Universidad Nacional de Rosario

Introducción al Aprendizaje Automatizado

Trabajo Práctico 0

Villagra Martín

26 de marzo de 2017

Introducción

Para realizar el trabajo se empleó el lenguaje *C++11*, el cual es directamente compatible con código *C*. Se utilizó la librería de álgebra lineal *Eigen*, simplificando el código.

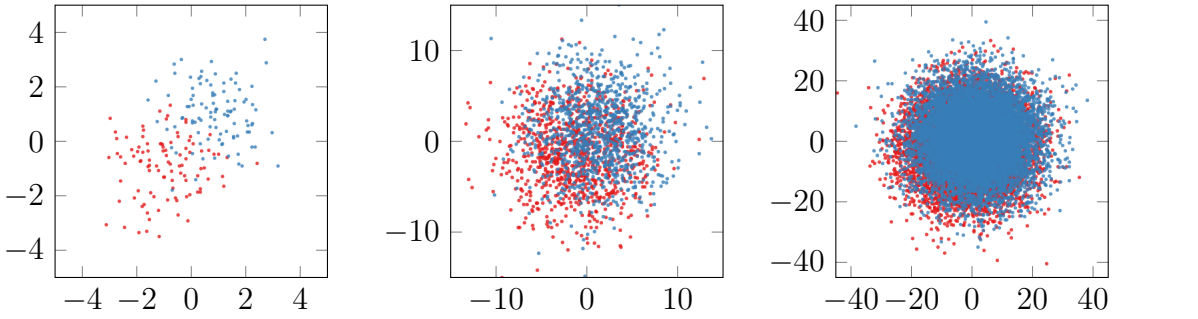
Además de los programas solicitados se creó un programa que calcula media y varianza de los datasets en formato *c4.5* y un pequeño script en *Python* para graficarlos. Con el fin de facilitar la compilación se provee un *Makefile* que compila todos los programas.

Ejercicio a

Se implementó el generador obteniendo los resultados mostrados en la Tabla 1 y en la Figura 1. Los errores mostrados son satisfactorios y disminuyen al aumentar la cantidad de puntos.

n	d	Clase	μ Esperado	μ	Error μ	σ Esperado	σ	Error σ
100	2	0	$-1{,}00$	$-0{,}95$	0,05 (5,47 %)	1,06	1,08	0,02 (1,85 %)
100	2	1	1,00	0,93	0,07 (7,13 %)	1,06	1,02	0,04 (3,81 %)
1000	4	0	$-1{,}00$	$-1{,}04$	0,04 (3,77 %)	4,00	4,05	0,05 (1,25 %)
1000	4	1	1,00	1,01	0,01 (0,61 %)	4,00	4,00	0,00 (0,07 %)
10000	8	0	$-1{,}00$	$-0{,}99$	0,01 (0,73 %)	8,49	8,49	0,00 (0,01 %)
10000	8	1	1,00	0,99	0,01 (0,76 %)	8,49	8,49	0,00 (0,05 %)

Tabla 1: Resultados del Ejercicio a. Tanto μ como σ se miden utilizando el promedio entre todas las dimensiones.



(a) $d = 2$; $n = 200$; $C = 0{,}75$ (b) $d = 4$; $n = 2000$; $C = 2{,}0$ (c) $d = 8$; $n = 20000$; $C = 3{,}0$

Figura 1: Proyección en el plano XY de los datasets generados en el Ejercicio a. Se puede observar a simple vista la diferencia entre la media de las clases.

Generación de distribución normal

Para generar con distribución normal utilizamos un Método de Aceptación-Rechazo [1].

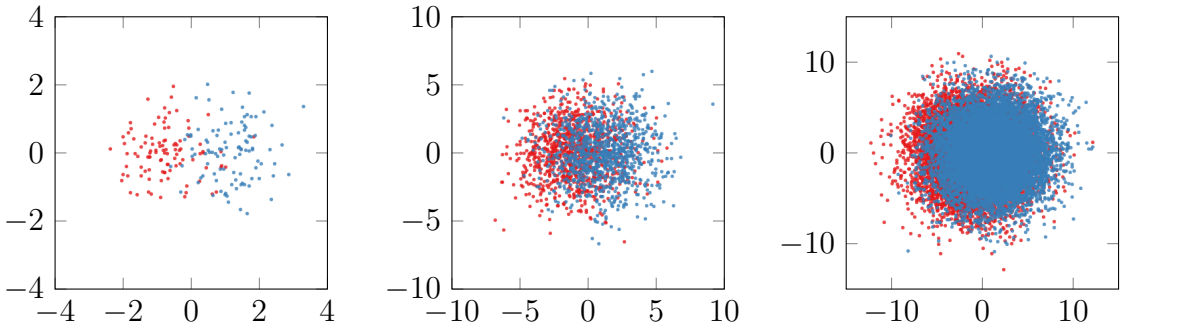
1. Generar U_1 y U_2 uniformes en $[0, 1]$. Definir $Y_1 = -\ln(U_1)$; $Y_2 = -\ln(U_2)$.
2. Si $Y_2 \geq \frac{(Y_1-1)^2}{2}$, $|Z| = Y_1$; sino volver a 1.
3. Generar U uniforme en $[0, 1]$. Definir $Z = |Z|$ si $U \leq 0{,}5$; o $Z = -|Z|$ en caso contrario. Z tiene distribución normal con $\mu = 0$ y $\sigma = 1$.

Ejercicio b

Se implementó el generador obteniendo los resultados mostrados en la Tabla 2 y en la Figura 2. Los errores mostrados son satisfactorios y disminuyen al aumentar la cantidad de puntos.

n	d	Clase	μ Esperado	μ	Error μ	σ Esperado	σ	Error σ
100	2	0	−1,00	−0,77	0,23 (23,33 %)	0,75	0,78	0,03 (3,96 %)
100	2	1	1,00	1,15	0,15 (14,82 %)	0,75	0,74	0,01 (0,84 %)
1000	4	0	−1,00	−1,03	0,03 (2,50 %)	2,00	2,03	0,03 (1,26 %)
1000	4	1	1,00	1,00	0,00 (0,47 %)	2,00	2,09	0,09 (4,53 %)
10000	8	0	−1,00	−1,03	0,03 (2,87 %)	3,00	3,03	0,03 (0,88 %)
10000	8	1	1,00	1,00	0,00 (0,06 %)	3,00	3,02	0,02 (0,69 %)

Tabla 2: Resultados del Ejercicio b. Tanto μ como σ miden solo la primera dimensión.

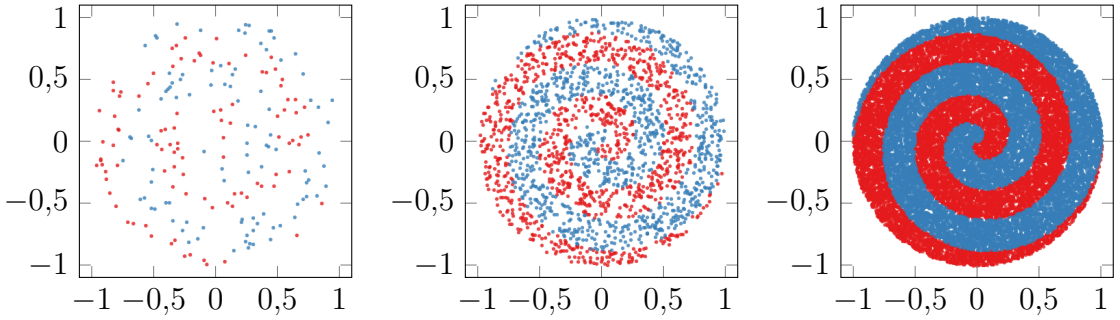


(a) $d = 2$; $n = 200$; $C = 0{,}75$ (b) $d = 4$; $n = 2000$; $C = 2{,}0$ (c) $d = 8$; $n = 20000$; $C = 3{,}0$

Figura 2: Proyección en el plano XY de los datasets generados en el Ejercicio b. Notar que las escalas son menores que en la Figura 1 por la disminución de la varianza.

Ejercicio c

La idea seguida para realizar este ejercicio fue generar puntos uniformemente dentro del círculo y luego verificar a que clase pertenecían. Los datasets generados se muestran en la Figura 3.



(a) $n = 200$ (b) $n = 2000$ (c) $n = 20000$

Figura 3: Gráfico de los datasets generados en el Ejercicio c.

Distribución uniforme dentro del círculo

Nótese que podríamos generar puntos dentro de un rectángulo circunscrito al círculo y descartar los que caen fuera del mismo. Tal algoritmo funciona pero tiene la desventaja que no se puede tener certeza de cuando terminaría.

Otra forma más eficiente para generar puntos dentro del circulo es utilizando coordenadas polares (ρ, θ), con $0 \leq \rho \leq 1$ y $0 \leq \theta \leq 2\pi$. Utilizar una distribución uniforme para generar θ es razonable pues queremos que cada ángulo tenga aproximadamente la misma cantidad de puntos. No sucede lo mismo con ρ, pues valores pequeños de ρ deberían contener menos puntos por tener cubrir una circunferencia de menor radio. Evidentemente, ρ no presenta una distribución uniforme por lo que decidimos utilizar el Teorema de Inversión [2], enunciado a continuación.

Teorema de Inversión. *Sea X una variable aleatoria con función de distribución de probabilidad acumulada F, continua e invertible, y sea F^{-1} su función inversa. Entonces, la variable aleatoria $U = F(X)$ tiene distribución uniforme en $[0, 1]$. Como consecuencia, si U es una variable aleatoria uniforme en $[0, 1]$ entonces la variable aleatoria $X = F^{-1}(U)$ satisface la distribución F.*

De las observaciones hechas anteriormente se puede suponer que la probabilidad de que un punto este a distancia ρ tiene que ser proporcional esa distancia. Por lo que si

f es la función de densidad: $f(\rho) = C\rho$. Podemos obtener C sabiendo que la integral de f de 0 a R es 1, donde R es el radio del círculo. De esta forma resulta $f(\rho) = \frac{2\rho}{R^2}$, cuya acumulada es

$$F(\rho) = \int_0^\rho f(r)\, dr = \int_0^\rho \frac{2r}{R^2}\, dr = \frac{r^2}{R^2}\bigg|_0^\rho = \frac{\rho^2}{R^2} \tag{1}$$

La inversa de esta función es $F^{-1}(u) = R\sqrt{u}$. Aplicando el Teorema de la Inversión podemos generar la distribución deseada si generamos u con distribución uniforme en el rango $[0, 1]$.

Determinación de la clase

Una vez generado un punto (θ, ρ), para determinar si pertenece a C_0 (clase 0) verificamos si el mismo pertenece a alguna de las secciones delimitadas por las curvas. Para obtener las secciones basta con evaluar las curvas en todos los ángulos de la forma $\theta + k2\pi,\ k \in \mathbb{Z}$. De esta forma tenemos

$$(\theta, \rho) \in C_0 \Leftrightarrow \exists k \in \mathbb{Z} \Big/ \frac{\theta + (2k)\pi}{4\pi} \leq \rho < \frac{\theta + (2k+1)\pi}{4\pi} \tag{2}$$

$$\Leftrightarrow \exists k \in \mathbb{Z} \Big/ 2k \leq 4\rho - \frac{\theta}{\pi} < 2k + 1 \tag{3}$$

$$\Leftrightarrow \lfloor 4\rho - \frac{\theta}{\pi} \rfloor \text{ es par} \tag{4}$$

Esta última expresión es fácilmente verificable en $O(1)$.

Referencias

[1] Karl Sigman. *Acceptance-Rejection Method*. New York, NY, USA, 2007. URL: http://www.columbia.edu/~ks20/4703-Sigman/4703-07-Notes-ARM.pdf.

[2] Luc Devroye. «The Inversion Method». En: *Non-Uniform Random Variate Generation*. New York Berlin Heidelberg Tokyo: Springer-Verlag, 1986. Cap. 2, págs. 27-28. ISBN: 0-387-96305-7. URL: http://luc.devroye.org/chapter_two.pdf.