro)
- Objetivo: construir un mecanismo que permita determinar automáticamente qué dígito corresponde a una nueva imagen dada en el mismo formato
- http://yann.lecun.com/exdb/mnist/

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina
- Conjunto de datos de entrenamiento

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina
- Conjunto de datos de entrenamiento
- N imágenes en el formato definido $\{x_1, \ldots, x_N\}$

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina
- Conjunto de datos de entrenamiento
- N imágenes en el formato definido $\{x_1, \ldots, x_N\}$
- x_i: vector de características de la imagen i

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina
- Conjunto de datos de entrenamiento
- N imágenes en el formato definido $\{x_1, \ldots, x_N\}$
- x_i: vector de características de la imagen i
- Para cada imagen conocemos la categoría (dígito que representa)

Ejemplo Reconocimiento de Dígitos

- Problema de aprendizaje de máquina
- Conjunto de datos de entrenamiento
- N imágenes en el formato definido $\{x_1, \ldots, x_N\}$
- x_i: vector de características de la imagen i
- Para cada imagen conocemos la categoría (dígito que representa)
- Categoría de la imagen i: vector t_i

Ejemplo Reconocimiento de Dígitos

- Resultado del algoritmo de aprendizaje de máquina: función $y(x)$

Ejemplo Reconocimiento de Dígitos

- Resultado del algoritmo de aprendizaje de máquina: función $y(x)$
- Para una imagen x dada, $y(x)$ es el dígito asociado

Ejemplo Reconocimiento de Dígitos

- Resultado del algoritmo de aprendizaje de máquina: función $y(x)$
- Para una imagen x dada, $y(x)$ es el dígito asociado
- Fase de **entrenamiento**: determinar $y(x)$ a partir de $\{(x_1, t_1), \dots, (x_N, t_1)\}$

Ejemplo Reconocimiento de Dígitos

- Resultado del algoritmo de aprendizaje de máquina: función $y(x)$
- Para una imagen x dada, $y(x)$ es el dígito asociado
- Fase de **entrenamiento**: determinar $y(x)$ a partir de $\{(x_1, t_1), \ldots, (x_N, t_1)\}$
- Fase de **prueba**: para unas imágenes x diferentes a las de entrenamiento, pero con categorías conocidas, probar la precisión de la función obtenida

Ejemplo Reconocimiento de Dígitos

- Resultado del algoritmo de aprendizaje de máquina: función $y(x)$
- Para una imagen x dada, $y(x)$ es el dígito asociado
- Fase de **entrenamiento**: determinar $y(x)$ a partir de $\{(x_1, t_1), \ldots, (x_N, t_1)\}$
- Fase de **prueba**: para unas imágenes x diferentes a las de entrenamiento, pero con categorías conocidas, probar la precisión de la función obtenida
- Capacidad de generalizar del modelo: responder correctamente a imágenes diferentes a las de entrenamiento

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)
- Reducción de variabilidad en los datos

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)
- Reducción de variabilidad en los datos
- Facilita la identificación de categorías

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)
- Reducción de variabilidad en los datos
- Facilita la identificación de categorías
- Simplifica las características a usar

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)
- Reducción de variabilidad en los datos
- Facilita la identificación de categorías
- Simplifica las características a usar
- Mejora eficiencia de los algoritmos de aprendizaje

Ejemplo Reconocimiento de Dígitos

Preprocesamiento:

- Transformación inicial de los datos antes de la fase de entrenamiento
- Dígitos: imágenes re-escaladas en cajas de tamaño fijo (28x28)
- Reducción de variabilidad en los datos
- Facilita la identificación de categorías
- Simplifica las características a usar
- Mejora eficiencia de los algoritmos de aprendizaje
- Extracción de características

Ejemplo Reconocimiento de Dígitos

- Datos de entrenamiento contienen tanto las características x_i como las categorías/etiquetas t_i

Ejemplo Reconocimiento de Dígitos

- Datos de entrenamiento contienen tanto las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

Ejemplo Reconocimiento de Dígitos

- Datos de entrenamiento contienen tanto las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

- Número de categorías finito

Ejemplo Reconocimiento de Dígitos

- Datos de entrenamiento contienen tanto las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

- Número de categorías finito

Clasificación

Ejemplo Reconocimiento de Dígitos

- Datos de entrenamiento contienen tanto las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

- Número de categorías finito

Clasificación

- Clasificar datos de entrada en una de un número finito de categorías

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i como las categorías/etiquetas t_i

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

- Resultado es una o varias variables continuas (no un número finito de categorías)

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i como las categorías/etiquetas t_i

Aprendizaje supervisado

- Resultado es una o varias variables continuas (no un número finito de categorías)

Regresión

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

- Objetivo es descubrir grupos similares

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

- Objetivo es descubrir grupos similares

Clustering (análisis de conglomerados)

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

- Objetivo es determinar la distribución de los datos en el espacio de entrada

Otros problemas de aprendizaje

- Datos de entrenamiento contienen las características x_i pero NO las categorías/etiquetas t_i

Aprendizaje no supervisado

- Objetivo es determinar la distribución de los datos en el espacio de entrada

Estimación de densidades

Modelos de Clasificación

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases ($C_k,\ k = 1, \ldots, K$)

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$
- Clases disyuntas: cada observación asignada a una sola de las clases (más usual)

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$
- Clases disyuntas: cada observación asignada a una sola de las clases (más usual)
- Espacio de entrada (donde representamos los datos de entrada) se divide en regiones de decisión

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$
- Clases disyuntas: cada observación asignada a una sola de las clases (más usual)
- Espacio de entrada (donde representamos los datos de entrada) se divide en regiones de decisión
- Fronteras o superficies entre regiones

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$
- Clases disyuntas: cada observación asignada a una sola de las clases (más usual)
- Espacio de entrada (donde representamos los datos de entrada) se divide en regiones de decisión
- Fronteras o superficies entre regiones
- Modelos lineales: superficies son funciones lineales del vector x (hiperplanos en espacio de dimensión D)

Modelos de Clasificación

Objetivo:

- Dado un vector de entrada x de dimensión D (características)
- Asignarlo a una de K clases ($C_k,\ k = 1, \ldots, K$)
- Clases disyuntas: cada observación asignada a una sola de las clases (más usual)
- Espacio de entrada (donde representamos los datos de entrada) se divide en regiones de decisión
- Fronteras o superficies entre regiones
- Modelos lineales: superficies son funciones lineales del vector x (hiperplanos en espacio de dimensión D)
- Datos linealmente separables: clases se pueden separar exactamente por funciones lineales

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- Dos clases: $t \in \{0, 1\}$

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- Dos clases: $t \in \{0, 1\}$
- $t = 1$: x en C_1
- $t = 0$: x en C_2

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- Dos clases: $t \in \{0, 1\}$
- $t = 1$: x en C_1
- $t = 0$: x en C_2
- Interpretar t como probabilidad de que x pertenezca a la clase C_1

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- $K > 2$ clases: t vector de longitud K igual a cero, excepto en la posición j donde es igual a 1 si x en C_j

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- $K > 2$ clases: t vector de longitud K igual a cero, excepto en la posición j donde es igual a 1 si x en C_j
- Ejemplo $K = 3$

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- $K > 2$ clases: t vector de longitud K igual a cero, excepto en la posición j donde es igual a 1 si x en C_j
- Ejemplo $K = 3$
- $t = (1, 0, 0)$ si x en C_1
- $t = (0, 1, 0)$ si x en C_2
- $t = (0, 0, 1)$ si x en C_3

Representación de categorías

- ¿Cómo representar categorías/etiquetas/objetivos t?
- $K > 2$ clases: t vector de longitud K igual a cero, excepto en la posición j donde es igual a 1 si x en C_j
- Ejemplo $K = 3$
- $t = (1, 0, 0)$ si x en C_1
- $t = (0, 1, 0)$ si x en C_2
- $t = (0, 0, 1)$ si x en C_3
- Interpretar t_k como probabilidad de que x pertenezca a la clase C_k

Funciones Discriminantes

Funciones Discriminantes

- Dado un vector de entrada x de dimensión D (características), asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$

Funciones Discriminantes

- Dado un vector de entrada x de dimensión D (características), asignarlo a una de K clases $(C_k,\ k = 1, \dots, K)$
- Discriminantes lineales: funciones de decisión son hiperplanos

Funciones Discriminantes

- Dado un vector de entrada x de dimensión D (características), asignarlo a una de K clases ($C_k,\ k = 1, \ldots, K$)
- Discriminantes lineales: funciones de decisión son hiperplanos
- Lineales en el espacio de las características

Funciones Discriminantes

- Dado un vector de entrada x de dimensión D (características), asignarlo a una de K clases $(C_k,\ k = 1, \ldots, K)$
- Discriminantes lineales: funciones de decisión son hiperplanos
- Lineales en el espacio de las características
- Inicialmente dos clases

Funciones Discriminantes Lineales - 2 Clases

- Dado un vector de entrada x de dimensión D (características), asignarlo a C_1 o C_2

Funciones Discriminantes Lineales - 2 Clases

- Dado un vector de entrada x de dimensión D (características), asignarlo a C_1 o C_2
- Discriminante lineal:

$$y(x) = w^T x + w_0$$

Funciones Discriminantes Lineales - 2 Clases

- Dado un vector de entrada x de dimensión D (características), asignarlo a C_1 o C_2
- Discriminante lineal:

$$y(x) = w^T x + w_0$$

- Equivalente:

$$y(x) = \sum_{d=1}^{D} w_d x_d + w_0$$

Funciones Discriminantes Lineales - 2 Clases

- Dado un vector de entrada x de dimensión D (características), asignarlo a C_1 o C_2
- Discriminante lineal:

$$y(x) = w^T x + w_0$$

- Equivalente:

$$y(x) = \sum_{d=1}^{D} w_d x_d + w_0$$

- Ejemplo $D = 3$ (características)

$$y(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0$$

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:

$$y(x) = w^T x + w_0$$

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:

$$y(x) = w^T x + w_0$$

- w: vector de pesos

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:

$$y(x) = w^T x + w_0$$

- w: vector de pesos
- w_0: sesgo

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:

$$y(x) = w^T x + w_0$$

- w: vector de pesos
- w_0: sesgo
- Para un x:
  - Si $y(x) \geq 0$, $x \in C_1$

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:
$$y(x) = w^T x + w_0$$
- w: vector de pesos
- w_0: sesgo
- Para un x:
  - Si $y(x) \geq 0$, $x \in C_1$
  - Si $y(x) < 0$, $x \in C_2$

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:
$$y(x) = w^T x + w_0$$
- w: vector de pesos
- w_0: sesgo
- Para un x:
  - Si $y(x) \geq 0$, $x \in C_1$
  - Si $y(x) < 0$, $x \in C_2$
- La frontera de decisión está dada por
$$y(x) = w^T x + w_0 = 0$$

Funciones Discriminantes Lineales - 2 Clases

- Discriminante lineal:
$$y(x) = w^T x + w_0$$
- w: vector de pesos
- w_0: sesgo
- Para un x:
  - Si $y(x) \geq 0$, $x \in C_1$
  - Si $y(x) < 0$, $x \in C_2$
- La frontera de decisión está dada por
$$y(x) = w^T x + w_0 = 0$$
- Hiperplano

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$
  - w es normal a la frontera

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$
  - w es normal a la frontera
- w_0 determina la ubicación de la frontera

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$
  - w es normal a la frontera
- w_0 determina la ubicación de la frontera
  - Distancia del origen al plano de frontera

$$\frac{w^T x}{||w||} = -\frac{w_0}{||w||}$$

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$
  - w es normal a la frontera
- w_0 determina la ubicación de la frontera
  - Distancia del origen al plano de frontera

$$\frac{w^T x}{||w||} = -\frac{w_0}{||w||}$$

  - Punto $t\frac{w}{||w||}$ tal que

$$w^T \frac{tw}{||w||} + w_0 = 0$$

Funciones Discriminantes Lineales - 2 Clases

- w determina la orientación de la frontera
  - Para x_A y x_B en la frontera $w^T(x_A - x_B) = 0$
  - w es normal a la frontera
- w_0 determina la ubicación de la frontera
  - Distancia del origen al plano de frontera

$$\frac{w^T x}{||w||} = -\frac{w_0}{||w||}$$

  - Punto $t\frac{w}{||w||}$ tal que

$$w^T \frac{tw}{||w||} + w_0 = 0$$

  - $$t = -\frac{w_0}{||w||}$$

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera
  - $\bar{x}$: proyección ortogonal de x en la frontera

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera
  - $\bar{x}$: proyección ortogonal de x en la frontera
  - $$x = \bar{x} + r\frac{w}{||w||}$$

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera
  - $\bar{x}$: proyección ortogonal de x en la frontera
  - $$x = \bar{x} + r\frac{w}{||w||}$$
  - r positivo o negativo de acuerdo a posición de x con respecto a la frontera

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera
  - $\bar{x}$: proyección ortogonal de x en la frontera
  - $$x = \bar{x} + r\frac{w}{||w||}$$
  - r positivo o negativo de acuerdo a posición de x con respecto a la frontera
  - $$w^T x + w_0 = w^T \bar{x} + w_0 + r\frac{w^T w}{||w||}$$

Funciones Discriminantes Lineales - 2 Clases

- El signo de $y(x)$ determina la ubicación de x con respecto a la frontera
  - $\bar{x}$: proyección ortogonal de x en la frontera
  - $$x = \bar{x} + r\frac{w}{||w||}$$
  - r positivo o negativo de acuerdo a posición de x con respecto a la frontera
  - $$w^T x + w_0 = w^T \bar{x} + w_0 + r\frac{w^T w}{||w||}$$
  - $$y(x) = r\frac{w^T w}{||w||} = r||w||$$

Funciones Discriminantes Lineales - 2 Clases

- Notación más compacta para $y(x) = w^T x + w_0$

Funciones Discriminantes Lineales - 2 Clases

- Notación más compacta para $y(x) = w^T x + w_0$
- $x_0 = 1$

Funciones Discriminantes Lineales - 2 Clases

- Notación más compacta para $y(x) = w^T x + w_0$
- $x_0 = 1$
- $\tilde{w} = (w_0, w)$

Funciones Discriminantes Lineales - 2 Clases

- Notación más compacta para $y(x) = w^T x + w_0$
- $x_0 = 1$
- $\tilde{w} = (w_0, w)$
- $\tilde{x} = (x_0, x)$

Funciones Discriminantes Lineales - 2 Clases

- Notación más compacta para $y(x) = w^T x + w_0$
- $x_0 = 1$
- $\tilde{w} = (w_0, w)$
- $\tilde{x} = (x_0, x)$
- $y(x) = \tilde{w}^T \tilde{x}$

Funciones Discriminantes Lineales - K Clases

- K funciones lineales

Funciones Discriminantes Lineales - K Clases

- K funciones lineales

$$y_k(x) = w_k^T x + w_{k0}$$

Funciones Discriminantes Lineales - K Clases

- K funciones lineales

$$y_k(x) = w_k^T x + w_{k0}$$

- Asignamos x a la clase C_k si

$$y_k(x) > y_j(x)$$

para todo $j \neq k$

Funciones Discriminantes Lineales - K Clases

- K funciones lineales

$$y_k(x) = w_k^T x + w_{k0}$$

- Asignamos x a la clase C_k si

$$y_k(x) > y_j(x)$$

para todo $j \neq k$

- Frontera de decisión entre clases C_k y C_j:

$$y_k(x) = y_j(x)$$

Funciones Discriminantes Lineales - K Clases

- K funciones lineales

$$y_k(x) = w_k^T x + w_{k0}$$

- Asignamos x a la clase C_k si

$$y_k(x) > y_j(x)$$

para todo $j \neq k$

- Frontera de decisión entre clases C_k y C_j:

$$y_k(x) = y_j(x)$$

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas
- Suponga que x_A y x_B en C_k

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas
- Suponga que x_A y x_B en C_k
- Combinaciones lineales de x_A y x_B:

$$\bar{x} = \alpha x_A + (1 - \alpha) x_B$$

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas
- Suponga que x_A y x_B en C_k
- Combinaciones lineales de x_A y x_B:

$$\bar{x} = \alpha x_A + (1-\alpha)x_B$$

- $$y_k(\bar{x}) = y_k(\alpha x_A + (1-\alpha)x_B) = \alpha y_k(x_A) + (1-\alpha)y_k(x_B)$$

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas
- Suponga que x_A y x_B en C_k
- Combinaciones lineales de x_A y x_B:

$$\bar{x} = \alpha x_A + (1-\alpha)x_B$$

- $$y_k(\bar{x}) = y_k(\alpha x_A + (1-\alpha)x_B) = \alpha y_k(x_A) + (1-\alpha)y_k(x_B)$$
- Como $y_k(x_A) > y_j(x_A)$ y $y_k(x_B) > y_j(x_B)$, entonces

$$y_k(\bar{x}) > y_j(\bar{x})$$

Funciones Discriminantes Lineales - K Clases

- K regiones son convexas
- Suponga que x_A y x_B en C_k
- Combinaciones lineales de x_A y x_B:

$$\bar{x} = \alpha x_A + (1-\alpha)x_B$$

- $$y_k(\bar{x}) = y_k(\alpha x_A + (1-\alpha)x_B) = \alpha y_k(x_A) + (1-\alpha)y_k(x_B)$$

- Como $y_k(x_A) > y_j(x_A)$ y $y_k(x_B) > y_j(x_B)$, entonces

$$y_k(\bar{x}) > y_j(\bar{x})$$

- $\bar{x}$ también está en C_k

Algoritmos para Estimar los Parámetros del Discriminante Lineal

Algoritmos para Estimar los Parámetros del Discriminante Lineal

- Objetivo: determinar los valores de w y w_0

Algoritmos para Estimar los Parámetros del Discriminante Lineal

- Objetivo: determinar los valores de w y w_0
- Usando los datos de entrenamiento

Algoritmos para Estimar los Parámetros del Discriminante Lineal

- Objetivo: determinar los valores de w y w_0
- Usando los datos de entrenamiento
- Buscamos los mejores parámetros (identifique de la mejor manera las clases)

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)
- Su clase es t_i (vector de tamaño K)

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)
- Su clase es t_i (vector de tamaño K)
- Modelo lineal para la clase C_k

$$y_k(x_i) = w_k^T x_i + w_{k0} = \tilde{w}_k^T \tilde{x}_i$$

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)
- Su clase es t_i (vector de tamaño K)
- Modelo lineal para la clase C_k

$$y_k(x_i) = w_k^T x_i + w_{k0} = \tilde{w}_k^T \tilde{x}_i$$

- $\tilde{W}$: matriz con k-ésima columna $\tilde{w}_k$

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)
- Su clase es t_i (vector de tamaño K)
- Modelo lineal para la clase C_k

$$y_k(x_i) = w_k^T x_i + w_{k0} = \tilde{w}_k^T \tilde{x}_i$$

- $\tilde{W}$: matriz con k-ésima columna $\tilde{w}_k$
- Modelo lineal:

$$y(x_i) = \tilde{W}^T \tilde{x}_i$$

Mínimos Cuadrados

- Para una observación x_i (vector de tamaño D)
- Su clase es t_i (vector de tamaño K)
- Modelo lineal para la clase C_k

$$y_k(x_i) = w_k^T x_i + w_{k0} = \tilde{w}_k^T \tilde{x}_i$$

- $\tilde{W}$: matriz con k-ésima columna $\tilde{w}_k$
- Modelo lineal:

$$y(x_i) = \tilde{W}^T \tilde{x}_i$$

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T (y(x_i) - t_i) = (\tilde{W}^T \tilde{x}_i - t_i)^T (\tilde{W}^T \tilde{x}_i - t_i)$$

Mínimos Cuadrados

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T(y(x_i) - t_i) = (\tilde{W}^T\tilde{x}_i - t_i)^T(\tilde{W}^T\tilde{x}_i - t_i)$$

Mínimos Cuadrados

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T (y(x_i) - t_i) = (\tilde{W}^T \tilde{x}_i - t_i)^T (\tilde{W}^T \tilde{x}_i - t_i)$$

- T: matriz cuya i-ésima fila es t_i

Mínimos Cuadrados

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T(y(x_i) - t_i) = (\tilde{W}^T\tilde{x}_i - t_i)^T(\tilde{W}^T\tilde{x}_i - t_i)$$

- T: matriz cuya i-ésima fila es t_i
- Traza de la matriz

$$(\tilde{X}\tilde{W} - T)^T(\tilde{X}\tilde{W} - T)$$

es igual a la suma de las diferencias para todos los datos

Mínimos Cuadrados

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T (y(x_i) - t_i) = (\tilde{W}^T \tilde{x}_i - t_i)^T (\tilde{W}^T \tilde{x}_i - t_i)$$

- T: matriz cuya i-ésima fila es t_i
- Traza de la matriz

$$(\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T)$$

  es igual a la suma de las diferencias para todos los datos
- Minimizar error (distancia):

$$\frac{1}{2} \mathrm{Tr} \left\{ (\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T) \right\}$$

Mínimos Cuadrados

- Distancia para dato x_i:

$$(y(x_i) - t_i)^T (y(x_i) - t_i) = (\tilde{W}^T \tilde{x}_i - t_i)^T (\tilde{W}^T \tilde{x}_i - t_i)$$

- T: matriz cuya i-ésima fila es t_i
- Traza de la matriz

$$(\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T)$$

  es igual a la suma de las diferencias para todos los datos
- Minimizar error (distancia):

$$\frac{1}{2} \mathrm{Tr} \left\{ (\tilde{X}\tilde{W} - T)^T (\tilde{X}\tilde{W} - T) \right\}$$

- Solución que minimiza el error:

$$\tilde{W} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T T = \tilde{X}^+ T$$

Mínimos Cuadrados

- La aplicación del modelo a un nuevo dato x es

$$y(x) = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^+)^T \tilde{x}$$

Mínimos Cuadrados

- La aplicación del modelo a un nuevo dato x es

$$y(x) = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^+)^T \tilde{x}$$

- Formula cerrada

Mínimos Cuadrados

- La aplicación del modelo a un nuevo dato x es

$$y(x) = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^+)^T \tilde{x}$$

- Formula cerrada
- Resultado no limitado al intervalo $[0, 1]$ (no interpretable como probabilidades)

Mínimos Cuadrados

- La aplicación del modelo a un nuevo dato x es

$$y(x) = \tilde{W}^T \tilde{x} = T^T (\tilde{X}^+)^T \tilde{x}$$

- Formula cerrada
- Resultado no limitado al intervalo $[0, 1]$ (no interpretable como probabilidades)
- Método sensible a datos extremos

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.
- Calculamos sus características promedio

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.
- Calculamos sus características promedio

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

- Separación entre medias como medida de separación entre clases

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.
- Calculamos sus características promedio

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

- Separación entre medias como medida de separación entre clases
- Escoger w para maximizar

$$m_2 - m_1 = w^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.
- Calculamos sus características promedio

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in C_1} x_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in C_2} x_n$$

- Separación entre medias como medida de separación entre clases
- Escoger w para maximizar

$$m_2 - m_1 = w^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

- m_i es la media de los datos de la clase C_i proyectados

Discriminante Lineal de Fisher

- Dos clases con N_1 y N_2 datos de entrenamiento resp.
- Calculamos sus características promedio

$$\boldsymbol{m}_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n \qquad \boldsymbol{m}_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n$$

- Separación entre medias como medida de separación entre clases
- Escoger w para maximizar

$$m_2 - m_1 = w^T(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

- m_i es la media de los datos de la clase C_i proyectados
- No es suficiente: puede haber mucho traslape en puntos proyectados

Discriminante Lineal de Fisher

- Criterio de Fisher: maximizar diferencia entre clases y al mismo tiempo minimizar la varianza al interior de cada clase

Discriminante Lineal de Fisher

- Criterio de Fisher: maximizar diferencia entre clases y al mismo tiempo minimizar la varianza al interior de cada clase
- La varianza al interior de cada clase (datos transformados) es

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- con $y_n = w^T x_n$

Discriminante Lineal de Fisher

- Criterio de Fisher: maximizar diferencia entre clases y al mismo tiempo minimizar la varianza al interior de cada clase
- La varianza al interior de cada clase (datos transformados) es

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- con $y_n = w^T x_n$
- Criterio de Fisher

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

Discriminante Lineal de Fisher

- Criterio de Fisher: maximizar diferencia entre clases y al mismo tiempo minimizar la varianza al interior de cada clase
- La varianza al interior de cada clase (datos transformados) es

$$s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$$

- con $y_n = w^T x_n$
- Criterio de Fisher

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Alternativamente

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Discriminante Lineal de Fisher

- Criterio de Fisher

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Discriminante Lineal de Fisher

- Criterio de Fisher

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- S_B es la matriz de covarianzas entre clases

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

Discriminante Lineal de Fisher

- Criterio de Fisher

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- S_B es la matriz de covarianzas entre clases

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

- S_W es la matriz de covarianzas al interior de cada clase

$$S_W = \sum_{n \in C_1} (x_n - \boldsymbol{m}_1)(x_n - \boldsymbol{m}_1)^T + \sum_{n \in C_2} (x_n - \boldsymbol{m}_2)(x_n - \boldsymbol{m}_2)^T$$

Discriminante Lineal de Fisher

- Criterio de Fisher

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

- S_B es la matriz de covarianzas entre clases

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$$

- S_W es la matriz de covarianzas al interior de cada clase

$$S_W = \sum_{n \in C_1} (x_n - \boldsymbol{m}_1)(x_n - \boldsymbol{m}_1)^T + \sum_{n \in C_2} (x_n - \boldsymbol{m}_2)(x_n - \boldsymbol{m}_2)^T$$

- Criterio de Fisher:

$$w \propto S_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

Scikit-Learn

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)
- Datos: tablas/dataframes

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)
- Datos: tablas/dataframes
- Tabla de características (features - X)

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)
- Datos: tablas/dataframes
- Tabla de características (features - X)
- Filas: observaciones (`n_samples`)
- Columnas: características (`n_features`)

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)
- Datos: tablas/dataframes
- Tabla de características (features - X)
- Filas: observaciones (`n_samples`)
- Columnas: características (`n_features`)
- Tabla de etiquetas (targets - y)

Scikit-Learn

- Librería para Python con múltiples algoritmos de aprendizaje de máquina
- Uso uniforme de los métodos disponibles (API)
- Datos: tablas/dataframes
- Tabla de características (features - X)
- Filas: observaciones (`n_samples`)
- Columnas: características (`n_features`)
- Tabla de etiquetas (targets - y)
- Filas: observaciones (`n_samples`)
- Columnas: etiquetas (`n_targets`)

Scikit-Learn

- Número de métodos limitado y similares

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:
  - Importar la clase de modelo buscado

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:
  - Importar la clase de modelo buscado
  - Escoger parámetros

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:
  - Importar la clase de modelo buscado
  - Escoger parámetros
  - Determinar matrices de características y objetivos

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:
  - Importar la clase de modelo buscado
  - Escoger parámetros
  - Determinar matrices de características y objetivos
  - Ajustar el modelo a los datos con el método `fit()`

Scikit-Learn

- Número de métodos limitado y similares
- Datos representados con arreglos de Numpy y dataframes de pandas
- Datos representados con arreglos de Numpy y dataframes de pandas
- Incluye valores de parámetros por defecto
- API:
    - Importar la clase de modelo buscado
    - Escoger parámetros
    - Determinar matrices de características y objetivos
    - Ajustar el modelo a los datos con el método `fit()`
    - Aplicar el modelo a nuevos datos con el método `predict()` (aprendizaje supervisado)

Librería Seaborn

- Librería para graficar en Python
- Seaborn y scikit-learn disponibles en **Anaconda**
- Ambiente de desarrollo: **Spyder**

```
import seaborn as sns
iris = sns.load_dataset('iris')
print(iris.head())
sns.pairplot(iris, hue = 'species', size = 1.5)
```

Scikit-Learn en los datos `iris`

- Continuando el código anterior
- Seleccionemos características y objetivos

```
X_iris = iris.drop('species', axis=1)
print(X_iris.shape)
y_iris = iris['species']
print(y_iris.shape)
```

Seleccionar datos de entrenamiento y prueba

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(
  X_iris, y_iris, random_state=1)
print(Xtrain.shape)
print(Xtest.shape)
```

Importar y Usar el Discriminante Lineal

```
from sklearn.discriminant_analysis import
    LinearDiscriminantAnalysis
model = LinearDiscriminantAnalysis()
model.fit(Xtrain, ytrain)
y_model = model.predict(Xtest)
```

Evaluar la Precisión del método

```
from sklearn.metrics import accuracy_score
import numpy
numpy.set_printoptions(threshold=numpy.nan)
acc_score = accuracy_score(ytest, y_model)
print("Precisión: ", acc_score)
print("Test")
print(ytest.values)
print("Modelo")
print(y_model)
```

Discriminante Lineal para Reconocer Dígitos

```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.images.shape)
```

Discriminante Lineal para Reconocer Dígitos

```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(
  10, 10, figsize=(8, 8),
  subplot_kw={'xticks':[], 'yticks':[]},
  gridspec_kw=dict(hspace=0.1, wspace=0.1)
  )
```

Discriminante Lineal para Reconocer Dígitos

```
for i, ax in enumerate(axes.flat):
    ax.imshow(digits.images[i])
    ax.text(0.05, 0.05, str(digits.target[i]),
            color='green'
            )
```

Discriminante Lineal para Reconocer Dígitos

```
X = digits.data
print("Tamaño X: ", X.shape)
y = digits.target
print("Tamaño y: ", y.shape)
```

Discriminante Lineal para Reconocer Dígitos

```
import pandas as pd
X = pd.DataFrame(X)
y = pd.Series(y)
```

Discriminante Lineal para Reconocer Dígitos

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = \
  train_test_split(X, y, random_state=0)
print("Tamaño Xtrain: ", Xtrain.shape)
print("Tamaño Xtest: ", Xtest.shape)
```

Discriminante Lineal para Reconocer Dígitos

```
from sklearn.discriminant_analysis import \
    LinearDiscriminantAnalysis
modelo = LinearDiscriminantAnalysis()
modelo.fit(Xtrain, ytrain)
y_modelo = modelo.predict(Xtest)
```

Discriminante Lineal para Reconocer Dígitos

```
from sklearn.metrics import accuracy_score
acc_score = accuracy_score(ytest, y_modelo)
print("Precisión: ", acc_score)
```

Discriminante Lineal para Reconocer Dígitos

```
from sklearn.metrics import confusion_matrix
con_mat = confusion_matrix(ytest, y_modelo)
```

Discriminante Lineal para Reconocer Dígitos

```
import seaborn as sns
plt.clf()
sns.heatmap(con_mat, square=True, \
  annot=True, cbar=False)
plt.xlabel('valor predicho')
plt.ylabel('valor real')
```

Discriminante Lineal para Reconocer Dígitos

```
fig, axes = plt.subplots( \
  10, 10, figsize=(8, 8), \
        subplot_kw={'xticks':[], 'yticks':[]}, \
        gridspec_kw=dict(hspace=0.1, wspace=0.1) \
)
```

Discriminante Lineal para Reconocer Dígitos

```
for i, ax in enumerate(axes.flat):
    ax.imshow(digits.images[ytest.index[i]], \
      cmap='binary', interpolation='nearest')
    ax.text(0.05, 0.05, str(y_modelo[i]), \
      color='green' \
      if (ytest.values[i] == y_modelo[i]) \
                            else 'red'
      )
```

Clasificador Bayesiano Ingenuo

Probabilidades condicionales

- A, B eventos

Probabilidades condicionales

- A, B eventos
- $A|B$: el evento A ocurre dado que el evento B ocurrió

Probabilidades condicionales

- A, B eventos
- $A|B$: el evento A ocurre dado que el evento B ocurrió
- $B|A$: el evento B ocurre dado que el evento A ocurrió

Probabilidades condicionales

- A, B eventos
- $A|B$: el evento A ocurre dado que el evento B ocurrió
- $B|A$: el evento B ocurre dado que el evento A ocurrió
- $P(A)$: probabilidad de que A ocurra

Probabilidades condicionales

- A, B eventos
- $A|B$: el evento A ocurre dado que el evento B ocurrió
- $B|A$: el evento B ocurre dado que el evento A ocurrió
- $P(A)$: probabilidad de que A ocurra
- $P(A|B)$: probabilidad de que A ocurra dado que B ocurrió

Probabilidades condicionales

- A, B eventos
- $A|B$: el evento A ocurre dado que el evento B ocurrió
- $B|A$: el evento B ocurre dado que el evento A ocurrió
- $P(A)$: probabilidad de que A ocurra
- $P(A|B)$: probabilidad de que A ocurra dado que B ocurrió
- $P(B|A)$: probabilidad de que B ocurra dado que A ocurrió

Probabilidades condicionales

- $P(A)$, $P(B)$: probabilidades a priori (prior)

Probabilidades condicionales

- $P(A)$, $P(B)$: probabilidades a priori (prior)
- $P(A|B)$, $P(B|A)$: probabilidades a posteriori (posterior)

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es mayor que 1

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es mayor que 1
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es mayor que 1
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es mayor que 1
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$
- $$P(A|B) = \frac{3}{5}$$

Probabilidades condicionales

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es mayor que 1
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$
- $$P(A|B) = \frac{3}{5}$$
- $$P(B|A) = \frac{3}{3} = 1$$

Teorema de Bayes

Si $P(B) > 0$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

Teorema de Bayes

Ejemplo: lanzamiento de un dado

Teorema de Bayes

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es primo

Teorema de Bayes

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es primo
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$

Teorema de Bayes

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es primo
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$
- $$P(B|A) = \frac{3}{3} = 1$$

Teorema de Bayes

Ejemplo: lanzamiento de un dado

- A el resultado es par
- B el resultado es primo
- $$P(A) = \frac{3}{6} = \frac{1}{2}$$
- $$P(B) = \frac{5}{6}$$
- $$P(B|A) = \frac{3}{3} = 1$$
- $$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{(1)(1/2)}{(5/6)} = \frac{3}{5}$$

Clasificador Bayesiano Ingenuo

- x: características de una observación

Clasificador Bayesiano Ingenuo

- x: características de una observación
- $C_1, \ldots, C_K$: categorías

Clasificador Bayesiano Ingenuo

- x: características de una observación
- $C_1, \ldots, C_K$: categorías
- $P(x)$: prob. de que una observación tenga las características x

Clasificador Bayesiano Ingenuo

- x: características de una observación
- $C_1, \ldots, C_K$: categorías
- $P(x)$: prob. de que una observación tenga las características x
- $P(C_k)$: prob. de que una observación sea de la categoría C_k

Clasificador Bayesiano Ingenuo

- x: características de una observación
- $C_1, \ldots, C_K$: categorías
- $P(x)$: prob. de que una observación tenga las características x
- $P(C_k)$: prob. de que una observación sea de la categoría C_k
- Clasificación de x:

$$P(C_k|x)$$

Clasificador Bayesiano Ingenuo

- x: características de una observación
- $C_1, \ldots, C_K$: categorías
- $P(x)$: prob. de que una observación tenga las características x
- $P(C_k)$: prob. de que una observación sea de la categoría C_k
- Clasificación de x:

$$P(C_k|x)$$

- Bayes:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Clasificador Bayesiano Ingenuo

Seleccionando entre dos clases C_k y C_j

- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Clasificador Bayesiano Ingenuo

Seleccionando entre dos clases C_k y C_j

- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$

Clasificador Bayesiano Ingenuo

Seleccionando entre dos clases C_k y C_j

- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- Cociente:
$$\frac{P(C_k|x)}{P(C_j|x)} = \frac{P(x|C_k)P(C_k)}{P(x|C_j)P(C_j)}$$

Clasificador Bayesiano Ingenuo

Seleccionando entre dos clases C_k y C_j

- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- Cociente:
$$\frac{P(C_k|x)}{P(C_j|x)} = \frac{P(x|C_k)P(C_k)}{P(x|C_j)P(C_j)}$$
- Si es mayor a 1 escogemos C_k, de lo contrario C_j

Clasificador Bayesiano Ingenuo

Seleccionando entre dos clases C_k y C_j

- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- $$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)}$$
- Cociente:
$$\frac{P(C_k|x)}{P(C_j|x)} = \frac{P(x|C_k)P(C_k)}{P(x|C_j)P(C_j)}$$
- Si es mayor a 1 escogemos C_k, de lo contrario C_j
- Entre K categorías: escogemos la categoría para la que el valor de $P(x|C_k)P(C_k)$ es mayor

Clasificador Bayesiano Ingenuo

- Cantidad clave:

$$P(x|C_k)P(C_k)$$

Clasificador Bayesiano Ingenuo

- Cantidad clave:
$$P(x|C_k)P(C_k)$$
- $P(C_k)$: estimar como fracción de observaciones en categoría C_k

Clasificador Bayesiano Ingenuo

- Cantidad clave:
$$P(x|C_k)P(C_k)$$
- $P(C_k)$: estimar como fracción de observaciones en categoría C_k
- $P(x|C_k)$: más complejo

Clasificador Bayesiano Ingenuo

- Cantidad clave:

$$P(x|C_k)P(C_k)$$

- $P(C_k)$: estimar como fracción de observaciones en categoría C_k
- $P(x|C_k)$: más complejo
- *Ingenuidad*: suponer un modelo *paramétrico* sencillo para $P(x|C_k)$

Clasificador Bayesiano Ingenuo

- Cantidad clave:

$$P(x|C_k)P(C_k)$$

- $P(C_k)$: estimar como fracción de observaciones en categoría C_k
- $P(x|C_k)$: más complejo
- *Ingenuidad*: suponer un modelo *paramétrico* sencillo para $P(x|C_k)$
- Modelo generativo

Clasificador Bayesiano Ingenuo Gaussiano

- Los datos en cada categoría tienen una distribución normal/Gaussiana

Clasificador Bayesiano Ingenuo Gaussiano

- Los datos en cada categoría tienen una distribución normal/Gaussiana
- A partir de los datos de entrenamiento en C_k estimar la media μ_k y la varianza σ_k^2 (parámetros) de los datos en C_k

Clasificador Bayesiano Ingenuo Gaussiano

- Los datos en cada categoría tienen una distribución normal/Gaussiana
- A partir de los datos de entrenamiento en C_k estimar la media μ_k y la varianza σ_k^2 (parámetros) de los datos en C_k
- Calcular $P(x|C_k)$ como la probabilidad de obtener x como resultado de una distribución normal con media y varianza calculadas

Clasificador Bayesiano Ingenuo Gaussiano

- Los datos en cada categoría tienen una distribución normal/Gaussiana
- A partir de los datos de entrenamiento en C_k estimar la media μ_k y la varianza σ_k^2 (parámetros) de los datos en C_k
- Calcular $P(x|C_k)$ como la probabilidad de obtener x como resultado de una distribución normal con media y varianza calculadas
- Para características continuas usar densidad de probabilidad $f(x|C_k)$

Clasificador Bayesiano Ingenuo Gaussiano

- Ejemplo con una sola característica (o cada característica independiente):

$$f(x|C_k) = \frac{1}{(2\pi\sigma)^{1/2}} \exp\{-\frac{1}{2}\frac{(x-\mu_k)}{\sigma_k}\}$$

Clasificador Bayesiano Ingenuo Gaussiano

- Ejemplo con una sola característica (o cada característica independiente):

$$f(x|C_k) = \frac{1}{(2\pi\sigma)^{1/2}} \exp\{-\frac{1}{2}\frac{(x-\mu_k)}{\sigma_k}\}$$

- Ejemplo con D características, vector promedio μ_k y matriz de covarianzas Σ:

$$f(x|C_k) = \frac{1}{2\pi^{D/2}\Sigma} \exp\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\}$$

Usando Python para visualizar datos Gaussianos

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_blobs

X,y = make_blobs(100, 2, centers=2, random_state=1, \
   cluster_std=1.5)
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='RdBu' )
```

make_blobs(número de muestras, número de características, número d centroides, semillas, desviación estándar)

El Clasificador Bayesiano Ingenuo en Scikit Learn

```python
from sklearn.naive_bayes import GaussianNB
modelo = GaussianNB()
modelo.fit(X,y)
```

El Clasificador Bayesiano Ingenuo en Scikit Learn

```
rng = np.random.RandomState(0)
Xnew = [-15, -8] + [17, 16] * rng.rand(1000, 2)
ynew = modelo.predict(Xnew)
```

El Clasificador Bayesiano Ingenuo en Scikit Learn

```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')
lim = plt.axis()
plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, \
  cmap='RdBu', alpha=0.1)
plt.axis(lim)
```

El Clasificador Bayesiano Ingenuo en Scikit Learn

```
yprob = modelo.predict_proba(Xnew)
print(yprob[0:9])
```

Clasificador Bayesiano Ingenuo para los datos Iris

```
import seaborn as sns
iris = sns.load_dataset('iris')
print(iris.head())
```

Seleccionar datos de entrenamiento y prueba

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = \
  train_test_split(X_iris, y_iris, random_state=1)
print(Xtrain.shape)
print(Xtest.shape)
```

Importar y Usar el Clasificador Bayesiano Ingenuo

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(Xtrain, ytrain)
y_model = model.predict(Xtest)
```

Evaluar la Precisión del método

```
from sklearn.metrics import accuracy_score
import numpy
numpy.set_printoptions(threshold=numpy.nan)
acc_score = accuracy_score(ytest, y_model)
print("Precisión: ", acc_score)
print("Test")
print(ytest.values)
print("Modelo")
print(y_model)
```

Support Vector Machines

Support Vector Machines

Máquinas de Vectores de Soporte

- Considere el problema de encontrar un clasificador lineal

Support Vector Machines

Máquinas de Vectores de Soporte

- Considere el problema de encontrar un clasificador lineal
- Vector de características: x

Support Vector Machines

Máquinas de Vectores de Soporte

- Considere el problema de encontrar un clasificador lineal
- Vector de características: x
- Encontrar una función $y(x)$

$$y(x) = w^T x + w_0$$

Support Vector Machines

Máquinas de Vectores de Soporte

- Considere el problema de encontrar un clasificador lineal
- Vector de características: x
- Encontrar una función $y(x)$

$$y(x) = w^T x + w_0$$

- Determinar los parámetros w y w_0

Support Vector Machines

Máquinas de Vectores de Soporte

- Datos de entrenamiento: características x_i, $i = 1, \ldots, N$

Support Vector Machines

Máquinas de Vectores de Soporte

- Datos de entrenamiento: características x_i, $i = 1, \ldots, N$
- Etiquetas o targets $t_i \in \{-1, 1\}$

Support Vector Machines

Máquinas de Vectores de Soporte

- Datos de entrenamiento: características x_i, $i = 1, \ldots, N$
- Etiquetas o targets $t_i \in \{-1, 1\}$
- y clasifica correctamente a x_i si $y(x_i) > 0$ para $t_i = 1$ y $y(x_i) < 0$ para $t_i = -1$

Support Vector Machines

Máquinas de Vectores de Soporte

- Datos de entrenamiento: características x_i, $i = 1, \ldots, N$
- Etiquetas o targets $t_i \in \{-1, 1\}$
- y clasifica correctamente a x_i si $y(x_i) > 0$ para $t_i = 1$ y $y(x_i) < 0$ para $t_i = -1$
- Para todos los puntos bien clasificados (e.g., datos linealmente separables)

$$t_i y(x_i) > 0$$

Support Vector Machines

Máquinas de Vectores de Soporte

- Distancia de un punto x al plano $y(x) = w^T x + w_0 = 0$

$$\frac{|y(x)|}{||w||}$$

Support Vector Machines

Máquinas de Vectores de Soporte

- Distancia de un punto x al plano $y(x) = w^T x + w_0 = 0$

$$\frac{|y(x)|}{||w||}$$

- Buscamos los parámetros w y w_0 que maximizen la distancia al plano de clasificación con puntos correctamente clasificados

Support Vector Machines

Máquinas de Vectores de Soporte

- Distancia de un punto x al plano $y(x) = w^T x + w_0 = 0$

$$\frac{|y(x)|}{||w||}$$

- Buscamos los parámetros w y w_0 que maximizen la distancia al plano de clasificación con puntos correctamente clasificados
- Distancia de punto x_i a superficie de decisión es

$$\frac{t_i y(x_i)}{||w||}$$

Support Vector Machines

Máquinas de Vectores de Soporte

- Distancia de un punto x al plano $y(x) = w^T x + w_0 = 0$

$$\frac{|y(x)|}{||w||}$$

- Buscamos los parámetros w y w_0 que maximizen la distancia al plano de clasificación con puntos correctamente clasificados
- Distancia de punto x_i a superficie de decisión es

$$\frac{t_i y(x_i)}{||w||}$$

- Problema de optimización:

$$\arg\max \left\{ \frac{1}{||w||} \min_i (t_i(w^T x_i + w_0)) \right\}$$

Support Vector Machines en Scikit Learn

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets.samples_generator import \
    make_blobs
X, y = make_blobs(n_samples=50, centers=2,
    random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, \
    cmap='autumn')
```

Support Vector Machines en Scikit Learn

```
from sklearn.svm import SVC
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
```

Support Vector Machines en Scikit Learn

```
def plot_svm(model):
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    X, Y = np.meshgrid(x, y)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    ax.contour(X, Y, P, colors='k', \
      levels=[-1, 0, 1], alpha=0.5,
      linestyles=['--', '-', '--'])
```

Support Vector Machines en Scikit Learn

```
plot_svm(model)

print("Vectores de soporte:\n", \
  model.support_vectors_)
```

Support Vector Machines en Scikit Learn

```
plot_svm(model)

print( "Vectores de soporte:\n", \
  model.support_vectors_)
```

Cambie el número de muestras y repita

Datos menos separables

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets.samples_generator import \
    make_blobs

X, y = make_blobs(n_samples=100, centers=2, \
    random_state=0, cluster_std=1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, \
    cmap='autumn');
```

Datos menos separables

```
from sklearn.svm import SVC
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
plot_svm(model)
```

Datos menos separables

```
from sklearn.svm import SVC
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
plot_svm(model)
```

Cambie C por 1000, 10 y 0.1

C: parámetro de penalización

- Si datos no son linealmente separables, el problema no tiene solución

C: parámetro de penalización

- Si datos no son linealmente separables, el problema no tiene solución
- Permitir un error para poder analizar datos no linealmente separables

C: parámetro de penalización

- Si datos no son linealmente separables, el problema no tiene solución
- Permitir un error para poder analizar datos no linealmente separables
- El error se penaliza con el parámetro C

C: parámetro de penalización

- Si datos no son linealmente separables, el problema no tiene solución
- Permitir un error para poder analizar datos no linealmente separables
- El error se penaliza con el parámetro C
- A mayor C menos tolerancia con el error

C: parámetro de penalización

- Si datos no son linealmente separables, el problema no tiene solución
- Permitir un error para poder analizar datos no linealmente separables
- El error se penaliza con el parámetro C
- A mayor C menos tolerancia con el error
- Suaviza el margen

Kernel SVM

Datos no Linealmente Separables

```
import matplotlib.pyplot as plt
from sklearn.datasets.samples_generator \
  import make_circles
X, y = make_circles(100, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, \
  cmap='autumn')
```

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características
- Proyección en un espacio de más dimensiones

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características
- Proyección en un espacio de más dimensiones
- Ejemplo:

$$z = \exp(-(x_1^2 + x_2^2))$$

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características
- Proyección en un espacio de más dimensiones
- Ejemplo:

$$z = \exp(-(x_1^2 + x_2^2))$$

- Intentar separar los datos linealmente en el espacio de mayor dimensión

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características
- Proyección en un espacio de más dimensiones
- Ejemplo:

$$z = \exp(-(x_1^2 + x_2^2))$$

- Intentar separar los datos linealmente en el espacio de mayor dimensión
- Transformación de las características $\phi(x)$

Proyección en un Espacio de más Dimensiones

- Datos no separables en el espacio de las características
- Proyección en un espacio de más dimensiones
- Ejemplo:

$$z = \exp(-(x_1^2 + x_2^2))$$

- Intentar separar los datos linealmente en el espacio de mayor dimensión
- Transformación de las características $\phi(x)$
- Nuevo problema: encontrar lo parámetros de la función y

$$y(x) = w^T \phi(x) + w_0$$

Proyección en un Espacio de más Dimensiones

```
z = np.exp(-(X ** 2).sum(1))
ax = plt.subplot(projection='3d')
ax.scatter3D(X[:, 0], X[:, 1], z, c=y, s=50, \
  cmap='autumn')
ax.view_init(elev=30, azim=30)
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('z')
```

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?
- Espacios de características de muchas dimensiones

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?
- Espacios de características de muchas dimensiones
- Kernel: construir proyecciones a partir de relaciones entre pares de puntos

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?
- Espacios de características de muchas dimensiones
- Kernel: construir proyecciones a partir de relaciones entre pares de puntos
- Ejemplo: permitir que cada punto sea el centro de la función de base radial

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?
- Espacios de características de muchas dimensiones
- Kernel: construir proyecciones a partir de relaciones entre pares de puntos
- Ejemplo: permitir que cada punto sea el centro de la función de base radial
- Escoger la mejor transformación de una familia

Proyección en un Espacio de más Dimensiones

- ¿Cuál proyección $\phi(\cdot)$ usar?
- Espacios de características de muchas dimensiones
- Kernel: construir proyecciones a partir de relaciones entre pares de puntos
- Ejemplo: permitir que cada punto sea el centro de la función de base radial
- Escoger la mejor transformación de una familia
- Scikit Learn: linear, poly, rbf, sigmoid

Kernel SVM en Scikit Learn

```
from sklearn.svm import SVC
model_kernel = SVC(kernel='rbf', C=1E6)
model_kernel.fit(X, y)
plot_svc(model_kernel)
```

Reconocimiento Facial con SVM

Reconocimiento Facial con SVM

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=60)
print(faces.target_names)
print(faces.images.shape)
```

Reconocimiento Facial con SVM

```
fig, ax = plt.subplots(3, 5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],
      xlabel=faces.target_names[faces.target[i]])
```

Reconocimiento Facial con SVM

```
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline

pca = PCA(svd_solver='randomized', n_components=150, \
  whiten=True, random_state=42)
svc = SVC(kernel='rbf', class_weight='balanced')
model = make_pipeline(pca, svc)
```

Reconocimiento Facial con SVM

```
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = \
  train_test_split(faces.data, faces.target, \
        random_state=42)
```

Reconocimiento Facial con SVM

```
from sklearn.model_selection import GridSearchCV
param_grid = {'svc__C': [1, 5, 10, 50], \
  'svc__gamma': [0.0001, 0.0005, 0.001, 0.005]}
grid = GridSearchCV(model, param_grid)
grid.fit(Xtrain, ytrain)
```

Reconocimiento Facial con SVM

```
model = grid.best_estimator_
yfit = model.predict(Xtest)
```

Reconocimiento Facial con SVM

```
fig, ax = plt.subplots(8, 5)
for i, axi in enumerate(ax.flat):
    axi.imshow(Xtest[i].reshape(62, 47), cmap='bone')
    axi.set(xticks=[], yticks=[])
    axi.set_ylabel( \
faces.target_names[yfit[i]].split()[-1], \
      color='black' if yfit[i] == ytest[i] \
                              else 'red')
```

Reconocimiento Facial con SVM

```
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(ytest, yfit)
sns.heatmap(mat.T, square=True, annot=True, \
  fmt='d', cbar=False,
  xticklabels=faces.target_names,
  yticklabels=faces.target_names)
plt.xlabel('etiqueta observada')
plt.ylabel('etiqueta predicha')
```

Árboles de Decisión y Bosques Aleatorios

Árboles de Decisión

- Decision Trees

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación
- Cada pregunta refina la clase a la que pertenece la observación

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación
- Cada pregunta refina la clase a la que pertenece la observación
- Aprendizaje de máquina:
    - Decision Trees

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación
- Cada pregunta refina la clase a la que pertenece la observación
- Aprendizaje de máquina:
    - Decision Trees
    - Escoger una dimensión y determinar un punto de corte para clasificar

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación
- Cada pregunta refina la clase a la que pertenece la observación
- Aprendizaje de máquina:
    - Decision Trees
    - Escoger una dimensión y determinar un punto de corte para clasificar
    - Continuar para refinar la clasificación

Árboles de Decisión

- Decision Trees
- Clasificación a partir de una sucesión de preguntas sobre las características de la observación
- Cada pregunta refina la clase a la que pertenece la observación
- Aprendizaje de máquina:
  - Decision Trees
  - Escoger una dimensión y determinar un punto de corte para clasificar
  - Continuar para refinar la clasificación
  - Aprende puntos de corte (preguntas) en cada dimensión

Árboles de Decisión en Scikit Learn

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=300, centers=4, \
  random_state=0, cluster_std=1.0)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, \
  cmap='rainbow')
```

Árboles de Decisión en Scikit Learn

```
from sklearn.tree import DecisionTreeClassifier
arbol = DecisionTreeClassifier()
arbol.fit(X, y)

viz_clas(arbol, X, y)
```

Árboles de Decisión en Scikit Learn

```
def viz_clas(model, X, y, cmap='rainbow'):
    plt.clf()
    ax = plt.gca()
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, \
        cmap=cmap, \
        clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
```

Árboles de Decisión en Scikit Learn

```
xx, yy = np.meshgrid(np.linspace(xlim[0], \
   xlim[1], num=200), \
   np.linspace(ylim[0], ylim[1],num=200))
Z = model.predict(np.c_[xx.ravel(), \
   yy.ravel()]).reshape(xx.shape)
   n_classes = len(np.unique(y))
ax.contourf(xx, yy, Z, alpha=0.3,
   levels=np.arange(n_classes + 1) - 0.5,
   cmap=cmap, clim=(y.min(), y.max()),
   zorder=1)
ax.set(xlim=xlim, ylim=ylim)
```

Bosques Aleatorios

- Random Forests

Bosques Aleatorios

- Random Forests
- Árboles de decisión: problemas de sobre ajuste

Bosques Aleatorios

- Random Forests
- Árboles de decisión: problemas de sobre ajuste
- Usar muchos árboles y escoger la clase más votada para cada observación

Bosques Aleatorios

- Random Forests
- Árboles de decisión: problemas de sobre ajuste
- Usar muchos árboles y escoger la clase más votada para cada observación
- Cada árbol se entrena con un subconjunto de los datos de entrenamiento

Bosques Aleatorios

- Random Forests
- Árboles de decisión: problemas de sobre ajuste
- Usar muchos árboles y escoger la clase más votada para cada observación
- Cada árbol se entrena con un subconjunto de los datos de entrenamiento
- Múltiples árboles: bosque

Bosques Aleatorios en Scikit Learn

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
tree = DecisionTreeClassifier()
bag = BaggingClassifier(tree, n_estimators=100, \
        max_samples=0.8, random_state=1)
bag.fit(X, y)
viz_clas(bag, X, y)
```

Bosques Aleatorios en Scikit Learn

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, \
    random_state=0)
model.fit(X,y)
viz_clas(model, X, y)
```

Un poquito de Scala y ML en Spark (Databricks)

Crear directorio

```
%scala
val basePath = "/tmp/mllib-persistence-example"
dbutils.fs.rm(basePath, recurse=true)
dbutils.fs.mkdirs(basePath)
```

Cargar datos de entrenamiento

```
entrenamiento = sqlContext.read.format("libsvm").\
option("numFeatures", "784").\
load("/databricks-datasets/mnist-digits/data-001/\
mnist-digits-train.txt")

entrenamiento.cache()

print("# de imagenes: %d " % entrenamiento.count())
```

Entrenar un clasificador

```
from pyspark.ml.classification import \
RandomForestClassifier
rf = RandomForestClassifier(numTrees=20)
modelo = rf.fit(entrenamiento)
```

Guardar modelo

```
basePath = "/tmp/ejemplo-mllib-persistencia"
modelo.save(basePath + "/modelo")
```

Guardar modelo

```
basePath = "/tmp/ejemplo-mllib-persistencia"
modelo.save(basePath + "/modelo")
```

Cargar modelo

```
%scala
import org.apache.spark.ml.classification
.RandomForestClassificationModel
val model = RandomForestClassificationModel
.load(basePath + "/modelo")
```

Probar el modelo

```
%scala
val test = sqlContext.read.format("libsvm")
.option("numFeatures", "784")
.load("/databricks-datasets/mnist-digits
/data-001/mnist-digits-test.txt")
```

Calcular y mostrar predicciones

```
%scala
val predicciones = model.transform(test)
display(predicciones.select("label", "prediction"))
```