Aprendizaje Automático – Guía de Ejercicios [*]

Departamento de Computación – FCEyN
Universidad de Buenos Aires

Primer cuatrimestre 2024
Versión: 10 de junio de 2024

Versiones

- 5 de abril:
  1. Agregados dos items en el Ejercicio 3.7
  2. Agregada las secciones 4 (selección, validación, evaluación) y 5 (métricas)
- 12 de abril: Agregada la sección 6
- 15 de abril: Corrección de typo en el ejercicio 6.2 (2)
- 19 de abril: Agregada las secciones 7 y 8
- 30 de abril: Agregada la sección 9 y corregido ejercicio 6.6
- 17 de mayo: Agregada la sección 10
- 22 de mayo: Agregada la sección 11 y 12
- 30 de mayo: Agregada la sección 13
- 10 de junio: Agregada la sección 14

1. Repaso probabilidad y estadística

Para resolver los siguientes ejercicios recomendamos leer el **Capítulo 6** del libro Mathematics for Machine Learning de Deisenroth, Faisal y Soon Ong. `https://mml-book.github.io/book/mml-book.pdf`

Ejercicio 1.1. Explique por qué los siguientes eventos son independientes de a pares pero no independientes entre todos. Dadas 2 monedas,

(a) la primera moneda es cara;

(b) la segunda moneda cara;

(c) las dos monedas son iguales.

Ejercicio 1.2. Demostrar el teorema de Probabilidad Total: dados una partición $\{A_i\}_{i=1}^n$ del espacio muestral tal que $P(A_i) > 0$ para todo i, y un evento B:

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i) \cdot P(A_i)$$

Ejercicio 1.3.

(a) Sugerencia, mirar este video: `https://www.youtube.com/watch?v=HZGCoVF3YvM&t=57s`(3blue1brown Bayes)

(b) Demostrar el teorema de Bayes: dados dos eventos A y B tal que $P(B) > 0$,

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

[*] Algunos ejercicios fueron adaptados de los libros "Machine Learning", de Tom Mitchell (McGraw-Hill, 1997); "Pattern Recognition and Machine Learning", de Christopher Bishop (Springer, 2006); y "An Introduction to Statistical Learning", de James, Witten, Hastie & Tibshirani (Springer, 2015).

(c) Un local vende dos marcas de televisores, A y B. El 40 % de los televisores vendidos son de la marca A y un 30 % de ellos tienen un defecto. Por otro lado, el 20 % de los televisores vendidos son de la marca B y el 10 % tienen un defecto. Si un televisor tiene un defecto, ¿cuál es la probabilidad de que sea de la marca A?

(d) Supongamos que tienes dos máquinas, A y B, que producen tornillos. La longitud de los tornillos producidos por cada máquina sigue una distribución normal. Se sabe que la máquina A produce tornillos con una longitud media de 10 cm y una desviación estándar de 0.5 cm, mientras que la máquina B produce tornillos con una longitud media de 10.5 cm y una desviación estándar de 0.7 cm.

Ahora, supongamos que se selecciona un tornillo al azar de la producción combinada de ambas máquinas y se encuentra que tiene una longitud de 10.2 cm. ¿Cuál es la probabilidad de que este tornillo provenga de la máquina B?

Ejercicio 1.4.

(a) Explicar con tus palabras qué es la media y qué es el desvío estándar.

(b) En una fábrica de producción de caramelos, se mide la longitud de los caramelos producidos. Si la longitud media es de $5cm$ y un desvío de $0.05cm^2$. Siendo que tener un caramelo de más de 5.05 cm o menos de 4.95 cm se considera defectuoso, ¿qué significa esto en términos de la calidad de los caramelos producidos por esa fabrica?. ¿Hicieron alguna suposición sobre la distribución?

Ejercicio 1.5.
Has realizado un experimento en el que lanzaste una moneda 10 veces y la secuencia observada de resultados fue HHHTHTTHHT.

1. Suponiendo que la moneda es justa (es decir, la probabilidad de que salga cara (P(H)) = 0,5 y la probabilidad de que salga cruz (P(T)) = 0,5), calcule la probabilidad de observar la secuencia dada.
2. Es más probable que la moneda esté sesgada hacia la cara en 70 % ó que la moneda este balanceada dados los datos observados. (HHHTHTTHHT).
3. Imaginen ahora que queremos estimar la carga de la moneda lo mejor posible dados los datos de la tirada. Plantear los pasos a seguir para encontrar este valor. Tip: están calculando máxima verosimilitud.
4. Calcularla :)
5. Sugerencia, ver: `https://www.youtube.com/watch?v=Dn6b9fCIUpM&t=195s`) (statquest)

2. Introducción

Ejercicio 2.1. Revisar y completar el notebook `notebook_1_herramientas.ipynb`.

Ejercicio 2.2. Describir para los siguientes problemas si se trata de aprendizaje supervisado o aprendizaje no supervisado. Especificar qué medida de performance y de un ejemplo de una base de datos que permita encarar el problema.

(a) detección de discurso de odio en tweets;

(b) recomendación de películas;

(c) diagnóstico de tumores por imágenes;

(d) autocompletar textos;

(e) segmentación comercial de clientes;

(f) autenticación biométrica (ej: huellas dactilares);

(g) detección de fraude en tarjetas de crédito.

Ejercicio 2.3. Determinar para los siguientes problemas de aprendizaje supervisado si se trata de problemas de clasificación o de regresión. Para cada caso, indique un ejemplo de instancia (el valor de sus atributos) junto a una etiqueta posible especificando el tipo de cada valor.

(a) Dado un tweet, determinar si habla en contra o a favor de un candidato presidencial.

(b) Predecir cuánto gastará una empresa en luz el próximo semestre.

(c) Dado un tweet, predecir la probabilidad de que hable en contra o a favor de un candidato.

(d) Predecir a qué distancia de la facultad vive una persona.

(e) Predecir si se gastará más o menos que $50.000 por mes de luz el próximo semestre. ¿Qué responderían si en la base de datos tenemos etiquetas reales? ¿Y qué si tuvieramos etiquetas categoricas (sí, no)?

(f) Predecir la probabilidad de que se gaste más o menos que $50.000 por mes de luz el próximo semestre.

(g) Predecir la nota que tendrá un alumno en un examen cuya nota puede ser $0, 1, 2, \ldots, 10$

(h) Predecir la nota que tendrá un alumno en un examen cuya nota puede ser "A", "R" o "I".

(i) Predecir dónde vive una persona.

(j) Predecir la próxima palabra a autocompletar dadas las oraciones anteriores.

(k) Predecir el valor que tomará el dolar en los próximos diez días.

Ejercicio 2.4. Sea un problema de clasificación en el cual cada instancia tiene 2 atributos numéricos (coordenadas x e y) y pertenece a una de dos clases posibles (blanco o negro).
Se tienen tres tipos de hipótesis ilustrados en la Figura que representan (a) rectas, (b) líneas verticales (hasta 30 líneas), (c) elipses (tantas como se quiera).
Para cada uno de ellos, se pide:

- Describir el espacio de hipótesis H;
- Identificar los parámetros de la hipótesis (el conjunto de valores que permiten describir una hipótesis en concreto, θ).[1]

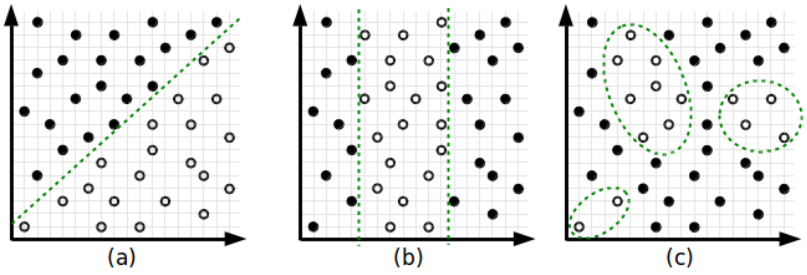


Figura 1: Tipos de Hipótesis

Ejercicio 2.5. Completar el notebook `notebook_2_titanic.ipynb` .

[1] "Machine Learning", de Tom Mitchell (McGraw-Hill, 1997);

3. Árboles de decisión

Cielo	Temperatura	Humedad	Viento	¿Salgo? (clase a predecir)
Sol	Calor	Alta	Débil	No
Sol	Calor	Alta	Fuerte	No
Nublado	Calor	Alta	Débil	Sí
Lluvia	Templado	Alta	Débil	Sí
Lluvia	Frío	Normal	Débil	Sí
Lluvia	Frío	Normal	Fuerte	No
Nublado	Frío	Normal	Fuerte	Sí
Sol	Templado	Alta	Débil	No
Sol	Frío	Normal	Débil	Sí
Lluvia	Templado	Normal	Débil	Sí
Sol	Templado	Normal	Fuerte	Sí
Nublado	Templado	Alta	Fuerte	Sí
Nublado	Calor	Normal	Débil	Sí
Lluvia	Templado	Alta	Fuerte	No

Tabla 1: Salgo a caminar

Ejercicio 3.1. Hacer en papel y lápiz un árbol de decisión correspondiente a entrenar con los datos de la Tabla Salgo a caminar. Utilizar el criterio "Gini Gain" para calcular el feature que mejor separa cada decisión. Armar luego tres ejemplos posibles de instancias nuevas (no existentes en la tabla) y usar el árbol para predecir la clase de salida. Calcular además la importancia de cada atributo (decrecimiento total en la impureza Gini).

Ejercicio 3.2. ¿Cómo cambiaría el árbol si restringiéramos su altura a 2 niveles?

(a) ¿Cuál sería el resultado de las predicciones del ejercicio anterior?

(b) ¿Algunas de las instancias existentes en la tabla, serían clasificadas incorrectamente?

Ejercicio 3.3. ¿Quién es quién?
El quién es quién es un juego (buscar en Google) en el que hay que adivinar el nombre del personaje del rival. Para ello se hacen preguntas que son respondidas por sí o por no.

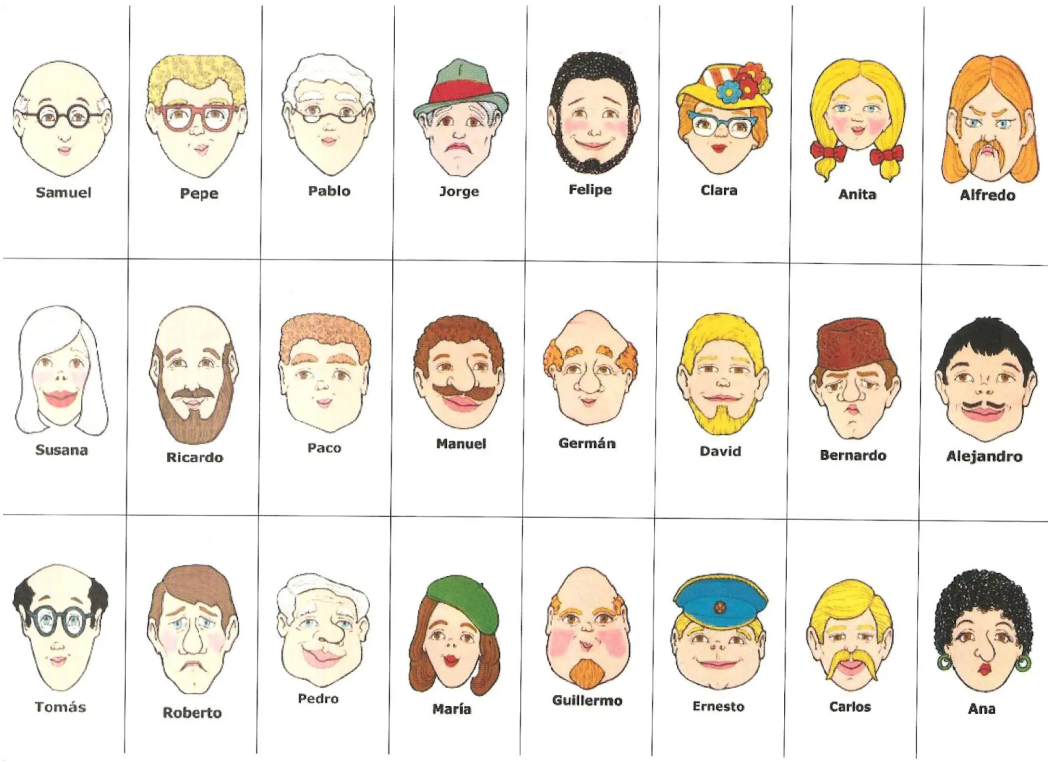


Figura 2: Juego de ¿quién es quién?

Tomemos la libertad de jugar únicamente a adivinar el sexo biológico (masculino/femenino).

(a) Si preguntamos si el personaje es calvo... ¿Cuál es la ganancia gini al separar por este atributo?

(b) Ordenar por valor de ganancia gini qué conviene como primera pregunta: i) ¿Tiene calvicie? ii) ¿Tiene cabello rubio? iii) ¿Tiene vello facial? (bigote y/o barba) iv) ¿Tiene sombrero?

Ejercicio 3.4. En la Figura Cortes en el espacio de atributos puede verse diversas regiones en el espacio de atributos.

- Determinar cuáles de ellas pueden haber sido generadas por árboles de decisión.
- Para las que lo sean, mostrar un árbol que hubiese generado estas regiones (suponer ejes x_1 y x_2)

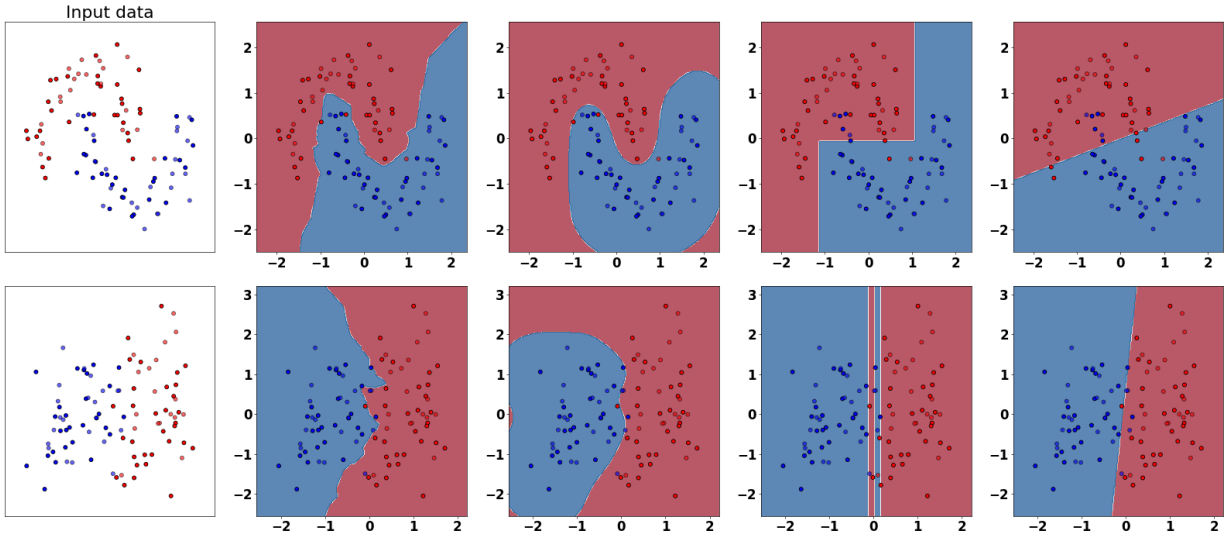
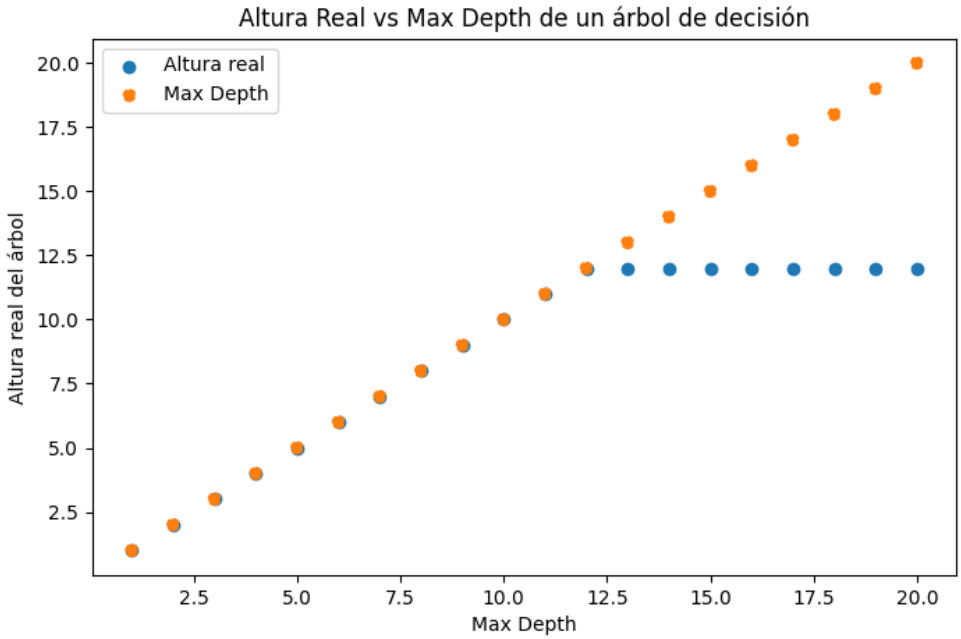


Figura 3: Cortes en el espacio de atributos

Ejercicio 3.5. Se entrenan 20 árboles distintos variando el parámetro max_{depth} desde 1 hasta 20 inclusive. Graficamos, para cada uno de estos árboles, el valor de max_{depth} y el valor de la altura real de estos. Responder:



- ¿Por qué sucede que, a partir de max_depth igual a 12, la altura real del árbol es constante?
- Dado n instancias, ¿cuál seria el valor máximo de max_depth que tiene sentido utilizar?

Ejercicio 3.6. Dado el algoritmo de construcción de árboles visto en clase para atributos continuos:
Sea S una muestra de instancias con atributos A. Para construir un árbol de decisión ejecutamos:

(1) Mientras no se cumpla un criterio de detención:

   (I) Creamos `nodo_actual`, un nodo que representa una decisión de corte.

   (II) Elegimos el par $a \in A$, $c \in R$ entre los posibles pares $\langle$ atributo, corte$\rangle$, que mejor divida a S para `nodo_actual` según $\Delta M(S, \langle a, c \rangle)$
   (ΔM es una función que devuelve cuánto gano si divido a S en dos utilizando $\langle a, c \rangle$ y según la medida M. Por ejemplo ΔM podría ser `GananciaDeInformación`).

   (III) Crear dos hijos de `nodo_actual`.

   (IV) Dividir las instancias de S en los nuevos nodos, según $\langle a, c \rangle$:

$$S_{\leq} \leftarrow \{x | x \in S \wedge x[a] \leq c\}$$
$$S_{>} \leftarrow \{x | x \in S \wedge x[a] > c\}$$

   (V) Repetir para cada hijo.

(2) El valor asignado a cada región resultante (a cada hoja) será el de la clase mayoritaria de las instancias que pertenezcan a esta región.

Se pide:

(a) Escribir el pseudocódigo (puede ser similar a python) para el paso (b) (elegir el mejor par a, c entre los posibles pares). Es decir, definir `mejor_corte`$(S, A, \Delta M)$, en donde `S` representa la muestra, `A` un conjunto de atributos y ΔM la función que computa la ganancia de una división. Puede suponer dadas funciones tales como `S[a]` que devuelve la columna de valores para el atributo a. Es importante que esta función no evalúe dos veces cortes que devuelven exactamente las mismas regiones para un mismo atributo.

(b) Introducir los cambios necesarios en el algoritmo general (y si fuera necesario, en la función que definieron para `mejor_corte`) que permita medir la importancia de cada atributo. La importancia deberá ser un valor numérico entre 0 y 1.

Ejercicio 3.7. Determinar cuáles de las siguientes son afirmaciones verdaderas:

(a) El objetivo de construir un árbol de decisión es crear el árbol de menor tamaño posible en el cual las hojas contengan valores de una sóla clase.

(b) Los algoritmos de construcción vistos (CART, ID3, etc) exploran todos los posibles árboles y se quedan con el que mejor separa a las instancias.

(c) La pureza describe qué tan cerca está un nodo de contener instancias de una sola clase.

(d) Un atributo puede aparecer sólo una vez en cada rama del árbol (llamamos rama a un camino directo desde una hoja hasta la raíz).

(e) Un par (atributo, corte) puede aparecer sólo una vez en cada rama del árbol (llamamos rama a un camino directo desde una hoja hasta la raíz).

(f) Para cada nueva instancia, un árbol permite predecir la clase a la que pertence. Por otra parte, para predecir **la probabilidad** de pertencer a una clase u otra, es necesario modificar el algoritmo de creación de árboles.

(g) Un árbol de decisión, con criterios de corte suficientemente laxos, puede siempre conseguir 100 % de aciertos en los datos de entrenamiento.

(h) Un árbol de decisión, con criterios de corte suficientemente laxos, puede siempre conseguir 100 % de aciertos en los datos de entrenamiento siempre y cuando no haya contradicciones entre las etiquetas de instancias iguales.

Ejercicio 3.8. Resolver el notebook `notebook_3_arboles_de_decision_sklearn.ipynb`.

Ejercicio 3.9. Preguntas conceptuales para discutir:

(a) ¿Cuál el sesgo inductivo del algoritmo que construye el árbol de decisión?

(b) ¿Qué sucede cuando dos atributos empatan en ganancia de información? ¿Esta decisión es parte del sesgo inductivo?

(c) ¿Cómo se comporta la ganancia de información en comparación a impureza Gini cuando se comparan atributos con gran cantidad de valores distintos? Por ejemplo, si el atributo x_1 tiene dos valores posibles (true y false) y el atributo x_2 tiene 40 valores distintos, ¿es justo usar ganancia de información para elegir entre ellos? ¿Qué desventajas tiene? ¿Cómo se podría mitigar?

(d) Dado un atributo a continuo en la cual queremos encontrar el mejor corte c según alguno de los criterios considerados (Gini o Entropy), ¿cuál es la complejidad de peor caso si se asume que tenemos n instancias en nuestro conjunto de datos?

Ejercicio 3.10. Completar el notebook `notebook_4_implementacion_arbol.ipynb`. Este notebook contiene una implementación parcial de un algoritmo de creación de árboles de decisión.

Ejercicio 3.11. Dado el algoritmo de construcción de árboles para atributos continuos (ver algoritmo en el ejercicio 3.6) ¿Qué cambios son necesarios para que funcione para problemas de regresión? Para pensar:

- ¿Pueden las medidas vistas en clase (CER, Gini, Entropía) ser utilizadas en este caso?
- ¿Qué cambiarían en el criterio de detención?
- ¿Cómo cambiarían el valor asignado a los nodos hojas?

4. Búsqueda de hiperparámetros, validación cruzada y Evaluación

Ejercicio 4.1.
Sea GRID_SEARCH(X : LIST⟨TUPLE⟨ALGORITMO, HYPERS⟩⟩, D : DATA, M : METRICA) : TUPLE⟨ALGORITMO, HYPERS, FLOAT⟩ la función que ejecuta una búsqueda exhaustiva para encontrar la mejor configuración entre una lista de posibilidades y retorna el valor esperado para dicha combinación.
Detalles de los parámetros:

- X es una lista de TUPLE⟨A, HS⟩ : TUPLE⟨ALGORITMO, HYPERS⟩, en donde A es el nombre del algoritmo y HS representa un diccionario de hiperparámetros a valores.
  Por ej, un elemento de la lista puede ser $\langle arbol\_de\_decision, \{altura\_max : 10, medida : gini, atributos : todos\}\rangle$
  otro puede ser $\langle arbol\_de\_decision, \{altura\_max : 5, medida : entropy, atributos : todos\}\rangle$
- D, un dataset de $n \times m$ (instancias $\times$ atributos)
- M, una métrica (por ejemplo, ACCURACY)

(a) Escribir el pseudocódigo de la función. Para ello, suponga ya implementada la función CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : METRICA) : FLOAT que devuelve el resultado de ejecutar validación cruzada para un algoritmo A e hiper-parámetros HS sobre la base de datos D y usando la métrica M. Retorna el valor obtenido.

(b) Escribir el pseudocódigo de la función CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : METRICA) : FLOAT. ¿Qué tipo de validación cruzada elegiste para tu pseudocódigo?

(c) Definir ahora la función RANDOMIZE_SEARCH(..., N : INT) : TUPLE⟨ALGORITMO, HYPERS, FLOAT⟩ con los mismos parámetros que GRID_SEARCH más N, un parámetro que indica la cantidad de intentos a probar. Suponer para esta función que H : HYPERS es un diccionario en donde los valores también pueden ser distribuciones probabilísticas a las que se las puede muestrear con la función SAMPLE(D: DISTRIBUCIÓN) : FLOAT (suponer que SAMPLE(CONSTANTE) = CONSTANTE. Ej, $\langle arbol\_de\_decision, \{altura\_max : Binomial(n = 20, p = 0.5), medida : entropy, atributos : todos\}\rangle$

Ejercicio 4.2. Validación cruzada. Verdadero o Falso. Justificar

(a) Hacer validación cruzada evita el sobreajuste (overfitting) de los modelos sobre los datos.

(b) Hacer validación cruzada ayuda a obtener estimaciones más realistas de la performance de un modelo sobre nuevos datos que al hacerlo sobre los mismos datos de entrenamiento.

(c) En K-fold cross validation, conviene que K se acerque a N. De esta manera el resultado será lo más realista posible ya que se tiende a generar N modelos independientes. El problema es que hay que entrenar demasiados modelos.

(d) Evaluar un modelo sobre el conjunto de evaluación (control) resultará en un valor siempre peor o igual al conseguido en desarrollo.

(e) Una vez elegido el mejor modelo durante desarrollo, es conveniente hacer k-fold cross validation nuevamente, pero ahora incluyendo también el conjunto de evaluación.

(f) Luego de seleccionar la mejor configuración, es ideal volver a correr el algoritmo pero esta vez utilizando todos los datos de desarrollo para luego evaluarlo en los datos de evaluación.

Ejercicio 4.3. Consideremos K-fold cross validation con las siguientes modificaciones:

- Para cada instancia x_i dentro de un conjunto de datos $\mathcal{D}$ se tiene un mapeo G que le asigna un único grupo $g \in \mathcal{G}$ tal que $G(x_i) = g$ (pueden pensarlo como un DICC(INSTANCIA, GRUPO)).
- Al dividir $\mathcal{D}$ en los k folds, se asegura que cada grupo g este contenido únicamente en un fold.

(a) Construir el algoritmo GROUP_K_CROSS_VAL(A : ALGORITMO, HS : HYPERS, D : DATA, M : MÉTRICA, G: GRUPOS, K: INT) : FLOAT que devuelve el resultado de evaluar el algoritmo dado con los hiperparámetros dados y algún valor de k respetando los grupos dados.

(b) ¿En que escenarios podría tener sentido utilizar este procedimiento? Desarrolle.

(c) ¿Qué cambiaría en su implementación si hubiese más folds que grupos?

Ejercicio 4.4. Explicar las posibles causas por las cuales un modelo entrenado y evaluado de la siguientes maneras puede funcionar peor que lo esperado al ser llevado a producción.

(a) Entrenado y evaluado sobre datos de entrenamiento.

(b) Seleccionado entre muchas posibilidades sobre datos de desarrollo (utilizando grid o random search junto a cross-validation).

(c) Seleccionado entre muchas posibilidades sobre datos de desarrollo (utilizando grid o random search junto a cross-validation) y evaluado en datos held-out.

Ejercicio 4.5. Preguntas para desarrollar.

(a) En la clase teórica vimos que al hacer cross validation los datos no siempre deben separarse al azar, ¿por qué?. Pensar ejemplos de al menos dos situaciones en las cuales no sea conveniente.

(b) ¿Por qué se deberían usar una sola vez los datos held-out?

(c) ¿Qué sería conveniente hacer si luego de un muy buen resultado en desarrollo, encuentro un pésimo resultado en evaluación?

Ejercicio 4.6. La técnica de SOBREMUESTREO (oversampling) consiste en crear copias de instancias al azar (a veces agregando mínimas modificaciones en alguno de sus atributos) y asignarle la etiqueta original.

Este proceso se utiliza mucho cuando las clases están muy desbalanceadas y no es suficiente la cantidad de instancias de alguna clase para entrenar un clasificador. En estos casos es normal, por ejemplo, sobremuestrear hasta lograr la misma cantidad de instancias para cada clase.

Por ejemplo, dado el problema de clasificar entre GATOS, PERROS y CONEJOS, y dado que la cantidad de instancias es 100, 2000 y 20 respectivamente, generamos un nuevo dataset de 2000, 2000, y 2000 instancias, en donde las instancias agregadas son copias de instancias de la clase correspondiente elegidas al azar y a las que se le agrega un poco de ruido a sus columnas numéricas.

Describir los pros y contras de aplicar la técnica en los siguientes pasos del proceso. Justificar en términos de posibilidades de sub o sobreestimar la performance de nuestros modelos y concluir cuál es la opción más segura:

(I) Antes de partir en desarrollo - evaluación

(II) Antes de partir en Folds en desarrollo.

(III) Luego de seleccionar los folds que se utilizarán para entrenar el modelo dentro de las iteraciones de K-fold cross val. Aplicarlo sólo a los datos de entrenamiento.

(IV) Luego de seleccionar los folds que se utilizarán para entrenar el modelo dentro de las iteraciones de K-fold cross val. Aplicarlo tanto a los datos de entrenamiento como a los de validación.

Ejercicio 4.7. Ordenar las siguientes acciones para evitar problemas (indentar si hay loops y especificar sobre qué datos se haría)

(a) Mandar reporte final a toda la empresa y submitear paper.

(b) Seleccionar la mejor configuración, entrenar con los datos completos.

(c) Partir los datos en desarrollo - evaluación

(d) Correr un clasificador super simple (lo más sencillo que les suela funcionar) y ver si tiene confianza alta para etiquetas incorrectas.

(e) Entrenar un modelo usando una configuración, testear su performance en datos que no fueron utilizados para entrenar. Guardar el resultado.

(f) Partir en 10 folds.

(g) Correr un proceso que lista los atributos según qué tanto se correlacionan con las etiquetas y conservar sólo el top10 para entrenar el modelo.

(h) Definir la métrica a utilizar (ej: accuracy pesada por clase)

(i) Mirar / escuchar / leer instancias del dataset para pensar buenos atributos.

(j) No me dio tan bien como esperaba. Empiezo de nuevo repensando algunas cosas.

(k) Armar una grilla de configuraciones a probar

(l) Después de correr lo anterior me di cuenta que podría haber probado tal modelo / tal hiperparámetro.

(m) Agregar a las configuraciones y repetir.

(n) Evaluar el modelo entrenado con los datos completos.

(ñ) Plotear la distribución de las etiquetas.

Ejercicio 4.8. Completar el notebook `notebook_seleccion_modelos.ipynb`

5. Métricas

Ejercicio 5.1. Matriz de Confusión

(a) En un problema de clasificación binaria, ¿a qué se denomina clase positiva y a qué clase negativa? Si nuestro problema consiste en clasificar spam vs. no-spam, ¿cuál es la clase positiva? Si nuestro problema es clasificar imágenes de perros vs. gatos, ¿cuál es la clase positiva?

(b) Explicar con tus palabras la definición de *verdadero positivo*, *verdadero negativo*, *falso positivo* y *falso negativo*.

(c) Completar la Primera Parte del notebook `notebook_metricas.ipynb`. El Test 1 debería pasar.

(d) ¿Por qué podría un falso positivo ser considerado más (o menos) importante que un falso negativo? Dar un ejemplo en donde es más grave tener falsos negativos que falsos positivos.

Ejercicio 5.2. Métricas de umbral fijo

(a) Explicar con tus palabras la definición de *accuracy*, *precision* y *recall*.

(b) Completar la Segunda Parte del notebook `notebook_metricas.ipynb`. El Test 2 debería pasar.

(c) ¿Por qué es un problema medir *accuracy* de un clasificador para compararlo con otro? Dar un ejemplo en donde sería engañoso utilizar esta comparación.

(d) Demuestre que F_β puede ser reescrito como $F_\beta = \dfrac{(1+\beta^2)\cdot \mathrm{TP}}{(1+\beta^2)\cdot \mathrm{TP} + \beta^2 \cdot \mathrm{FN} + \mathrm{FP}}$

(e) Demuestre que la métrica recall vista como función del umbral de decisión, es una función monótona decreciente ($recall_{M,D}(\mu_1) \leq recall_{M,D}(\mu_2)$ si $\mu_1 > \mu_2$)

(f) Muestre cómo la métrica precision vista como función del umbral de decisión, **no necesariamente** es una función monótona creciente ($precision_{M,D}(\mu_1) \not\leq precision_{M,D}(\mu_2)$ si $\mu_1 < \mu_2$)

Ejercicio 5.3. Considerar la Figura Umbral de clasificación. En esta figura se ven instancias ordenadas según la probabilidad detectada por un clasificador (entre 0 y 1). Además, se encuentran marcados cuatro umbrales de decisión.

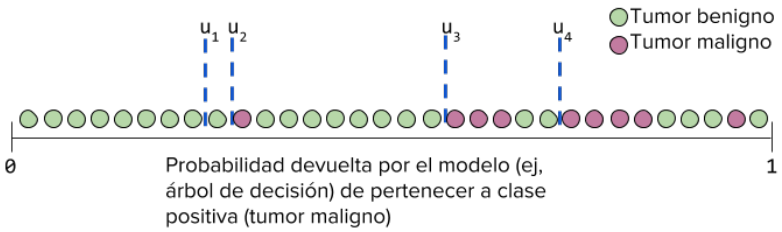


Figura 4: Umbral de clasificación

(a) Calcular las tablas de confusión resultantes para cada uno de los cuatro umbrales de decisión. Recordar que si la probabilidad está por debajo del umbral, la instancia será clasificada como perteneciente a la clase negativa; si está por encima, como clase positiva.

(b) ¿Cuál es el mejor umbral?

(c) Calcular la curva ROC para dicha clasificación.

Ejercicio 5.4.

(a) Escribir el pseudocódigo de la siguiente función:
CURVA-PR(LABELS : LIST⟨BOOL⟩, PROBAS: LIST⟨FLOAT⟩) : LIST⟨TUPLE⟨UMBRAL, VALORPREC, VALORREC⟩⟩
que devuelve los valores de precisión y recall junto a cada umbral explorado.

(b) ¿Qué cambios son necesarios para que esta función devuelva **también** los valores necesarios para construir una curva ROC?

Ejercicio 5.5. Verdadero o Falso (justificar)

(a) El AUC-ROC es invariante de escala. Mide qué tan bien se clasifican las predicciones, en lugar de sus valores absolutos.

(b) En caso que el ordenamiento de las instancias sea el mismo, AUC-ROC va poder distinguir un clasificador muy confiado (instancias clasificadas como positiva tienen scores muy altos, instancias clasificadas como negativas scores muy bajos) de uno poco confiado (scores menos extremos).

(c) AUC-ROC es invariante de umbral de clasificación. Mide la calidad de las predicciones del modelo independientemente del umbral de clasificación elegido.

(d) En los casos en los que existen grandes disparidades en el costo de los falsos negativos frente a los falsos positivos, puede ser fundamental minimizar un tipo de error de clasificación. Por ejemplo, al detectar spam, es probable que desee priorizar la minimización de los falsos positivos (incluso si eso resulta en un aumento significativo de los falsos negativos). AUC-ROC no es la mejor métrica para este tipo de optimización.

Ejercicio 5.6. Sea A un clasificador que tiene un F_1 de 0.80 (con un umbral de clasificación de 0.5), y sea B un clasificador que tiene un F_1 0.70 (también con un umbral de clasificación de 0.5). Sin embargo al cambiar el umbral a 0.4 obtenemos F_1 de 0.76 y 0.80 respectivamente.

(a) Explicar por qué puede suceder este fenómeno dando un ejemplo aproximado.

(b) ¿Podemos concluir algo sobre el AUC Prec-Recall de estos dos modelos?

Ejercicio 5.7. Verdadero o Falso (justificar)

(a) Tanto *recall* como *precision* no toman en cuenta qué tan bien el modelo maneja los casos negativos.

(b) Un modelo que no produce falsos positivos tiene *precision* $= 1.0$.

(c) Un modelo que no produce falsos negativos tiene *recall* $= 1.0$.

(d) Si un clasificador devuelve probabilidades, la matriz de confusión se construye de manera ponderada según la probabilidad de cada clase.

(e) Si un clasificador devuelve probabilidades, hay muchas matrices de confusión asociadas dependiendo del umbral de clasificación.

(f) Si un clasificador devuelve probabilidades, hay infinitas matrices de confusión asociadas dependiendo del umbral de clasificación.

(g) Aumentar el umbral de clasificación produce que la *precision* siempre suba.

(h) Aumentar el umbral de clasificación produce que el *recall* baje o se mantenga igual.

(i) La métrica *precision* es parte fundamental del cálculo de la *curva ROC*.

Ejercicio 5.8. ¿Binaria o 2 clases?
Sean A y B clasificadores que distinguen entre imágenes de perros e imágenes de gatos. Al medir la performance del clasificador (utilizando F_1 para evitar los problemas de utilizar *accuracy*) y "gato" como clase positiva, obtenemos $F_1(A) = 0.9$, $F_1(B) = 0.8$. ¿Podemos concluir que el clasificador A es mejor que el clasificador B para este problema?
Resolver los siguientes ítems para poder responder a la pregunta:

(a) Al calcular F_1 utilizando "gato" como clase positiva, ¿importa qué ocurre con los perros que fueron clasificados correctamente? Revisar la Tercera Parte del notebook `notebook_metricas.ipynb` y decidir cuál clasificador funciona mejor, basándose en las métricas obtenidas. Observar el cambio que ocurre al intercambiar cuál es la clase positiva.

(b) ¿Para qué sirve el parámetro average en la función `f1_score` de la librería sklearn?

(c) ¿Qué sucede si la cantidad de instancias sobre las que fueron testeados es distinta? ¿Cómo se ve afectada la métrica F_1 al cambiar los True Negatives? Correr la Cuarta Parte del notebook `notebook_metricas.ipynb`. El gráfico muestra cómo varía la métrica F_1 al aumentar la cantidad de True Negatives (observar que estamos cambiando la cantidad de instancias sobre las que testeamos). ¿Qué se puede concluir de este experimento?

Ejercicio 5.9. Recordemos la métrica de `accuracy` para un problema de clasificación, para simplificar solo consideraremos el caso binario, sea y el vector de etiquetas verdadera, $\hat{y}$ las respectivas predicciones, y N la cantidad total de muestras ($y, \hat{y} \in \{0,1\}^N$):

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} 1(\hat{y}^{(i)} = y^{(i)})$$

Donde $1(x)$ es la función indicadora (vale 1 si el argumento es verdadero, 0 caso contrario). Vamos a introducir una pequeña modificación en como contamos los aciertos para cada clase, supongamos que p es la proporción de la clase minoritaria (positiva):

$$\texttt{balanced\_accuracy}(y, \hat{y}) = \frac{1}{N} \left(\frac{1}{2p} \sum_{i=1}^{N} 1(\hat{y}^{(i)} = y^{(i)} = 1) + \frac{1}{2(1-p)} \sum_{i=1}^{N} 1(\hat{y}^{(i)} = y^{(i)} = 0) \right)$$

(a) ¿Qué rango de valores posibles tiene esta métrica? ¿Cuánto vale esta métrica si tenemos un clasificador constante que predice siempre la clase mayoritaria $\hat{y} = 0$? ¿Y si predice siempre la clase minoritaria?

(b) Escribir `balanced_accuracy` en términos de TP, FP, TN y FN.

(c) Mostrar que en el caso binario, esta métrica es equivalente al promedio aritmético entre *sensitividad* (true positive rate) y la *especificidad* (true negative rate).

6. Clasificadores

Ejercicio 6.1. Explica con tus palabras qué es el clasificador óptimo de Bayes. ¿Cuál es la razón para no construir siempre este clasificador en la práctica?

Ejercicio 6.2. Dibujar las fronteras de decisión, indicando la clase de predicción de cada región, para un ejemplo de modelo para clasificación binaria que:

I) sobreajusta (overfitting).

II) subajusta (underfitting).

III) es el resultado de aplicar un árbol de decisión de altura máxima 3 (raíz, hijos, nietos).

IV) es el resultado de aplicar K-vecinos más cercanos con K = n.

V) es el resultado de aplicar K-vecinos más cercanos con K = 1.

VI) es el resultado de aplicar LDA (tener en cuenta las probabilidades a priori).

Ejercicio 6.3. Se tienen instancias con sólo dos atributos: altura de una persona (medido en metros) y edad de la persona (medida en años). Se quiere saber si la persona es o no es basquetbolista profesional tomando en cuenta la experiencia de muchas personas.

(a) ¿Es buena idea utilizar el algoritmo de K-vecinos más cercanos con estos datos?

(b) ¿Suponiendo que se utiliza dicho modelo, será útil realizar alguna transformación a los datos previo a ejecutar el algoritmo? ¿Cuál? ¿Por qué?

Ejercicio 6.4. The Prosecutor's Fallacy.
Imaginen que se encuentran en un juicio en donde se debe determinar si una persona (Mónica Gaztambide) se encontraba o no conversando con un atracador en la escena de un crimen. Para ello se cotejan grabaciones de Mónica con grabaciones de la persona que se encuentra conversando con el atracador en el momento del robo.

Un modelo entrenado para reconocer voces de personas sobre una gran cantidad de datos devuelve la siguiente verosimilitud: $P(X = voz_{sospechosa}|Y = monica) = 0.99$. ¿Dirían en base a este resultado que Mónica es culpable?

Ejercicio 6.5. Determinar cuales de las siguientes distribuciones alcanzan por sí solas para decidir la clase de una instancia $x^{(t)}$ siguiendo la receta del clasificador óptimo de bayes (suponer clasificación binaria con clases "0" y "1").

(a) $P(Y = 1|X = x^{(t)})$

(b) $P(X = x^{(t)})$

(c) $P(X = x^{(t)}|Y = 1)$

(d) $P(X = x^{(t)}|Y = 1)$ y $P(X = x^{(t)}|Y = 0)$

(e) $P(X = x^{(t)}|Y = 1)$ y $P(Y = 0)$

(f) $P(Y = 1)$ y $P(Y = 0)$

Ejercicio 6.6.
Consideremos $\delta_c(x)$ la *función discriminante* para la clase c en un problema de clasificación multiclase con k clases y $x \in \mathbb{R}^p$ con $p = 1$:

$$\delta_c(x) = x \cdot \frac{\mu_c}{\sigma^2} - \frac{\mu_c^2}{2\sigma^2} + \log \pi_c \qquad \text{en donde } \pi_c = P(Y = c)$$

Bajo las suposiciones de LDA, se puede derivar

$$Pred(x^{(i)}) = \underset{c \in Clases}{\arg\max}\, P(Y{=}c|X{=}x^{(i)}) = \underset{c \in Clases}{\arg\max}\, \delta_c(x^{(i)})$$

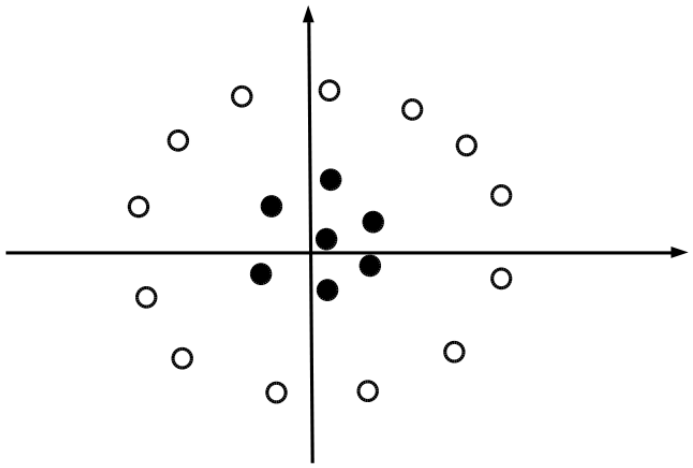
Es decir, para predecir la clase de una instancia, basta con evaluar esta función discriminante para cada clase y conservar la que retorne el mayor valor.

(a) Demostrar esta igualdad para el caso $p = 1$.

(b) Demostrar que la frontera de decisión para el caso con $k = 2$, $\pi_1 = \pi_2$ es $x = (\hat{\mu}_1 + \hat{\mu}_2)/2$.

(c) La palabra *linear* en LDA se debe a que $\delta_k(x)$ es una función lineal en x. Mostrar que si se elimina la suposición de que todas las clases tienen la misma varianza, entonces la función discriminante pasa a ser cuadrática en x. (A esa técnica se la conoce como *quadradic discriminant analysis*, o QDA).

Ejercicio 6.7. En la Figura 7 se muestran 20 puntos bidimensionales. Explicar qué puede hacer SVM con algún kernel para discriminarlos correctamente, y por qué SVM con un kernel lineal fallaría inexorablemente.



Ejercicio 6.8. Describir el sesgo inductivo de Gaussian Naive Bayes (recordar la siguiente fórmula). Pista, hay dos símbolos de approx ($\approx$) en la fórmula. Pista 2: ver los subíndices en la última línea de la fórmula.

$$
\begin{aligned}
Pred(x^{(i)}) &= \arg\max_{c \in Clases} P(Y{=}c | X{=}x^{(i)}) && (1)\\
&= \arg\max_{c \in Clases} \frac{P(Y{=}c) P(X{=}x^{(i)} | Y{=}c)}{P(X{=}x^{(i)})} && (2)\\
&= \arg\max_{c \in Clases} P(Y{=}c) P(X{=}x^{(i)} | Y{=}c) && (3)\\
&= \arg\max_{c \in Clases} P(Y{=}c) P(X_1 = x_1^{(i)} \wedge \cdots \wedge X_p = x_p^{(i)} | Y{=}c) && (4)\\
&\approx \arg\max_{c \in Clases} P(Y{=}c) \prod_{j=1}^{p} P(X_i = x_j^{(i)} | Y{=}c) && (5)\\
&= \arg\max_{c \in Clases} P(Y{=}c) \prod_{j=1}^{p} pdf_c(x_j^{(i)}) && (6)\\
&\approx \arg\max_{c \in Clases} \hat{P}(Y{=}c) \prod_{j=1}^{p} f_{norm}(x_j^{(i)}; \hat{\mu_{c,j}}, \hat{\sigma_j})) && (7)
\end{aligned}
$$

7. Sesgo y Varianza

Ejercicio 7.1. Queremos demostrar el Bias-Variance decomposition para problemas de regresión:

$$
\begin{aligned}
Error\_esperado(x^{(i)}; L) &= \mathrm{E}_{D_n}\left[error\big(y^{(i)}, \hat{h}_{(L,D_n)}(x^{(i)})\big)\right]\\
&\overset{\text{reg}}{=} \Big(\text{Sesgo}\left[\hat{h}_{(L,D_n)}(x^{(i)}); f(x^{(i)})\right]\Big)^2 + \text{Var}\left[\hat{h}_{(L,D_n)}(x^{(i)})\right] + \text{Var}(\varepsilon)
\end{aligned}
$$

I) Según lo visto en clase, ¿cuál es la métrica de error que utilizamos en regresión para la descomposición de sesgo-varianza?

II) Demostrar la descomposición de sesgo y varianza haciendo las suposiciones hechas en clase.

III) Si ahora definimos error como $error(y^{(i)}, pred^{(i)}) = y - pred$, ¿Se obtiene la misma descomposición? (en caso de no poder llegar a una fórmula aclarar dónde encuentran un problema).

Ejercicio 7.2. Verdadero o Falso:

(a) Aumentar la cantidad de datos suele ayudar a contrarrestar problemas de varianza.

(b) Aumentar la cantidad de datos suele ayudar a contrarrestar problemas de sesgo.

(c) Un modelo muy complejo suele producir sesgo alto.

(d) Un modelo muy complejo suele producir varianza alta.

(e) Sesgo alto está asociado a problemas de undefitting.

(f) Varianza alta está asociado a problemas de overfitting.

Ejercicio 7.3. Supongamos que se construyen cuatro clasificadores para discriminar grabaciones de conversaciones en inglés contra grabaciones de conversaciones en español. La siguiente tabla muestra los resultados obtenidos según cuatro algoritmos.

(a) ¿Cuáles de los algoritmos dirían que sufren de alto sesgo?

(b) ¿Cuáles de los algoritmos dirían que sufren de alta varianza?

(c) Imaginen ahora que las grabaciones tienen mucho ruido de fondo y hasta un humano tiene problemas para detectar su origen, ¿dirían que el algoritmo A2 tiene sesgo alto? (suponer que los demás resultados no existen)

Algoritmo:	**A1**	**A2**	**A3**	**A4**
Accuracy (sobre entrenamiento)	.99	.90	.90	.99
Accuracy (sobre validación)	.89	.89	.75	.98

Tabla 2: Sesgo y Varianza

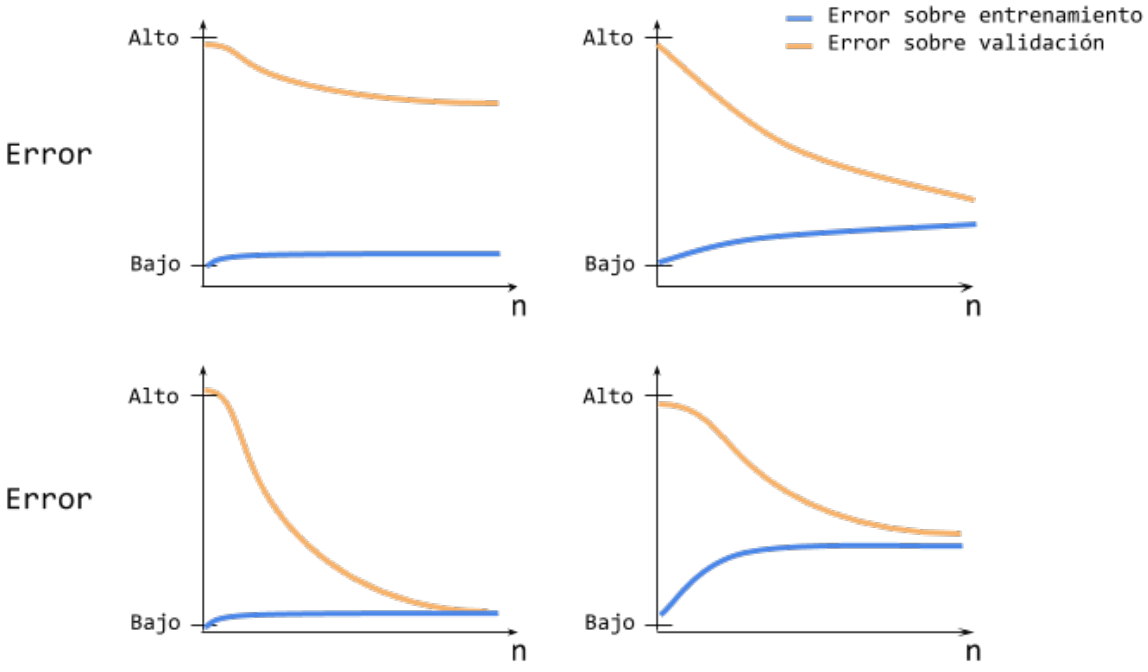
Ejercicio 7.4. Dadas las siguientes curvas de aprendizaje, diagnosticar si se trata de algoritmos con alto o bajo sesgo y alta o baja varianza. En base a su análisis, discutir posibles formas de mejorar la performance de sus modelos. Incluir:

(a) ¿Servirá recolectar más datos de entrenamiento?

(b) ¿Servirá construir un ensamble basado en el algoritmo analizado?

(c) ¿Servirá agregar o quitar features de nuestros datos?

(d) ¿Servirá modificar la altura máxima en árboles o cambiar el parámetro C en SVM?



8. Ensambles

Ejercicio 8.1. Dar una explicación general del algoritmo Bagging.

Ejercicio 8.2. Dar una explicación general del algoritmo Random Forest. Basar la explicación en el algoritmo original presentado en el artículo: Breiman, Leo. "Random Forests." Machine learning 45.1 (2001). Incluir:

- ¿Cuál es su principal diferencia con Bagging?
- ¿Cómo funciona la estimación de error "out-of-bag"? [2]
- ¿Cómo propone Breiman medir la importancia de features? [3] ¿Qué diferencia hay con la manera en que la importancia se mide en el paquete sckit-learn de python? [4]

Ejercicio 8.3. Sea un clasificador binario de tipo Random Forest entrenado sobre un conjunto de datos de entrenamiento. Al evaluar 10 instancias el clasificador devuelve las siguientes probabilidades de pertenencia a la clase positiva: `[0.75, 0.75, 0.25, 0.75, 1.0, 0.0, 0.50, 0.50, 0.75, 1.0]`. Determinar la cantidad de árboles utilizados en el ensamble. Justificar.

Ejercicio 8.4. Utilizar la implementación propia de árboles de decisión para construir modelos Random Forest siguiendo la interfaz de sklearn (es decir, implementar las funciones `fit, predict_proba y predict`)

Ejercicio 8.5. Explicar la idea conceptual del meta-algoritmo AdaBoost. Incluir:

(a) ¿Qué significa "weak-learners"?

(b) ¿Cómo se calculan los pesos para las instancias en cada iteración?

(c) ¿Cuál es el criterio para determinar la cantidad de modelos en el ensamble?

Ejercicio 8.6. Explicar la diferencia entre AdaBoost y GradientBoosting.

Ejercicio 8.7. Verdadero o Falso (justificar)

(a) En Bagging, cada subconjunto tiene la misma cantidad de instancias que el dataset original.

(b) En RF, cada subconjunto tiene la misma cantidad de instancias que el dataset original.

(c) En RF, cada árbol es entrenado sólo con un subconjunto de los atributos.

(d) En Random Forest, tomar m = 1 significa que cada árbol tendría a lo sumo un nivel.

(e) En Random Forest, tomar m = 1 significa que cada árbol tendría a lo sumo un atributo en todo el árbol.

(f) En Random Forest, la importancia de atributos puede medirse como la suma (entre todos los árboles) de la ganancia obtenida en cada corte por cada atributo

(g) En Random Forest, la importancia de atributos puede medirse como la suma (entre todos los árboles) de la ganancia obtenida en cada corte por cada atributo dividido B.

(h) La varianza que se reduce utilizando Bagging debería ser mayor que la que se reduce utilizando Random Forest.

(i) En Bagging, se puede estimar el error de generalización sólo utilizando un train set (sin necesidad de utilizar CrossVal) y los resultados seguramente estén sub-estimando el error real.

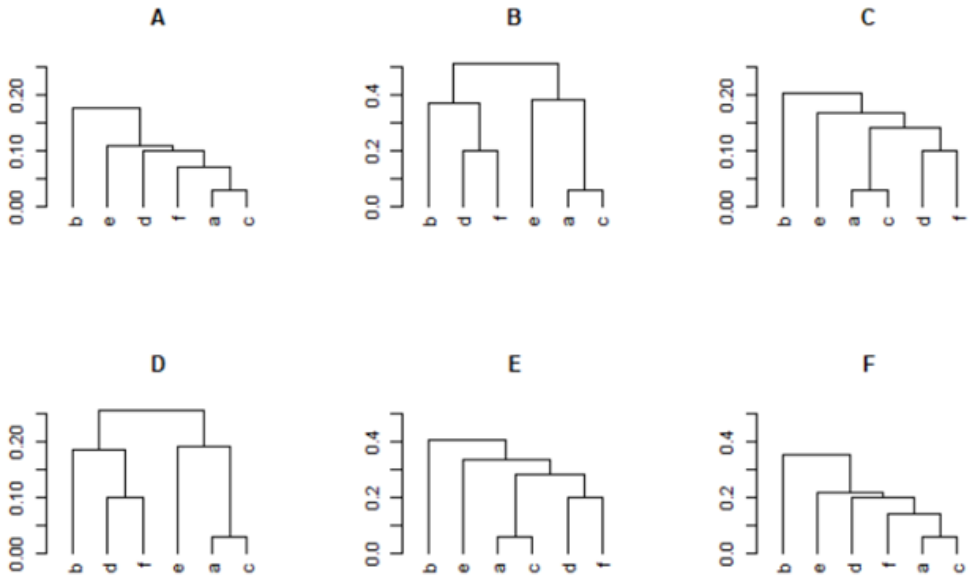
(j) En Bagging, se puede estimar el error de generalización sólo utilizando un train set (sin necesidad de utilizar CrossVal) y los resultados seguramente estén sobre-estimando el error real.

[2] ver Sección 3 del paper.
[3] ver Sección 10 del paper.
[4] ver la sección *1.11.2.5. Feature importance evaluation* en la documentación de scikit-learn sobre ensambles

9. Aprendizaje no supervisado: Clustering

Ejercicio 9.1. ¿Qué tipo de tareas necesitan algoritmos de clustering? Pensar en 3 ejemplos.

Ejercicio 9.2. La figura que se muestra abajo contiene 6 dendrogramas, de los cuales se sabe lo siguiente:



Tres de los seis fueron obtenidos a partir de una matriz de distancias dada a la que llamamos "L", mientras que los restantes tres fueron obtenidos a partir de la matriz de distancias 2 * L (o sea cada elemento de L multiplicado por 2).

Dos de los seis fueron obtenidos utilizando complete linkage como método de aglomeración, dos de los seis fueron obtenidos utilizando single linkage como método de aglomeración y los restantes dos utilizando average linkage como método de aglomeración.

Teniendo esto en cuenta, responda los siguientes puntos:

1. Indique cuáles de los dendrogramas fueron creados tomando como input la matriz de distancias L y cuáles fueron creados tomando como input la matriz de distancias 2 * L. Justifique brevemente su respuesta.
2. Sabiendo que L es la matriz que se muestra debajo y que los dendrogramas obtenidos son los que se mostraron previamente, indique qué método de aglomeración (single, complete o average) se utilizó en cada uno de los seis dendrogramas. Justifique brevemente su respuesta.

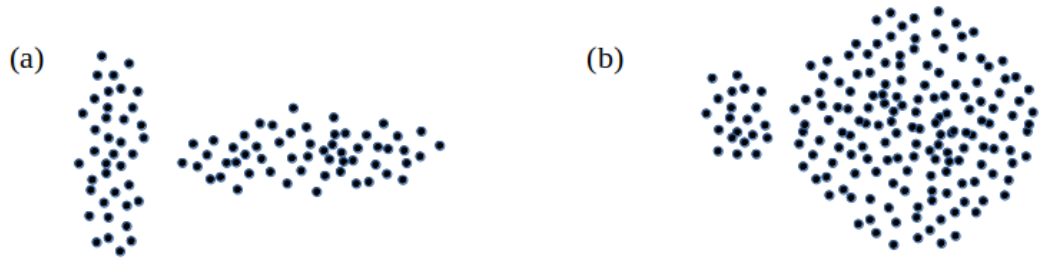
	a	b	c	d	e	f
a	0.000	0.255	0.030	0.120	0.190	0.170
b	0.255	0.000	0.205	0.185	0.195	0.175
c	0.030	0.205	0.000	0.200	0.115	0.070
d	0.120	0.185	0.200	0.000	0.250	0.100
e	0.190	0.195	0.115	0.250	0.000	0.110
f	0.170	0.175	0.070	0.100	0.110	0.000

Ejercicio 9.3. Suponga que para un determinado conjunto de datos se lleva adelante un análisis de clustering jerárquico. Primero calcula la matriz de distancias (en la cual no se observan valores duplicados excepto por los 0 de la diagonal principal). Después, por un lado se obtiene un dendrograma utilizando average linkage y por el otro se obtiene otro dendrograma utilizando single linkage. Responda las siguientes dos preguntas:

a) Suponga que en cierto paso en el dendrograma que utiliza average linkage, un cluster formado únicamente por las observaciones {A, B} y un cluster formado únicamente por las observaciones {C, D, E} se fusionan. A su vez, en el dendrograma que utiliza single linkage, también ocurre que en cierto paso un cluster formado únicamente por las mismas observaciones {A, B} y otro formado únicamente por las mismas observaciones {C, D, E} se fusionan. ¿Cuál de las dos fusiones ocurrirá más arriba en el dendrograma? ¿O ocurrirán a la misma altura? ¿O no se dispone de información suficiente para responder? Justifique su respuesta

b) Suponga que en cierto punto en el dendrograma que utiliza average linkage, un cluster formado únicamente por la observación {L} y un cluster formado únicamente por la observación {M} se fusionan a una altura X. Suponga también que en el dendrograma que utiliza single linkage ocurre que en cierto punto un cluster formado únicamente por la observación {R} y un cluster formado por las observaciones {S, T} se fusionan a una altura Y. A su vez, asuma que X es menor a Y ($X < Y$). Si llamamos $d(i, j)$ a la distancia entre la observación i y la observación j, ¿es cierto o falso que $d(R, S) < d(L, M)$? Justifique su respuesta. Si considera que no se dispone de la información suficiente para responder esta pregunta indique qué información adicional se requiere para responderla.

Ejercicio 9.4. ¿Qué resultará de ejecutar K-Means y GMM (en ambos casos con K=2) para cada uno de los siguientes datasets? Justificar.



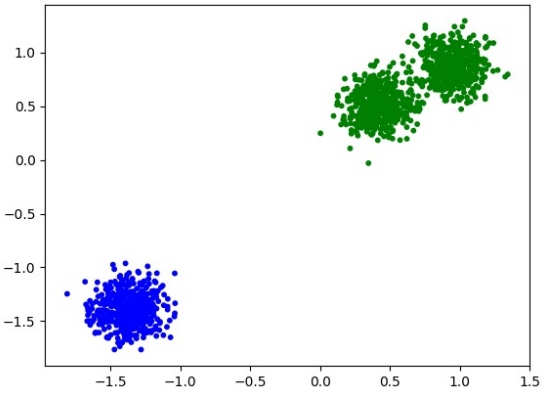
Ejercicio 9.5. Dada la densidad de un modelo de mezcla de gaussianas p-dimensional (i.e. $x \in \mathbb{R}^p$):

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(x)$$

donde $g_k = \mathcal{N}(\mu_k, \mathbf{I} \cdot \sigma^2)$ y $\pi_k \geq 0$ con $\sum_{k=1}^{K} \pi_k = 1$. Aquí $\{\mu_k, \pi_k\}$, $k = 1, \ldots, K$ y σ^2 son parámetros desconocidos. Supongamos que tenemos datos $x_1, x_2, \ldots, x_N \sim g(x)$ y queremos ajustar el modelo de mezcla.

1. Escribir la función de log-verosimilitud de los datos.
2. Derivar un algoritmo EM para calcular las estimaciones de máxima verosimilitud.
3. Mostrar que si σ tiene un valor conocido y tomamos $\sigma \to 0$, entonces en un sentido este algoritmo EM coincide con K-means.

Ejercicio 9.6. En el `notebook_7_clustering` se construye un dataset de ejemplo, ejecuta K-Means y grafica los resultados:



(a) Experimentar con diferentes valores para `cluster_std` (desvío estándar de las nubes de puntos) y `n_clusters` (valor de K en K-Means).

(b) Generar datasets con otras formas, como por ejemplo:[5]

```
datasets.make_circles(n_samples=N, factor=.5, noise=.05)
datasets.make_moons(n_samples=N, noise=.05)
```

[5]Ver más ejemplos en `http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`

Ejecutar K-means, y también DBSCAN con el comando `cluster.DBSCAN(eps=.2)`.
Experimentar con diferentes valores para `noise` y `eps`.

Ejercicio 9.7. Completar las implementaciones del `notebook_7_clustering` de:

(a) K-Means.

(b) Clustering Jerárquico Aglomerativo.

Comparar cómo funcionan sus implementaciones sobre los datos del ejercicio anterior.

Ejercicio 9.8. Completar la implementación de Expectation Maximization para GMMs de dimensión 1 del notebook `notebook_7_clustering`. Guiar el algoritmo en las fórmulas detalladas en el notebook.

(a) Completar la función `inicializacion`. Para este punto recomendamos utilizar K-Means (de sklearn o propio) como algoritmo para encontrar las medias iniciales.

(b) Completar la función `e`.

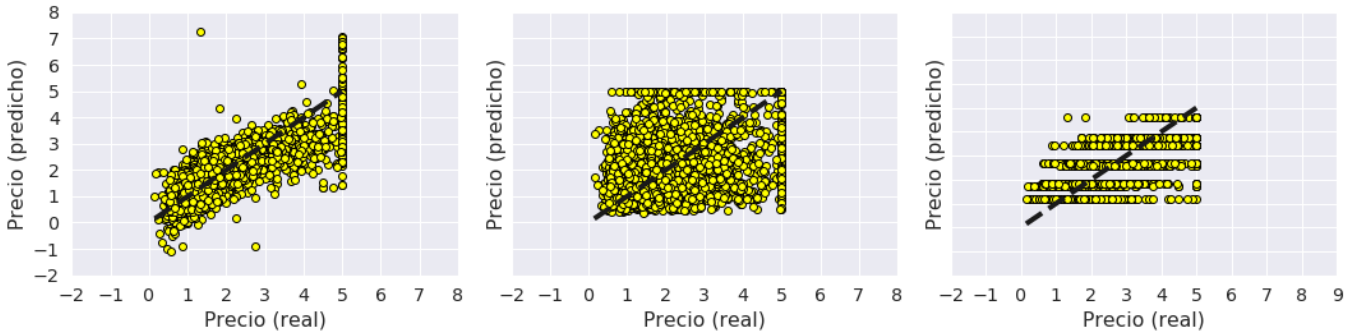
(c) Completar la función `m`.

10. Regresión y Regularización

Ejercicio 10.1. Verdadero o Falso

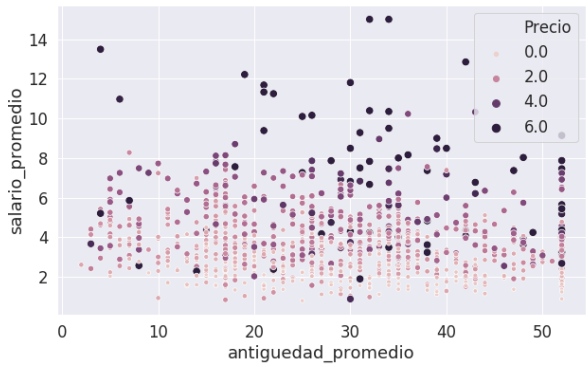
1. El algoritmo de árboles de decisión no puede ser adaptado para un problema de regresión ya que la selección del mejor corte $< atributo, corte >$ es imposible cuando la variable a predecir es continua.
2. KNN para regresión con K = n (tamaño del conjunto de datos) devuelve una única región de decisión, para cada nueva instancia siempre se devolverá el mismo número y este número será el promedio de los datos de entrenamiento.
3. En árboles de decisión para regresión, los dos cambios principales son (a) utilizar, por ejemplo, la Varianza de lar región como medida de impureza de una región, (b) calcular el valor de una hoja como el promedio de las instancias que caen en esa región.
4. En árboles de decisión para regresión, si utilizamos MSE como medida de impureza, tomando como valor "real" para la región al promedio de las instancias, es equivalente a reducir la varianza.

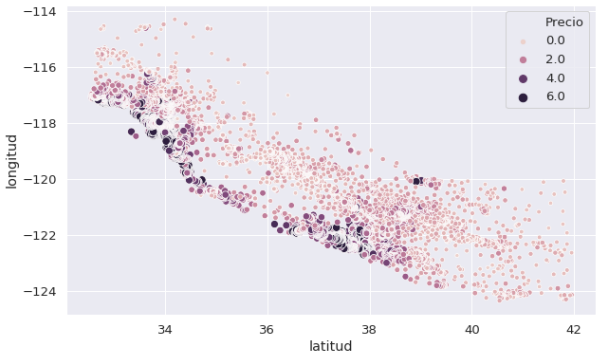
Ejercicio 10.2. En la siguiente imagen puede verse el resultado de tres tipos de modelos distintos entrenados sobre los datos del dataset *California Housing Dataset*. Cada gráfico muestra el valor real y el valor predicho para instancias no vistas en entrenamiento (conjunto de validación).



(a) ¿Qué se esperaría de estos gráficos si la regresión fuese perfecta?

(b) Determinar qué figura corresponde a una Regresión Lineal, cuál a un Decision Tree Regressor (profundidad = 4) y cuál a un KNN Regressor (K = 1). Justifique.





Ejercicio 10.3. El dataset *California Housing Dataset* [6] contiene una serie de instancias que representan zonas de California en las que interesa conocer el valor promedio de las viviendas. Cada instancia está representada por atributos tales como latitud, longitud, antigüedad promedio de las viviendas, y salario promedio de las personas en la zona, entre otros. A continuación se muestran dos gráficos generados a partir de estos datos:
Imagine que se ajusta una regresión lineal a dichas instancias. Dibuje una representación de cómo esperaría que dicha regresión impacte sobre cada uno de estos gráficos.

Ejercicio 10.4. Contestar las siguientes preguntas en el contexto de regresión, justificar:

1. ¿Cómo afecta la normalización de los datos para el modelo de regresión lineal sin regularización?
2. ¿Qué sucede al momento de aplicar regularización Ridge por ejemplo?
3. ¿Afecta al error MSE aplicar o no aplicar normalización a los atributos para el cálculo del error?
4. ¿Afecta la escala de Y (si cambiamos la salida y las etiquetas de km. a mts. por ejemplo) en las decisiones que toma el modelo si utilizamos MSE como métrica a optimizar?

Ejercicio 10.5. Gradient Descent

1. Escribir el pseudocódigo de la función "Descenso por gradiente", comentando brevemente qué se espera de cada argumento (junto a su tipo).
2. Comentar cuál es la función a minimizar en el caso de una regresión lineal.
3. Explicar cómo se obtiene el gradiente de la función a minimizar en el caso de una regresión lineal.
4. Escribir el pseudocódigo de *mini-batch gradient descent*.

Ejercicio 10.6. Pensando al error cuadrático medio (MSE) como una función de pérdida:

$$\text{MSE}_{\text{X,y}} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{h}(x^{(i)}) - y^{(i)}\right)^2$$

Verdadero o Falso:

1. Esta métrica se define de esta manera para regresión lineal, para otros métodos (tal como árboles de regresión) hay que redefinir la fórmula anterior.
2. La función, vista como una función de pérdida, es siempre convexa.
3. Considerar las siguientes afirmaciones en el contexto de regresión lineal con pesos w:
   - I) Se puede usar como función de pérdida viéndola como una función de X e y (los datos).
   - II) Se puede usar como función de pérdida viéndola como una función de w.
   - III) La función, vista como una función de pérdida, es siempre convexa.
   - IV) Se busca minimizar esta función para X e y fijos en cada iteración de descenso de gradiente.
   - V) En mini-batch gradient descent, en cada batch se minimiza una función de pérdida distinta.

[6] `http://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html`

Ejercicio 10.7. Verdadero o Falso

1. Entrenar una regresión lineal significa minimizar los pesos w.
2. Entrenar un modelo significa minimizar una función de costo y esta función depende de los pesos w.
3. Regularizar significa minimizar el MSE(w) manteniéndolo cerca de 0.
4. Regularizar significa minimizar el MSE(w) sumado a algún término de costo c(w) que habla de la magnitud de estos pesos.
5. Para un problema como el de predicción de casas, el w asociado a una variable como "distancia al obelisco" no depende de la unidad. Es decir, obtendremos el mismo w si medimos la distancia en km. o en mts.
6. Existen otras funciones de costo que permiten ajustar los w, por ejemplo, el error absoluto promedio (MAE).
7. La métrica MSE(w) penaliza w grandes.

Ejercicio 10.8. Resolver el notebook `notebook_descenso_gradiente.ipynb`.

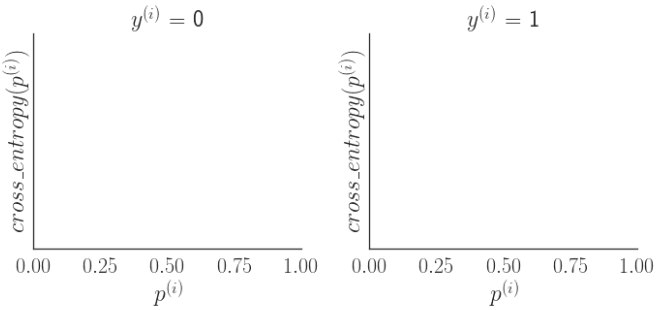
11. Regresión Logística (Clasificación)

Ejercicio 11.1. Para clasificación a través de la técnica de regresión logística, suele utilizarse la función "Binary Cross-Entropy" para definir el error de una predicción en particular. Este error se puede escribir como:

$$\text{Binary\_CE}(\mathbf{y}^{(i)}, \mathbf{p}^{(i)}) = \begin{cases} -\log(\mathbf{p}^{(i)}) & \text{si } \mathbf{y}^{(i)} = 1 \\ -\log(1 - \mathbf{p}^{(i)}) & \text{si } \mathbf{y}^{(i)} = 0 \end{cases}$$

(a) ¿A qué hacen referencias las variables $\mathbf{y}^{(i)}$ y $\mathbf{p}^{(i)}$ en este cálculo?

(b) Completar los gráficos dibujando el costo asociado al error según la clase original de la instancia:



(c) Explicar con sus palabras por qué es bueno que este costo esté cerca de cero y en qué caso se acerca a infinito.

(d) Expresar la fórmula completa del costo asociado a un conjunto de datos $\mathbf{X}$ y sus etiquetas $\mathbf{y}$.

Ejercicio 11.2. ¿Cuál es el valor esperado de predicción de un modelo de regresión logística que utiliza regularización L2 (Ridge) con λ tendiendo a infinito?

Ejercicio 11.3.

1. Escribir el pseudocódigo de la función "Descenso por gradiente" para el caso de regresión logística, comentando brevemente qué se espera de cada argumento (junto a su tipo).
2. Explicar cómo se obtiene el gradiente de la función a minimizar.
3. Escribir el pseudocódigo de mini-batch gradient descent.

Ejercicio 11.4. Los gráficos de la Figura 4 han sido generados mediante la función sigmoidea: $sigm(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$. Para cada uno, $z^{(i)}$ fue calculado utilizando distintos w_0 y w_1 siguiendo la fórmula $z^{(i)} = w_0 + w_1 * x_1^{(i)}$.

1. Determinar en qué dibujos w_1 es positivo y en cuáles negativo. ¿En qué afecta el signo de los distintos w_s en el caso de instancias multidimensionales?
2. Por fila, ordenar los dibujos según su valor de w_0.

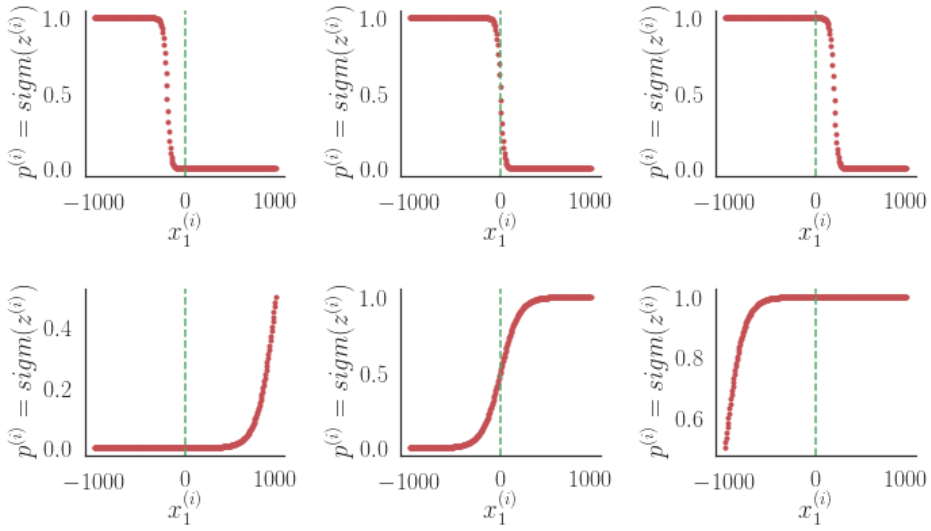


Figura 5: Sigmoideas

12. Redes Neuronales (FNNs)

Ejercicio 12.1. Verdadero o Falso

1. Un perceptrón simple con función de activación sigmoidea es equivalente a un modelo de regresión logística.
2. Un perceptrón simple con función de activación lineal es equivalente a una regresión lineal.
3. En un problema de regresión de $\mathbb{R}^p \to \mathbb{R}^q$, una red sin capas ocultas con q neuronas de salida aprendería los mismos pesos que entrenar q regresiones lineales simples. Ejemplo, predecir no sólo el valor de una casa sino también sus metros cuadrados según p atributos.
4. En un problema de regresión de $\mathbb{R}^p \to \mathbb{R}^q$, una red con capas ocultas con q neuronas de salida aprendería los mismos pesos que entrenar q regresiones lineales (con la misma arquitectura pero sólo 1 neurona de salida).

Ejercicio 12.2. Demostrar que una red neuronal con una neurona de salida, con función de activación lineal en todas las capas salvo la última, y activación sigmoidea en la última capa es equivalente a una regresión logística. ¿Tiene sentido utilizar una red con muchas capas en este caso?

Ejercicio 12.3. Dada una red neuronal con la siguiente arquitectura, en donde X hace referencia al tamaño de la entrada; $W^{[l]}$ a la matriz de pesos que conecta la capa l con la capa $l-1$; $A^{[l]}$ a las activaciones de la capa l

$$X \in \mathbb{R}^{5\times 3}, \quad W^{[1]} \in \mathbb{R}^{*\times 2}, \quad A^{[1]} \in \mathbb{R}^{*\times *}, \quad W^{[2]} \in \mathbb{R}^{*\times 1}, \quad A^{[2]} \in \mathbb{R}^{*\times *}, \quad W^{[3]} \in \mathbb{R}^{*\times 2}, \quad Y \in \mathbb{R}^{*\times *}$$

1. Escribir la fórmula denotada por esta red. Es decir, $Y = \dots$ suponiendo funciones de activación g_i para toda capa intermedia, y g_o para la salida. Escribir dos versiones, una en la que los términos de bias están explícitos, una en la que no. Para lo segundo, utilizar la notación $ext(M)$ que simboliza agregar una columna de unos en primer lugar en la matriz M.
2. Completar los valores faltantes denotado con asteriscos (siempre refiriéndose a las versiones no extendidas).
3. Dibujar el esquema de la red neuronal.

Ejercicio 12.4. Construir a mano un perceptrón simple que resuelva el operador lógico AND: dadas dos variables X_1 y X_2, devuelve *True* o *False*. En las variables de entrada, interpretar $X_i = 1$ como *True* y $X_i = 0$ como *False*. Ídem para los operadores OR, NOR y NAND.

Ejercicio 12.5. Cantidad de parámetros de una red neuronal

1. Sea una red neuronal densa con *biases* con una capa de entrada con 3 neuronas, una capa oculta con 4 neuronas y una capa de salida con 2 neuronas. Realizar un diagrama de dicha red y calcular la cantidad de pesos en esta red neuronal.

2. Sea una red neuronal con densa M capas ocultas, donde la i-ésima capa oculta tiene N_i neuronas ($i = 1, 2, \ldots, M$), una capa de entrada con I neuronas y una capa de salida con O neuronas. Calcular la cantidad total de pesos en esta red neuronal.

3. Suponga que cada capa oculta tiene una cantidad fija de N_h neuronas para todas las capas. Determinar cómo crece la cantidad total de pesos en la red en términos de M, es decir, determinar la '*complejidad del modelo*' en términos de la cantidad de capas.

Ejercicio 12.6. En caso de estar resolviendo un problema de regresión:

- ¿Cuándo tiene sentido utilizar una función de activación lineal en la última capa?
- ¿Cuándo tiene sentido utilizar una función de activación ReLu en la última capa?

Ejercicio 12.7. Backpropagation
En este ejercicio, consideraremos una red neuronal con las siguientes definiciones:

- La entrada z de una neurona j en la capa l está dada por:

$$z_j^{[l]} = \sum_i w_{i,j}^{[l]} a_i^{[l-1]} + b_j^{[l]}$$

  donde $a_i^{[l]} = \sigma(z_i^{[l]})$, y σ es la función de activación. Aquí, i representa los índices de las neuronas de la capa anterior. Además $w_{i,j}^{[l]}$ representa el peso desde la neurona i de la capa $l-1$ a la neurona j de la capa l.

- La función de costo C para la red neuronal está definida como:

$$C = \sum_i \frac{1}{2}(y_i^{[L]} - a_i^{[L]})^2$$

  donde L es la última capa de la red neuronal.

- Definimos el siguiente término para la última capa:

$$\frac{\partial C}{\partial z_j^{[L]}} = \delta_j^{[L]}$$

- Para la última capa, se tiene que:

$$\frac{\partial C}{\partial z_j^{[L]}} = \delta_j^{[L]}$$

Utilizando estas definiciones, resuelve los siguientes problemas:

1. Demuestra que:

$$\delta_j^{[L]} = (a_j^{[L]} - y_j^{[L]})\sigma'(z_j^{[L]})$$

2. Demuestra que:

$$\delta_j^{[l]} = \sigma'(z_j^{[l]}) \sum_k \delta_k^{[l+1]} w_{jk}^{[l+1]}$$

   Sugerencia: usa la siguiente igualdad:

$$\delta_j^{[l]} = \sum_k \frac{\partial C}{\partial z_k^{[l+1]}} \frac{\partial z_k^{[l+1]}}{\partial z_j^{[l]}}$$

   donde k es el número de neuronas de la capa $l+1$. Backpropagation

3. Demuestra que:

$$\frac{\partial C}{\partial w_{ij}} = \delta_j^{[l]} a_i^{[l-1]}$$

4. Demuestra que:

$$\frac{\partial C}{\partial b_j^{[l]}} = \delta_j^{[l]}$$

5. Conceptualmente, ¿qué representan los δ?

Para resolver los siguientes ejercicios, usar el playground de TensorFlow disponible en `http://playground.tensorflow.org`.

Ejercicio 12.8. Elegir el tercer dataset en el playground, que tiene dos grupos de puntos bien separados. Experimentar con diferentes configuraciones de capas ocultas, nodos y atributos, y estudiar el comportamiento de cada parte de la red durante el entrenamiento. Por ejemplo, usar:

- un solo atributo (X_1) y una capa con un nodo;
- un solo atributo (X_1) y dos o más capas con dos o más nodos;
- dos atributos (X_1, X_2) y una sola capa con un nodo; etc.

Ejercicio 12.9. Usando sólo los atributos X_1 y X_2, construir una configuración mínima (en cantidad de capas y de nodos por capa) de un perceptrón multicapa que resuelva el operador lógico XOR. Usar el segundo dataset del playground.

Ejercicio 12.10. Usando sólo los atributos X_1 y X_2, construir una perceptrón multicapa que pueda aprender los otros dos problemas no linealmente separables incluidos en el playground: el círculo azul rodeado de amarillo (fácil) y la doble espiral (difícil).

Ejercicio 12.11. Estudiar cómo impacta en el aprendizaje de los dos ejercicios anteriores la inclusión de otros atributos (por ejemplo, $sin(X_1)$), así como la elección de distintas funciones de activación (lineal, tanh, sigmoid).

13. Predicción de secuencias

Ejercicio 13.1. Se te ha dado un conjunto de datos de mensajes etiquetados con sentimientos positivos y negativos. Suponer que las palabras en los mensajes están representadas mediante embeddings de dimensión 100.

1. Dibujar el esquema de la arquitectura de una Red Neuronal Recurrente (RNN) con **dos capas ocultas** que permita resolver este problema. Dibujar dos versiones: una compacta y una desplegada en el tiempo. Indicar la dimensión de la entrada, los estado ocultos y la salida.
2. ¿Qué función de pérdida utilizaría?
3. ¿Sobre qué instantes de tiempo aplicaría la función de pérdida? En caso que haya instantes de tiempo en el cuál no calcula la función de pérdida, ¿qué haría en esos instantes para no tener que cambiar los algoritmos de minimización?

Ejercicio 13.2.
Dada una RNN con los siguientes parámetros:

- Entrada $\mathbf{x}_{\langle t \rangle}^{(k)} \in \mathbb{R}^p$
- Salida $\mathbf{o}_{\langle t \rangle}^{(k)} \in \mathbb{R}^k$
- Estado oculto $\mathbf{h}_{\langle t \rangle}^{(k)} \in \mathbb{R}^q$
- Matrices de pesos $W_{xh} \in \mathbb{R}^{q \times p}$, $W_{hh} \in \mathbb{R}^{q \times q}$, y $W_{hy} \in \mathbb{R}^{k \times q}$
- Vectores de sesgo $b_h \in \mathbb{R}^q$ y $b_y \in \mathbb{R}^k$
- Función de activación: tangente hiperbólica: $\tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$

Se pide:

1. Escribe las ecuaciones para el estado oculto $\mathbf{h}_{\langle t \rangle}^{(k)}$ y la salida $\mathbf{o}_{\langle t \rangle}^{(k)}$ en el paso de tiempo t.
2. Supongamos que tenemos los siguientes valores:
   $p = 3, q = 2, k = 1$

   $$x = \left[\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 4 \\ 2 \end{pmatrix} \right], \quad h_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

   $$W_{xh} = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix}, \quad W_{hh} = \begin{pmatrix} 0.7 & 0.8 \\ 0.9 & 1.0 \end{pmatrix}, \quad W_{hy} = \begin{pmatrix} 1.1 & 1.2 \end{pmatrix}, \quad b_h = \begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}, \quad b_y = \begin{pmatrix} 0.3 \end{pmatrix}$$

3. Calcular los valores resultantes del estado oculto y las salidas en cada instante de tiempo. Tip $\mathbf{h}_{\langle 1 \rangle}^{(k)} = \begin{pmatrix} 0.53 \\ 0.92 \end{pmatrix}$
4. Programar la cuenta anterior utilizando el siguiente código base:

```
import numpy as np

# Definir la función de activación tanh
def tanh(z):
    return np.tanh(z)

# Valores iniciales y parámetros
x = [np.array([1, 2, 0]), np.array([0, 4, 2])]
h_0 = np.array([0, 0])

W_xh = np.array([[0.1, 0.2, 0.3], [0.4, 0.5, 0.6]])
W_hh = np.array([[0.7, 0.8], [0.9, 1.0]])
W_hy = np.array([[1.1, 1.2]])

b_h = np.array([0.1, 0.2])
b_y = np.array([0.3])
```

Ejercicio 13.3. Se extiende a las RNN como muestra la imagen. Dar las fórmulas asociadas a la nueva arquitectura (en donde la línea punteada corresponde a conexiones recurrentes).

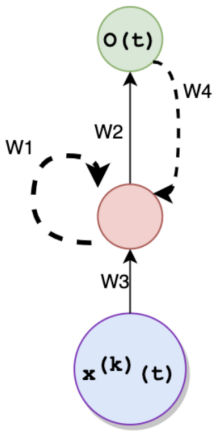


Figura 6: Diagrama de RNN

Ejercicio 13.4. Diseñar la arquitectura de una red neuronal de tipo Encoder - Decoder que permita resumir texto. Contestar luego las siguientes preguntas:

1. ¿Qué tipo de red utilizó en el Encoder?
2. ¿Qué tipo en el Decoder?
3. ¿Cómo conecta el Encoder con el Decoder?
4. ¿Utilizó algún mecanismo de atención?
5. ¿Utilizó redes que leen la entrada de manera secuencial?
6. ¿Utilizó redes que leen la entrada de manera bidireccional?
7. ¿Utilizó redes que leen la entrada todo al mismo tiempo?
8. ¿Hay algún límite en el tamaño de la entrada?

14. Ingeniería de Atributos

Ejercicio 14.1. Se tienen datos de un famoso sitio de compra/venta de inmuebles, y se nos pide estimar estimar el valor de una propiedad en base a sus atributos continuos. Al normalizar los atributos utilizando la técnica de min-max notamos que la mayoría de los valores quedaron en cero (o muy cerca de cero) y sólo un valor quedo en 1.

(a) Explicar qué puede haber ocurrido.

(b) Explicar qué técnica alternativa utilizaría si se pensara que el atributo sigue una distribución normal.

Ejercicio 14.2. En un problema de clasificación, decidimos imputar los valores faltante de un atributo utilizando la media del resto de los valores. Al comparar la performance de nuestro modelo entrenado contra la realidad (al llevarlo a producción) vemos que hubo una subestimación del error.

(a) Explicar qué puede haber ocurrido.

(b) Indique claramente en qué momento del proceso calcularía el valor a imputar (pensar en la partición desarrollo-control, y en validación cruzada con K-folds). Especificar qué valores se utilizarían para el cálculo.

(c) ¿Con qué valor deberíamos imputar los valores que sigamos obteniendo en producción?

Precio	Ubicación	Metros cuadrados	Habitaciones	Fecha de publicación	Baños
100000	Nuñez	80	2	20/10/2021	2
150000	Almagro	100	3	02/05/2021	N/A
200000	Caballito	N/A	2	N/A	1
N/A	N/A	120	N/A	03/09/2021	2

Tabla 3: Datos de propiedades

Ejercicio 14.3. Supongamos que tenemos un conjunto de datos de 2021 que contiene información sobre el precio y la ubicación de diferentes propiedades inmobiliarias en Capital Federal. Estamos interesados en predecir si la propiedad se vendió o no durante ese mismo año.

Dada la siguiente porción de la base de datos, Proponer 3 nuevas columnas que puedan aportar nueva información y ser útiles para el problema en cuestión que no se puedan deducir de las columnas actuales. Describa todas las transformaciones que te parecen necesarias para poder utilizar esta tabla en un algoritmo de KNN implementado en sklearn. Para cada transformación, ser concreto en cómo realizan esa transformación y de algún ejemplo.

Ejercicio 14.4. Dados los siguientes problemas de clasificación, indique qué transformaciones aplicaría a los datos antes de utilizar el algoritmo de K-vecinos más cercanos. En caso de no tener que hacer transformaciones justificar por qué. Todos los atributos agregados deberán ser transformaciones de atributos ya existentes entre los datos.

(a) Se quiere determinar si una persona utilizará una bicicleta de la ciudad para volver de su trabajo en base a: momento en el que salió del trabajo (timestamp que contiene fecha y hora), latitud y longitud en donde se encuentra su trabajo.

(b) Se quiere determinar si una persona visitará o no el barrio chino el día de hoy. Para eso, tenemos el día (fecha completa), la temperatura (medida en grados) y la longitud y latitud en la que se encuentra. Suponer que hay una gran cantidad de gente que visita el barrio en el año nuevo.

(c) Se quiere predecir la temperatura en función de la distancia en kilómetros a la montaña más cercana, la altitud y la cantidad de picos montañosos a menos de 10 kilómetros de la ciudad.

(d) Se quiere determinar la raza de un perro en base a: color de ojos (azul, verde, gris, etc.), color del pelo predominante (marrón, verde, gris, etc.), y tamaño del animal (chico, mediano, grande)

(e) Se quiere determinar si una persona nació o no en una región en base a su nombre y apellido.

¿Cómo modificaría la respuesta para el punto (c) en caso de utilizar árboles de decisión en vez de KNN?

Ejercicio 14.5. Tenemos un problema de clasificación con instancias con p atributos. ¿Cuántos posibles subconjuntos de atributos habría que explorar, si quisiéramos hacer una búsqueda completa? ¿Qué complejidad temporal tendría? ¿Qué complejidad temporal tienen las técnicas de *Forward selection* y *Backward elimination*? ¿Cuál es la complejidad de utilizar RFE (recursive feature elimination)?