Contents

Listado de Figuras

Listado de Tablas

Listado de código

1 Tioremas

```
print("hello world")
print("iolii mundo")
```

Listing 1: Un cacho e' código

Definición 1.1 (La mar en coche)**:** A natural number is called a *prime number* if it is greater than 1 and cannot be written as the product of two smaller natural numbers.

Observación: O sea, sería tremendo hundirse en medio de la mar Como se explica en Definición 1.1, es muy arriesgado cruzar el océano en un auto. O sea, no.

2 Notación

Ahora se ve lo que escribo?

- $\mathbb{R}$: los números reales
- d_x
- $\mathbb{R}^{d_x}$
- $[k]$, el conjunto de los k números enteros, $\{1, ..., k\}$
- $\mathcal{M}$
- $\mathbf{H}$
- $\|\cdot\|$
- $\{\mathbf{X}\}$
- $X_{i,j}$

X H

$\left|x + \frac{2}{\frac{1}{1+\theta}}\right|$

1

$\Pr(x \in A)$ $\mathbb{P}$

$\mathbb{1}\left(\frac{x}{y}\right)$

3 Preliminares

3.1 El problema de clasificación

Definición 3.1.1 (problema de clasificación):

Definición 3.1.2 (clasificador vecino más cercano):

3.1.1 Definición del problema unidimensional

[muestra aleatoria]Sea

Consideremos el problema de clasificación: [problema de clasificación] Sea $X = \left\{X_{\{1\}}, ..., X_{\{N\}}\right\}, X_{\{i\}} \in R^{\{\text{dimx}\}} \forall i \in [N]$ una muestra de N observaciones aleatorias d_x −dimensionales, repartidas en M clases $C_{\{1\}}, ..., C_{\{M\}}$ mutuamente excluyentes y conjuntamente exhaustivasfootnote{es decir, $\forall i \in [N] \equiv \{1, ..., N\}, X_{\{i\}} \in C_{\{j\}} \Leftrightarrow\Leftrightarrow X_{\{i\}}\neg \in C_{\{k\}}, k \in \mu, k \overset{\equiv}{=} j$. Asumamos además que la muestra está compuesta de observaciones independientes entre sí, y las observaciones de cada clase están idénticamente distribuidas según su propia ley: si $|C_{\{j\}}| = N_{\{j\}}$ y $X_{\{i\}}^{\{(j)\}}$representa la i-ésima observación de la clase j, resulta que $X_{\{i\}}^{\{(j)\}}\{L\}_{\{j\}}(X) \forall j \in \mu, i \in \left[N_{\{j\}}\right]$.

Dada una nueva observación $x_{\{0\}}$ cuy

3.1.2 Clasificadores “duros” y “suaves”

Definición 3.1.2.1 (clasificación dura): ¿a qué clase deberíamos asignarla?

Definición 3.1.2.2 (clasificación suave): ¿qué probabilidad tiene de pertenecer a cada clase $C_{\{j\}}, j \in [M]$?

Cualquier método o algoritmo que pretenda responder el problema de clasificación, prescribe un modo u otro de combinar toda la información muestral disponible, ponderando las N observaciones relativamente a su cercanía o similitud con $x_{\{0\}}$. Por caso, k −vecinos más cercanos (k −NN) asignará la nueva observación $x_{\{0\}}$ a la clase modal - la más frecuente - entre las k observaciones de entrenamiento más cercanasemph{ }en distancia euclídea $\texttt{norm}\left\{x_{\{0\}} - .\right\}$. k −NN no menciona explícitamente

las leyes de clase L}, lo cual lo mantiene sencillo a costa de ignorar la estructura del problema.

Definición 3.1.2.3 (k-NN): k-Nearest Neighbors

3.1.3 Clasificador de Bayes

3.1.4 KDE: Estimación de la densidad por núcleos

Definición 3.1.4.1 (KDE):

3.2 La maldición de la (alta) dimensionalidad

3.2.1 NB: El clasificador "ingenuo" de Bayes

Definición 3.2.1.1 (Naïve Bayes):

3.2.1.1 Una muestra adversa

3.2.2 KDC Multivariado

Definición 3.2.2.1 (KDE Multivariado):

Definición 3.2.2.2 (KDC):

3.2.2.1 El caso 2-dimensional

3.2.2.2 Relación entre H y la distancia de Mahalanobis

3.3 La hipótesis de la variedad

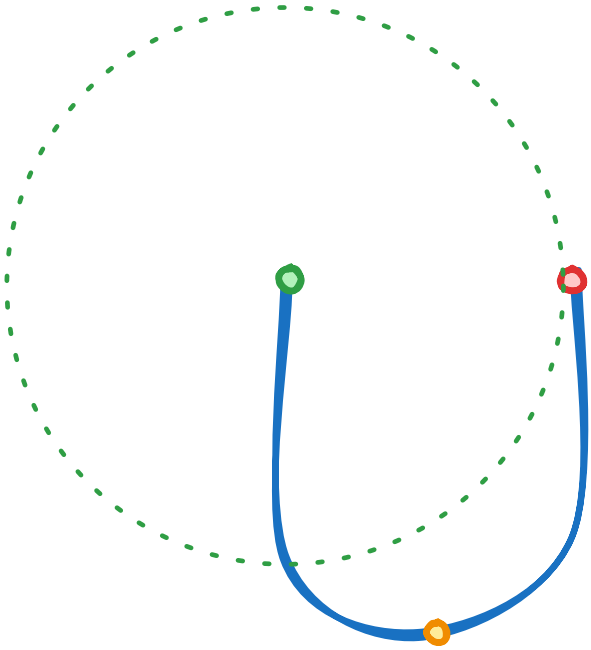

Figure 1: La variedad $\mathcal{U}$ con $\dim(\mathcal{U}) = 1$ embebida en $\mathbb{R}^2$. Nótese que en el espacio ambiente, el punto rojo está más cerca del verde, mientras que a través de $\mathcal{U}$, el punto amarillo está más próximo que el rojo

3.3.1 KDE en variedades de Riemann

3.3.2 Variedades desconocidas

3.4 Aprendizaje de distancias

3.4.1 Isomap

3.4.2 Distancias basadas en densidad

3.5 Distancia de Fermat

- Groisman & Jonckheere (Groisman, Jonckheere and Sapienza, 2019)
- Little & Mackenzie (Little, McKenzie and Murphy, 2021)
- Bijral (Bijral, Ratliff and Srebro, 2012)
- Vincent & Bengio (Vincent and Bengio, 2003)

Definición 3.5.1 (Distancia Muestral de Fermat):

4 Propuesta Original

Habiendo andado este sendero teórico, la pregunta natural que asoma es: ¿es posible mejorar un algoritmo de clasificación reemplazando la distancia euclídea por una aprendida de los datos, como la de Fermat? Para investigar la cuestión, nos propusimos:

1. Implementar un clasificador basado en estimación de densidad por núcleos (Definición 3.1.4.1) según (Loubes and Pelletier, 2008), que llamaremos "KDC". Además,
2. Implementar un estimador de densidad por núcleos basado en la distancia de Fermat, a fines de poder comparar la *performance* de KDC con distancia euclídea y de Fermat.

Nótese que el clasificador enunciado al inicio (k-NN, Definición 3.1.2.3), tiene un pariente cercano, $\varepsilon - \text{NN}$

Definición 4.1 ($\varepsilon - \text{NN}$):

Definición 4.1 es esencialmente equivalente a KDC con un núcleo "rectangular", $k(t) = \frac{\mathbb{1}(d(x,t)<\varepsilon)}{\varepsilon}$, pero su definición es considerablemente más sencilla. Luego, propondremos también

3. Implementar un clasificador cual Definición 3.1.2.3, pero con distancia muestral de Fermat en lugar de euclídea.

4.1 KDC con Distancia de Fermat Muestral

4.2 f-KNN

5 Evaluación

Nos interesa conocer en qué circunstancias, si es que hay alguna, la distancia muestral de Fermat provee ventajas a la hora de clasificar por sobre la distancia euclídea. Además, en caso de existir, quisiéramos en la medida de lo posible comprender por qué (o por qué no) es que tal ventaja existe. A nuestro entender resulta imposible hacer declaraciones demasiado generales al respecto de la capacidad del clasificador: la cantidad de *datasets* posibles, junto con sus *configuraciones de evaluación* es tan densamente infinita como lo permita la imaginación del evaluador. Con un ánimo exploratorio, nos proponemos explorar la *performance* de nuestros clasificadores basados en distancia muestral de Fermat en algunas *tareas* puntuales.

5.1 Métricas de *performance*

En tareas de clasificación, la métrica más habitual es la *exactitud*[1]

Definición 5.1.1 (exactitud): Sean $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{n\times p} \times \mathbb{R}^n$ una matriz de n observaciones de p atributos y sus clases asociadas. Sea además $\hat{\mathbf{y}} = \mathrm{R}(\mathbf{X})$ las predicciones de clase resultado de una regla de clasificación R. La *exactitud* (exac) de R en $\mathbf{X}$ se define como la proporción de coincidencias con las clases verdaderas $\mathbf{y}$:

$$\mathrm{exac}(\mathrm{R}|\ \mathbf{X}) = n^{-1} \sum_{i=1}^{n} \mathbb{1}(\hat{y}_i = y_i)$$

La exactitud está bien definida para cualquier clasificador que provea una regla *dura* de clasificación, segun Definición 3.1.2.1. Ahora bien, cuando un clasificador provee una regla *suave* (Definición 3.1.2.2), la exactitud como métrica "pierde información": dos clasificadores binarios que asignen respectivamente 0.51 y 1.0 de probabilidad de pertenecer a la clase correcta a todas als observaciones tendrán la misma exactitud, 100%, aunque el segundo es a las claras mejor. A la inversa, cuando un clasificador erra al asignar la clase: ¿lo hace con absoluta confianza, asignando una alta probabilidad a la clase equivocada, o con cierta incertidumbre, repartiendo la masa de probabilidad entre varias clases que considera factibles?

Una métrica natural para evaluar una regla de clasificación suave, es la *verosimilitud* (y su logaritmo) de las predicciones.

Definición 5.1.2 (verosimilitud): Sean $(\mathbf{X}, \mathbf{y}) \in \mathbb{R}^{n\times p} \times \mathbb{R}^n$ una matriz de n observaciones de p atributos y sus clases asociadas. Sea además $\hat{\mathbf{Y}} = \mathcal{R}(\mathbf{X}) \in \mathbb{R}^{n\times k}$ la matriz de probabilidades de clase resultado de una regla suave de clasificación $\mathcal{R}$. La *verosimilitud* (vero) de $\mathcal{R}$ en $\mathbf{X}$ se define como la probabilidad conjunta que asigna $\mathcal{R}$ a las clases verdaderas $\mathbf{y}$:

$$L(\mathcal{R}) = \mathrm{vero}(\mathcal{R}|\ \mathbf{X}) = \Pr(\hat{\mathbf{y}} = \mathbf{y}) = \prod_{i=1}^{n} \Pr(\hat{y}_i = y_i) = \prod_{i=1}^{n} \hat{\mathbf{Y}}_{(i,y_i)}$$

Por conveniencia, se suele considerar la *log-verosimilitud promedio*,

$$\ell(\mathcal{R}) = n^{-1} \log(L(\mathcal{R})) = n^{-1} \sum_{i=1}^{n} \log\left(\hat{\mathbf{Y}}_{(i,y_i)}\right)$$

[1] Más conocida por su nombre en inglés, *accuracy*.

Definición 5.1.3 (R^2 de McFadden): Sea $\mathcal{R}_0$ el clasificador "nulo", que asigna a cada observación y posible clase, la frecuencia empírica de clase encontrada en la muestra de entrenamiento $\mathbf{X}_{\text{train}}$. Para todo clasificador suave $\mathcal{R}$, definimos el R^2 de McFadden como

$$R^2(\mathcal{R}|\ \mathbf{X}) = 1 - \frac{\ell(\mathcal{R})}{\ell(\mathcal{R}_0)}$$

Observación: $R^2(\mathcal{R}_0) = 0$. A su vez, para un clasificador perfecto $\mathcal{R}^\star$ que otorgue toda la masa de probabilidad a la clase correcta, tendrá $L(\mathcal{R}^\star) = 1$ y log-verosimilitud igual a 0, de manera que $R^2(\mathcal{R}^\star) = 1 - 0 = 1$.

Sin embargo, un clasificador *peor* que $\mathcal{R}_0$ en tanto asigne bajas probabilidades a las clases correctas, puede tener un R^2 infinitamente negativo.

Visto y considerando que tanto $\mathcal{F}$-KDC como $\mathcal{F}$-kNN son clasificadores suaves, evaluaremos su comportamiento en comparación con ambas métricas, la exactitud y el R^2 de McFadden[2]

5.2 Algoritmos de referencia

Además de medir qué (des)ventajas otorga el uso de una distancia aprendida de los datos en la tarea de clasificación, quisiéramos entender (a) por qué sucede, y (b) si tal (des)ventaja es significativa en el amplio abanico de algoritmos disponibles. Pírrica victoria sería mejorar con la distancia de Fermat la *performance* de cierto algoritmo, para encontrar que aún con la mejora, el algoritmo no es competitivo en la tarea de referencia.

Consideraremos a modo de referencia los siguientes algoritmos:

- Naive Bayes Gaussiano (Definición 3.2.1.1, `GNB`),
- Regresión Logistica (`LR`) y
- Clasificador de Soporte Vectorial (`SVC`)

Esta elección no pretende ser exhaustiva, sino que responde a un "capricho informado" del investigador. `GNB` es una elección natural, ya que es la simplificación que surge de asumir independencia en las dimensiones de $\{\mathbf{X}\}$ para KDE multivariado (Definición 3.2.2.1), y se puede computar para grandes conjuntos de datos en muy poco tiempo. `LR` es "el" método para clasificación binaria, y su extensión a múltiples clases no es particularmente compleja: para que sea mínimamente valioso un nuevo algoritmo, necesita ser al menos tan bueno como `LR`, que tiene ya más de 65 años en el campo (TODO REF bliss1935, cox1958). Por último, fue nuestro deseo incorporar algún método más cercano al estado

[2]de aquí en más, R^2 para abreviar

del arte. A tal fin, consideramos incorporar alguna red neuronal (TODO REF), un método de *boosting* (TODO REF) y el antedicho clasificador de soporte vectorial, `svc`. Finalmente, por la sencillez de su implementación dentro del marco elegido[3] y por la calidad de los resultados obtenidos, decidimos incorporar `svc`, en dos variantes: con núcleos (*kernels*) lineales y RBF.

5.2.1 Uno complejo: SVC

Definición 5.2.1.1 (clasificador por sporte vectorial):

5.2.2 Uno conocido: LR - tal vez?

Definición 5.2.2.1 (regresión logística multinomial):

5.3 Metodología

La unidad de evaluación de los algoritmos a considerar es una `Tarea`, que se compone de:

- un *diccionario de algoritmos* a evaluar en condiciones idénticas, definidas por
- un *dataset* con el conjunto de n observaciones en d_x dimensiones repartidas en k clases, $\{\mathbf{X}\}_n \in R^{n \times d_x}, \{\boldsymbol{y}\}_n \in [k]^n$,
- un *split de evaluación* $r \in (0, 1)$, que determina las proporciones de los datos a usar durante el entrenamiento $(1 - r)$ y la evaluación (r), junto con
- una *semilla* $s \in [2^{32}]$ que alimenta el generador de números aleatorios y define determinísticamente cómo realizar la división antedicha.

5.3.1 Entrenamiento de los algoritmos

La especificación completa de un clasificador, requiere, además de la elección del algoritmo, la especificación de sus *hiperparámetros*, de manera tal de optimizar su rendimiento bajo ciertas condiciones de evaluación. Para ello, se definió de antemano para cada clasificador una *grilla* de hiperparámetros: durante el proceso de entrenamiento, la elección de los "mejores" hiperparámetros se efectuó maximizando la exactitud (Definición 5.1.1) con una búsqueda exhaustiva por convalidación cruzada de 5 pliegos[4] sobre la grilla entera.

[3] Utilizamos *scikit-learn*, un poderoso y extensible paquete para tareas de aprendizaje automático en Python

[4] Conocida en inglés como *Grid Search 5-fold Cross-Validation*

5.3.2 Estimación de la variabilidad en la *performance* reportada

En última instancia, cualquier métrica evaluada, no es otra cosa que un *estadístico* que representa la "calidad" del clasificador en la Tarea a mano. A fines de conocer no sólo su estimación puntual sino también darnos una idea de la variabilidad de su performance, para cada dataset y colección de algoritmos, se entrenaron y evaluaron 16 Tareas idénticas salvo por la semilla s, que luego se usaron para estimar la varianza y el desvío estándar en la exactitud (Definición 5.1.1) y el pseudo-R^2 (Definición 5.1.3).

Cuando el conjunto de datos proviene del mundo real y por lo tanto *preexiste a nuestro trabajo*, las 16 semillas $s_1, ..., s_{16}$ fueron utilizadas para definir el split de entrenamiento/evaluación. Por el contrario, cuando el conjunto de datos fue generado sintéticamente, las semillas se utilizaron para generar 16 versiones distintas pero perfectamente replicables del dataset, y en todas se utilizó una misma semilla maestra $s^\star$ para definir el split de evaluación.

5.4 Resultados

5.4.1 Chequeo de sanidad: `blobs`

Antes de considerar ningún tipo de sofisticación, comenzamos asegurándonos que en condiciones benignas, nuestros clasificadores funcionan correctamente. La

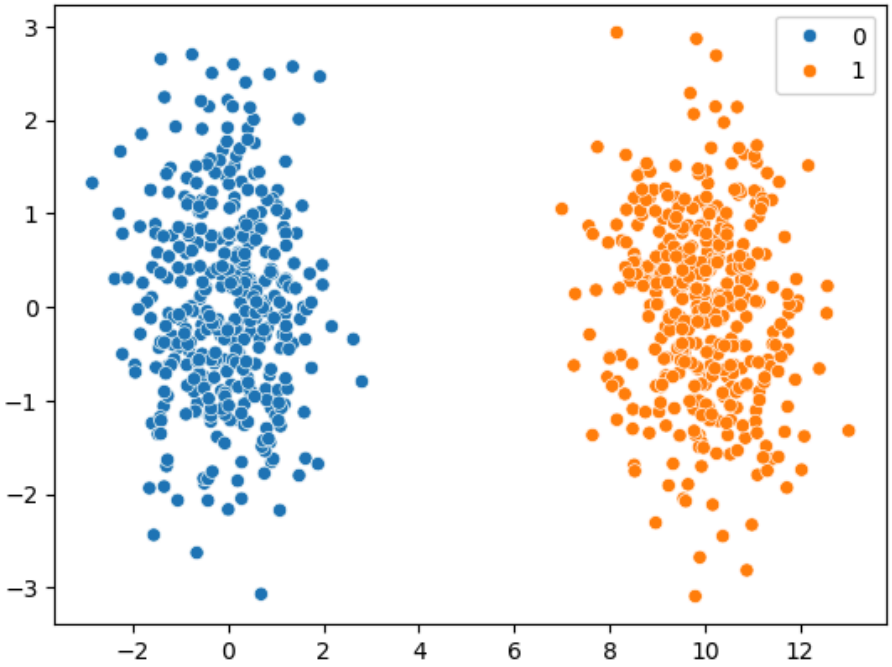

Figure 2: `make_blobs(n_features=2, centers=((0, 0), (10, 0)), random_state=1984)`

En este ejemplo, $d_{\mathcal{M}} = d_x = 2$; $k = 2$; $n_1 = n_2 = 400$ tenemos dos clases perfectamente separables, con lo cual cualquier clasificador razonable debería alcanzar exac $\approx 1, \ell \approx 0, R^2 \approx 1$. La evaluación de nuestros clasificadores resulta ser:

	ℓ	R^2	exac
$\mathcal{F}$-KDC	−0.0	1.0	1.0
KDC	−0.0	1.0	1.0
`GNB`	−0.0	1.0	1.0
$k-$NN	−0.0	1.0	1.0
$\mathcal{F}$-kNN	−0.0	1.0	1.0
`LR`	−4.5678	0.9589	1.0
`SVC`			1.0
`LSVC`			1.0
base	−111.1567	0.0	0.4625

Table 1: Resultados de entrenamiento en Figure 2

¡Excelentes noticias! Todos los clasificadores bajo estudio tienen exactitud perfecta, y salvo por una ligeramente negativa ℓ para `LR`, el resto

da exactamente 0. Pasemos entonces a algunos dataset mínimamente más complejos.

5.4.2 Datasets sintéticos baja dimensión

Consideremos ahora algunas curvas unidimensionales embebidas en $\mathbb{R}^2$:

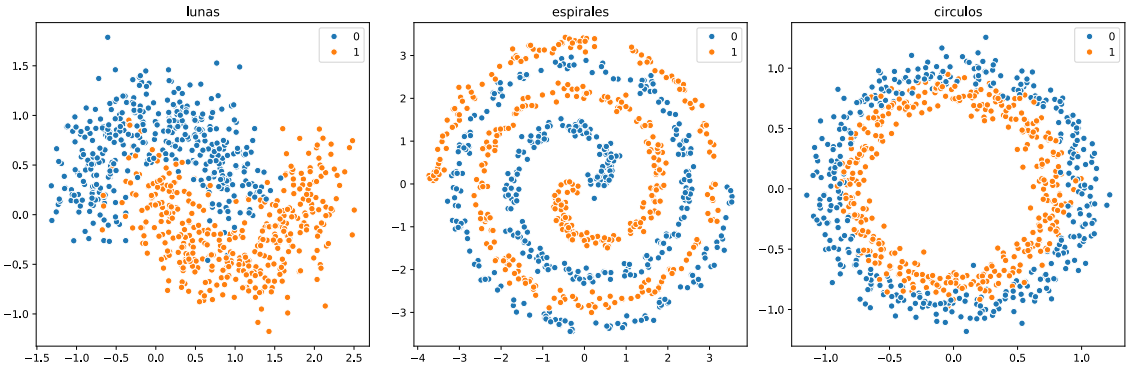

Figure 3: "Lunas", "Círculos" y "Espirales", con $d_x = 2, d_{\mathcal{M}} = 1$ y $s = 4107$

Resultará obvio al lector que los conjuntos de datos expuestos en Figure 3 no son exactamente variedades "1D" embebidas en "2D", sino que tienen un poco de "ruido blanco" agregado para incrementar la dificultad de la tarea.

Definición 5.4.2.1 (ruido blanco): Sea $X = (X_1, ..., X_d) \in \mathbb{R}^d$ una variable aleatoria tal que $E(X_i) = 0, \text{Var}(X_i) = \sigma \ \forall i \in [d]$. Llamaremos "ruido blanco con escala σ" a toda realización de X.

Veamos entonces cómo les fue a los contendientes, considerando primero la exactitud. Recordemos que para cada experimento se realizaron 16 repeticiones: en cada celda reportaremos la exactitud *promedio*, y a su lado entre paréntesis el error estándar cpte.:

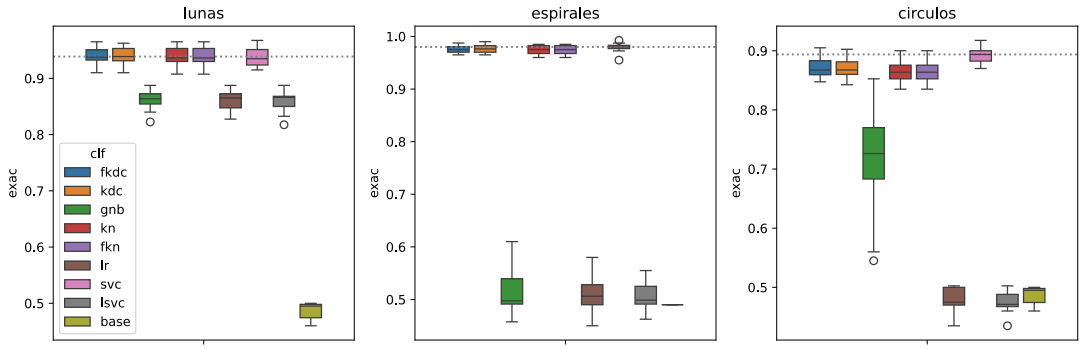

Figure 4: Boxplots con la distribución de dxactitud en las 16 repeticiones de cada experimento de Figure 3

dataset	circulos	circulos	espirales	espirales	lunas	lunas
	mean	std	mean	std	mean	std
clf						
base	48.61	1.42	49.0	0.0	48.61	1.42
fkdc	65.31	2.79	84.88	2.18	81.7	2.22
fkn	64.69	2.46	84.97	2.57	81.86	2.29
gnb	63.98	3.37	51.41	4.0	80.5	2.1
kdc	65.38	2.98	85.09	2.21	81.8	2.44
kn	64.69	2.46	84.97	2.57	81.86	2.29
lr	48.12	1.74	51.88	3.01	80.25	2.17
lsvc	47.86	1.79	50.55	2.68	80.52	1.99
svc	67.8	2.6	86.8	1.82	82.02	2.44

Table 2: "mi caption, bo".

KDC (en sus dos variantes), KNN y SVC (con kernel RBF) parecieran ser los métodos más competitivos, con mínimas diferencias de performance entre sí: sólo en "círculos" se observa un ligero ordenamiento de los métodos, `SVC` $\succ$ KDC $\succ k-$NN, aunque la performance mediana de `SVC` está dentro de "los bigotes" de todos los métodos antedichos. La tarea "lunas" pareciera ser la más fácil de todas, en la que hasta una regresión logística sin modelado alguno es adecuada. Para "espirales" y "círculos", `GNB`, `LR` y `LSVC` no logran performar significativamente mejor que el clasificador base.

Definición 5.4.2.2 (clasificador base):

¿Cómo se comparan los métodos en términos de la log-verosimilitud y el R^2?

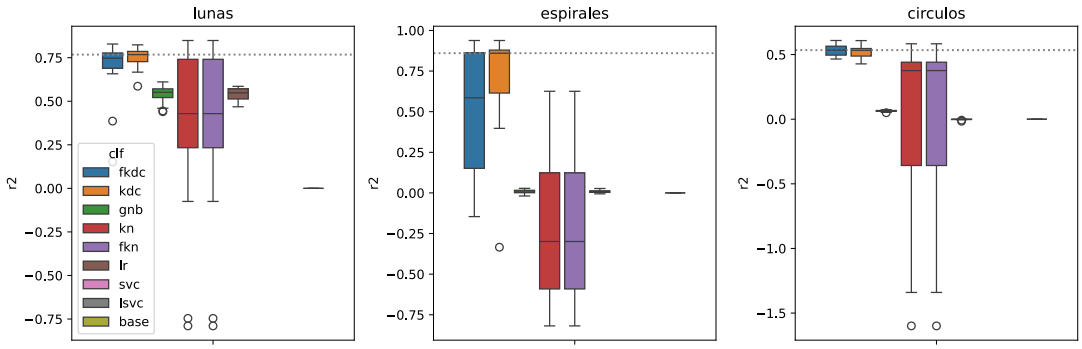

Figure 5: Boxplots con la distribución de R^2 en las 16 repeticiones de cada experimento.

dataset	circulos	circulos	espirales	espirales	lunas	lunas
	mean	std	mean	std	mean	std
clf						
base	0.0	0.0	0.0	0.0	0.0	0.0
fkdc	0.077	0.061	−0.067	1.596	0.347	0.133
fkn	−0.082	0.465	−3.081	3.123	0.299	0.153
gnb	0.05	0.009	0.007	0.012	0.384	0.045
kdc	0.094	0.026	−0.556	2.374	0.374	0.06
kn	−0.082	0.465	−3.081	3.123	0.299	0.153
lr	−0.002	0.004	0.009	0.009	0.364	0.044

Table 3: “mi caption, bo-bo”.

Como los métodos basados en máquinas de soporte vectorial resultan en clasificadores *duros* (Definición 3.1.2.1), no es posible analizar la log-verosimilitud u otras métricas derivadas. De entre los dos métodos con exactitud similar a esos, es notoriamente mejor el R^2 que alcanzan ambos KDC. A primera vista, se ve que la dispersión de la métrica es considerable, pues las “cajas” del rango intercuartil son bastante amplias, y aún así se observan *outliers*. En las tres tareas, los clasificadores de estimación de densidad por núcleos tienen las cajas más angostas y los bigotes más cortos, con KDC mostrando una dispersión menor o igual que $\mathcal{F}$-KDC. En la Figure 4, observamos que la exactitud de los métodos de k vecinos más cercanos era muy similar a la de KDC y `SVC`, sin embargo en términos de R^2,

- en el dataset de “espirales” el R^2 promedio y mediano son *negativos*, y
- en el de “círculos”, aunque la locación[5] es positiva, la distribución tiene una pesada cola a izquierda, que entra de lleno en los negativos.

En otras palabras, pareciera ser que aunque la exactitud de los métodos basados en vecinos más cercanos es buena, cuando clasifican *mal*, lo hacen con *alta seguridad*, lo que resulta en un pésimo R^2.

En esta terna inicial de *datasets*, obtenemos unos resultados aceptables:

- Observamos que el clasificador de (Loubes and Pelletier, 2008) es competitivo (es decir que no sólo Loubes y Pelletier propusieron),
- aunque la distancia de Fermat muestral no parece mejorar significativamente la exactitud de los calsificadores en ella basados.

Que la bondad de los clasificadores *no empeore* con el uso de $D_{Q,\alpha}$ en lugar de $\|\cdot\|_2$ es importante. Por una parte, cuando $\alpha = 1$ y $n \to \infty$, $D_{Q,\alpha} \to \mathcal{D}_{f,\beta} = \|\cdot\|_2$, con lo cual $\mathcal{F}$-KDC debería performar al

[5]Entendemos tanto al *promedio* o *media* y la *mediana* como *medidas de locación*

menos tan bien como KDC cuando la grilla de hiperparámetros en la que lo entrenamos incluye a $\alpha = 1$. Sin embargo, el cómputo de $D_{Q,\alpha}$ es numéricamente bastante complejo, y bien podríamos haber encontrado dificultades computacionales[6].

5.4.2.1 Comparación entre KDC y $\mathcal{F}$-KDC para `("circulos", 4479)`

Concentrémosnos en un segundo en una corrida específica de un ecperimento particular. Por caso, tomemos el dataset "circulos", con la semilla 4479. Los parámetros óptimos de $\mathcal{F}$-KDC resultaron ser `(alpha: 1.1875, bandwidth: 0.0562)` , mientras que los de KDC fueron `(bandwidth: 0.1202)`. Los anchos de banda son diferentes, y el α óptimo encontrado por $\mathcal{F}$-KDC es distinto de 1. Sin embargo, la exactitud de $\mathcal{F}$-KDC fue 0.885, y la de KDC, 0.88, prácticamente idénticas[7]. ¿Por qué? ¿Será que los algoritmos no son demasiado sensibles a los hiperparámetros elegidos?

Recordemos que la elección de hiperparámetros se hizo con una búsqueda exhaustiva por convalidación cruzada de 5 pliegos. Por lo tanto, *durante el entrenamiento* se generaron suficientes datos como para graficar la exactitud promedio en los pliegos, en función de (α, h). A esta función de los hiperparámetros a una función de pérdida[8] se la suele denominar *superficie de pérdida*.

[6] De hecho, hubo montones de ellas, cuya resolución progresiva dio lugar a la pequeña librería que acompaña esta tesis y documentamos en el anexo. A mi entender, ningún error de cálculo persiste en el producto final

[7] Con 400 observaciones para evaluación, dichos porcentajes representan 352 y 354 observaciones correctamente clasificadas, resp.

[8] En realidad, la exactitud es un "score" o puntaje - mientras más alto mejor-, pero el negativo de cualquier puntaje es una pérdida - mientras más bajo, mejor.

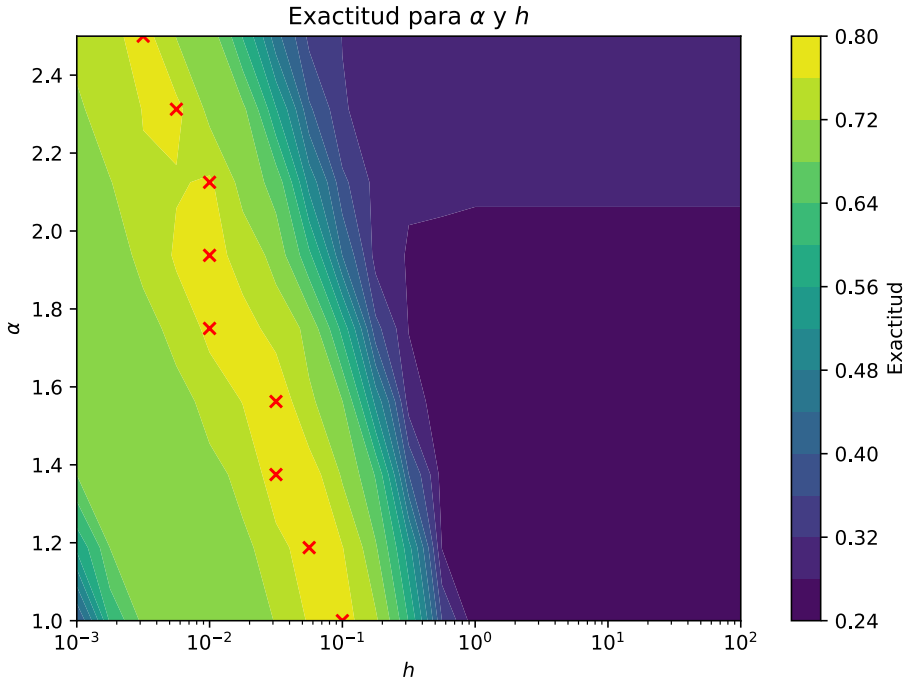

Figure 6: Exactitud promedio en entrenamiento para la corrida `("circulos", 4479)`. Las cruces rojas indican la ventana h óptima para cada α.

Nótese que la región amarilla, que representa los máximos puntajes durante el entrenamiento, se extiende diagonalmente a través de todos los valores de α. Es decir, no hay un *par* de hiperparámetros óptimos $(\alpha^\star, h^\star)$, sino que fijando α, siempre pareciera existir un(os) $h^\star(\alpha)$ que alcanza (o aproxima) la máxima exactitud *posible* con el método en el dataset. En este ejemplo en particular, hasta pareciera ser que una relación log-lineal captura bastante bien el fenómeno, $\log(h^\star) \propto \alpha$. En particular, entonces, $\text{exac}(h^\star(1), 1) \approx \text{exac}(h^\star, \alpha^\star)$, y se entiende que el algoritmo $\mathcal{F}$-KDC, que agrega el hiperparámetro α a KDC no mejore significativamente su exactitud.

Ahora bien, esto es sólo en *un* dataset, con *una* semilla especfíca. ¿Se replicará el fenómeno en los otros datasets estudiados? Y si tomásemos datasets con otras características?

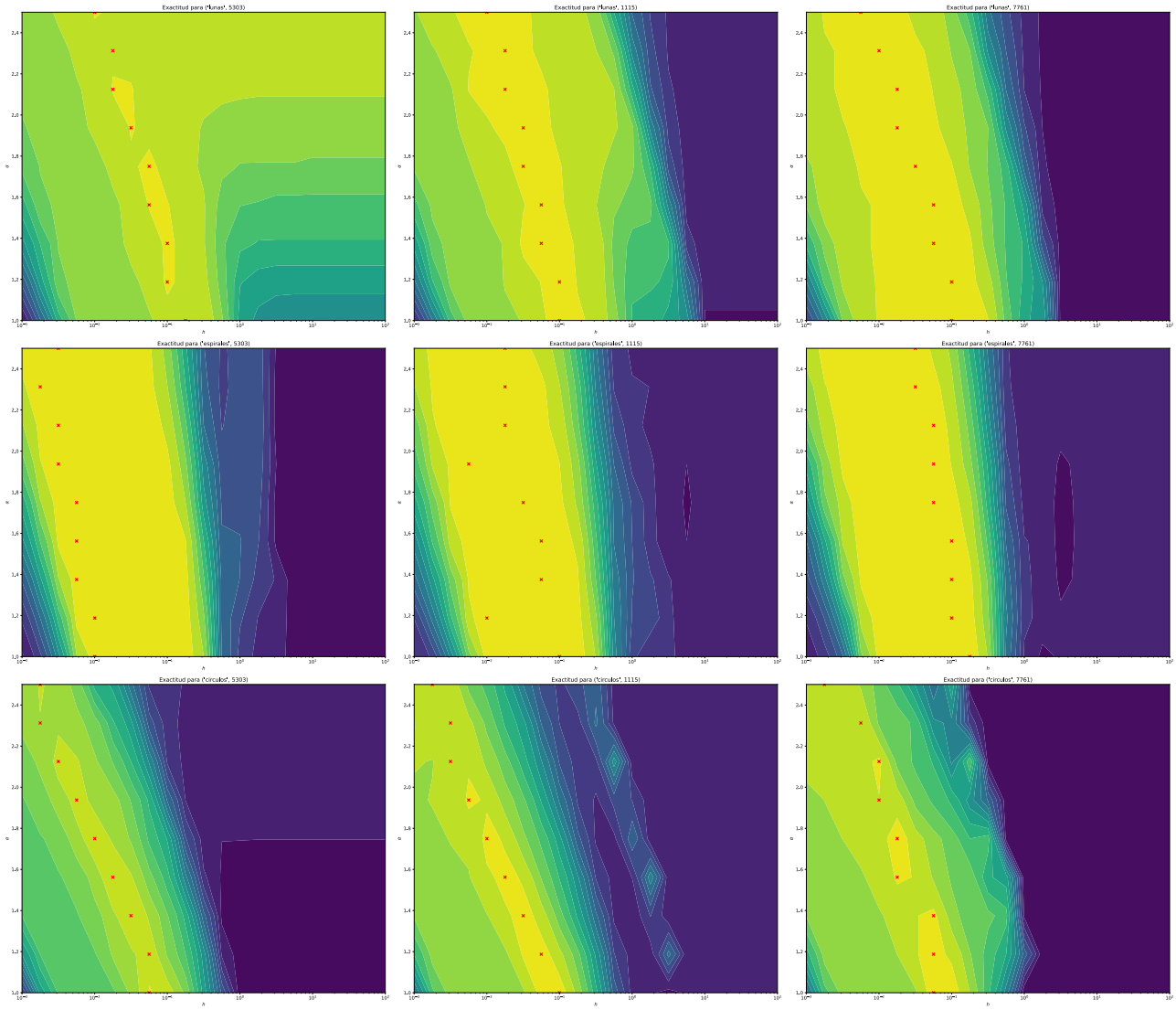

Figure 7: It does replicate

Antes de avanzar hacia el siguiente conjunto de datos, una pregunta más: ¿qué sucede si aumentamos el nivel de ruido? Es decir, mantenemos los dataset hasta aquí considerados, pero subimos σ de Definición 5.4.2.1?

Esto digo yo

(Rosenblatt, 1956; Chacón and Duong, 2013; Carpio *et al.*, 2019)

Bibliography

Bijral, A. S., Ratliff, N. and Srebro, N. (2012) "Semi-Supervised Learning with Density Based Distances." arXiv. Available at: https://doi.org/10.48550/arXiv.1202.3702

Carpio, A. *et al.* (2019) *Fingerprints of Cancer by Persistent Homology*. Available at: https://doi.org/10.1101/777169

Chacón, J. E. and Duong, T. (2013) "Data-Driven Density Derivative Estimation, with Applications to Nonparametric Clustering and Bump Hunting," *Electronic Journal of Statistics*, 7(none). Available at: https://doi.org/10.1214/13-EJS781

Groisman, P., Jonckheere, M. and Sapienza, F. (2019) "Nonhomogeneous Euclidean First-Passage Percolation and Distance Learning." arXiv

Little, A., McKenzie, D. and Murphy, J. (2021) "Balancing Geometry and Density: Path Distances on High-Dimensional Data." arXiv

Loubes, J.-M. and Pelletier, B. (2008) "A Kernel-Based Classifier on a Riemannian Manifold," *Statistics & Decisions*, 26(1), pp. 35–51. Available at: https://doi.org/10.1524/stnd.2008.0911

Rosenblatt, M. (1956) "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, 27(3), pp. 832–837

Vincent, P. and Bengio, Y. (2003) "Density Sensitive Metrics and Kernels," in *Proceedings of the Snowbird Workshop*