

FACULTAD DE INGENIERÍA

UBA

# TALLER DE PROCESAMIENTO DE SEÑALES

(TA136)


Guía de Trabajos Prácticos

Versión 1.1


Segundo cuatrimestre 2024

## Guía 1

---

**1.1**  Sin utilizar loops (for/while) convertir a escala de grises la imagen `pikachu_vs_charmander.jpeg` implementando las siguientes técnicas:

- (a)  $\frac{\min(R,G,B) + \max(R,G,B)}{2}$
- (b)  $\frac{R+G+B}{3}$
- (c)  $0.3R + 0.59G + 0.11B$  (utilice el comando @)

: Utilice las funciones `imread` e `imshow` (matplotlib).

---

**1.2**  Sin utilizar loops (for/while), utilizando indexación edite la imagen `AFALogo.bmp` para

- (a) Cortar las letras dentro del logo.
  - (b) Cortar las estrellas y transponerlas.
  - (c) Generar una mascara separando el color de fondo del logotipo.
  - (d) Cambiar el color de fondo de negro a blanco.
  - (e) Espejar la imagen (izquierda a derecha).
  - (f) Dibujar una grilla sobre la imagen cada 4 píxeles.
  - (g) Agregar la 3era estrella.
- 

**1.3** Sea la función de densidad de probabilidad


$$p_{XY}(x, y) = \frac{4}{5} \mathbb{1} \{0 < y < 1 + x^3, 0 < x < 1\}$$

(a) Calcular y graficar en una misma figura el soporte, la esperanza condicional  $\mathbb{E}[Y|X = x]$  y la recta de regresión.

(b) Calcular el error bayesiano.

: Las únicas integrales que debe resolver son con respecto a la marginal  $p_X(x)$ . El resto de los cálculos debe hacerse utilizando propiedades.

---

**1.4**  Una conocida cadena de comida rápida desea predecir la ganancia de una sucursal en función de la cantidad de habitantes de la ciudad para decidir si conviene abrirla o no. El archivo `mc.txt` contiene la base de datos a utilizar. La primera columna es la población de la ciudad (de a 10.000 personas) y la segunda es la ganancia (de a \$USD 10.000). Los valores negativos indican pérdidas.

(a) Implemente su propio código, utilizando matrices, para realizar una regresión lineal que minimice el error cuadrático medio. ¿Cuanto vale dicho error?


(b) Visualizar los datos con `scatter` (matplotlib) y superponer la recta de regresión estimada sobre ellos.

(c) Diseñe una grilla de puntos que permita graficar el riesgo empírico (en un gráfico 3d) en función de los parámetros (w y b), utilizando `plot_surface` (matplotlib).


Compruebe que los parámetros encontrados en el inciso (a) son efectivamente los que minimizan el riesgo.

(d) Predecir la ganancia de una ciudad de 35.000 habitantes.

(e) Repetir la regresión lineal utilizando `LinearRegression` (sklearn). Comparar resultados.


**1.5**  Se desea analizar los vientos que ocurren en un parque eólico. El archivo `molinos.csv` contiene datos de potencias acumuladas por un parque eólico para los diferentes vientos. La columna `Velocity` contiene el módulo de la velocidad del viento en ese instante y la columna `Direction` el ángulo de la velocidad medido en sentido horario ubicando el cero en vientos que provienen del norte. Finalmente las columnas `P` contiene las potencias acumuladas por cada molino.

(a) Las potencias negativas son errores de medición. Reemplazar todos estos valores con el valor medio de los valores restantes.


: El comando `SimpleImputer` (sklearn) puede ser útil.

(b) Expresar la velocidad en coordenadas cartesianas.

(c) Entrenar un regresor lineal que estime la velocidad del viento (dos dimensiones cartesianas) en función de las potencias.


: El comando `MultiOutputRegressor` (sklearn) puede ser de gran utilidad.

(d) Utilizar `mean_squared_error` (sklearn) para calcular el ECM de entrenamiento. Si cada coordenada de la velocidad tiene asociado un error cuadrático medio, ¿Por qué sklearn devuelve un solo valor? ¿Qué cálculo está haciendo?

**1.6**  Una inmobiliaria desea automatizar la tarea de tasar terrenos. El archivo `inmobiliaria.csv` contiene la base de datos de casas en California.


(a) Explorar los datos usando `read_csv` (pandas). Indicar cantidad de muestras, nombre y tipo de dato de cada *feature*.

(b) Indicar la proporción de las variables categóricas representando probabilidades.

: Si no sabe que tipo de variable aleatoria es la categórica, deberá buscar dicha información.

(c) Para analizar las variables numéricas utilice el comando `pairplot` (seaborn). Explique que representan los gráficos.

(d) Utilice el comando `SimpleImputer` (sklearn) para completar los valores faltantes con los más frecuentes.

(e) Utilice el comando `get_dummies` (pandas) para codifique las variables categóricas como *one-hot*. : En caso de no conocer el concepto, buscar información sobre *one-hot encoding*.

(f) Utilice el comando `train_test_split` (sklearn) para definir dos conjuntos con las proporciones 75 % y 25 %. Grafique los histogramas de ambos conjuntos (superpuestos) de la mediana del valor de las propiedades.


(g) Utilice el comando `StandardScaler` (sklearn) para normalizar cada variable exceptuando a la mediana del valor de la propiedad. Utilice el conjunto de entrenamiento para fijar la normalización y aplíquela a ambos conjuntos.

(h) Realizar una regresión lineal para predecir la mediana del valor de la propiedad en función del resto de las variables. Indicar el ECM de entrenamiento y testeo.

---

**1.7** Hallar una solución matricial al problema de regresión lineal sin sesgo y con regularización L2. ¿A que se aproxima la solución si el algoritmo está muy regularizado (pero no tanto como para pensar que es cero)?

---

**1.8**  Se desea estimar la cantidad de agua que fluye por una presa a partir de la variación del nivel de agua. El archivo `represa.csv` contiene los datos a utilizar, definiendo los conjuntos de entrenamiento, validación y testeo.

(a) Visualice los tres dataset a partir de un gráfico `scatter` utilizando colores diferentes para cada conjunto.

(b) Realice una regresión lineal utilizando `LinearRegression` de sklearn. Grafique la recta de regresión estimada sobre la el `scatter`.

(c) Realice una regresión polinómica de orden 8 sin regularización. Grafique la función de regresión estimada sobre el `scatter`.

(d) Utilizando `sklearn.linear_model.Ridge`, repetir el inciso anterior regularizando con  $\lambda = 1$  y  $\lambda = 100$ .

(e) Graficar el error cuadrático medio en función del hiperparámetro de regularización  $\lambda \geq 0$  para el conjunto de entrenamiento y validación. ¿Que valor minimiza el error de validación?

(f) Calcular el error cuadrático medio de testeo para el hiperparámetro elegido en el inciso anterior.

---

## Guía 2

---

**2.1** Por un canal de comunicaciones se emiten bits de forma aleatoria, siendo el 75 % de ellos 1. Dependiendo del bit transmitido, la comunicación es afectada por un ruido aditivo normal de media nula y varianzas: 4 si el bit es un 0 y 1 si el bit es un 1. Sea  $X$  la señal recibida y  $Y$  el bit emitido.

- (a) Hallar y graficar  $p_X(x)$ .
- (b) Hallar y graficar  $P_{Y|X}(1|x)$ .
- (c) ¿Para que valores de  $x$  ocurre que  $P_{Y|X}(1|x) > P_{Y|X}(0|x)$ ?
- (d) Calcular el error bayesiano expresando el resultado de las integrales en función de  $\Phi(\cdot)$  (función de distribución de la normal estándar), para luego computar los cuantiles. Comparar el resultado contra un *clasificador al azar* y contra un *clasificador dummy*.

---

**2.2** Sean  $p$  y  $q$  dos distribuciones Bernoulli de parámetros  $\frac{1}{2}$  y  $\frac{1}{3}$  respectivamente. Calcular  $\text{KL}(p\|q)$  y  $\text{KL}(q\|p)$ . Expresar el resultado en nats.

---

**2.3** Sea  $p = \sigma(z)$  la función sigmoide.

- (a) Calcular la función inversa  $\sigma^{-1}(p)$  con  $p \in (0, 1)$ .
- (b) Calcular la derivada  $\sigma'(z)$ . Encontrar sus valores ínfimo y supremo, y (en caso que existan) los puntos donde los alcanza.
- (c) Escribir la derivada en función de  $p$ .

---

**2.4** 📖 Un profesor desea estimar si un alumno va a aprobar o no la materia en base a la nota de dos parcialitos. El archivo `parcialitos.txt` contiene una base de datos con las notas de cada estudiante en los parcialitos y si, efectivamente, aprobó o no la materia (1 es aprobar).

- (a) Hallar una expresión analítica para la función costo y su gradiente.
- (b) Realizar una regresión logística utilizando `LogisticRegression` (sklearn) y graficar la frontera de decisión sobre un `scatter`.
- (c) Predecir si un estudiante con notas 45 y 85 aprobaría la materia.
- (d) Graficar la curva ROC del clasificador, implementando su propio código. Indicar el punto correspondiente a la decisión tomada en el inciso (b) y el EER.

---

**2.5** 📖 El gerente de producción de una fábrica de circuitos integrados desea predecir si un determinado integrado pasará el control de calidad. El archivo `microchips.txt` posee datos de la evaluación de dos pruebas diagnóstico de diferentes integrados, y una tercer columna que indica si pasaron el mencionado control (1 es pasar la inspección).

- (a) Construir un mapa polinómico hasta orden 6 inclusive. ¿Como puede relacionar la cantidad de parámetros con el grado del polinomio y la cantidad de *features*? Encontrar una expresión matemática que vincule esas magnitudes.

(b) Realizar una regresión logística utilizando `LogisticRegression` (sklearn) y graficar la frontera de decisión sobre un `scatter` sin regularización.

(c) Realizar una regresión logística y graficar la frontera de decisión sobre un `scatter` con regularización L2 y  $\lambda = 1000$ .

(d) Realizar una regresión logística y graficar la frontera de decisión sobre un `scatter` con regularización L2 y  $\lambda = 1$ .

🔗: Funciones como `meshgrid` (numpy) y `contour` (matplotlib) pueden ser útiles para graficar las fronteras.

**2.6** 📖 La base de datos MNIST posee imágenes de los dígitos manuscritos (del 0 al 9). Se desea entrenar un clasificador que, a partir de una imagen, prediga que dígito aparece en ella.

(a) Cargar la base de datos utilizando `tensorflow.keras.datasets.mnist.load_data`. Utilizando `imshow` (matplotlib) represente 10 muestras del conjunto de testeo elegidas al azar.

(b) Realizar una regresión logística e indicar el *accuracy* de entrenamiento y testeo.

(c) Utilizando `ConfusionMatrixDisplay` (sklearn) represente la matriz de confusión normalizada (testeo) para mostrar la probabilidad de cada predicción para cada clase con 3 decimales.

**2.7** 📖 Se denomina formante a las frecuencias donde se dan los picos de intensidad en el espectro de un sonido. El archivo `formantes.txt` contiene ejemplos de los 3 primeros formantes del sonido de las vocales /a/, /o/ y /u/. Utilizando solamente los dos primeros formantes:

(a) Graficar las muestras en un `scatter`, representando los formantes de cada vocal con colores distintos.

(b) Superponer a la gráfica anterior las medias y las covarianzas de cada gaussiana (una curva de nivel) del modelo de LDA.

(c) Implementar un algoritmo de LDA para clasificar los formantes.

(d) Graficar la predicción de las muestras y la frontera de decisión.

(e) Generar 50 muestras sintéticas y graficarlas junto a las fronteras. Representar los formantes de cada vocal con colores distintos.

🔗: Tenga en cuenta que, además de las medias y varianzas, deberá utilizar las probabilidades  $c_k$  aprendidas durante el entrenamiento. Funciones como `random.choice` y `random.multivariate_normal` (numpy) pueden ser útiles.

**2.8** 📖 [ver **Ejercicio 2.5**] La fábrica de circuitos integrados desea predecir si un determinado integrado pasará el control de calidad a partir del archivo `microchips.txt`.

(a) Graficar la frontera de decisión de un algoritmo 1NN sobre el `scatter` de la base de datos. ¿Que puede decir del error de entrenamiento? ¿Puede extraer una conclusión general al respecto?

(b) Repetir para un 7NN. Relacionar el valor de  $K$  con los conceptos de *overfitting* y regularización.

(c) Graficar  $\hat{P}(1|x)$  para un algoritmo 1NN y 7NN entrenados solamente con la primera de las pruebas diagnóstico.

🔗: La función `argsort` (numpy) puede ser útil. El algoritmo KNN debe estar implementado con una sola función, de modo que ésta pueda usarse para todo  $k \in \mathbb{N}$  y cualquier cantidad de *features*.

**2.9** 📖 El archivo `ejs_svm.pkl` contiene un par de bases de datos. Utilizando la base de datos *1er Dataset*:

(a) Implementar una clasificación SVM utilizando `solve_qp` (qpsolvers), resolviendo el problema primal. Graficar la frontera de decisión y las rectas de vectores soportes sobre un `scatter`.

(b) Repetir el inciso (a) resolviendo el problema dual.

(c) Repetir el inciso (a) relajando los márgenes (utilizando  $C = 1$ ) y resolviendo el problema primal.

(d) Hallar el problema dual con márgenes relajados. Puede buscarlo en la bibliografía o deducirlo. 🔗: Se recomienda practicar la deducción.

(e) Repetir el inciso (a) relajando los márgenes (utilizando  $C = 1$ ) y resolviendo el problema dual.

**2.10** 📖 El archivo `ejs_svm.pkl` contiene un par de bases de datos. Utilizando la base de datos *2do Dataset*, implementar una clasificación SVM con Kernel gaussiano ( $\gamma = 50$ ) utilizando `svm.SVC` (sklearn) con  $C = 1$ . Graficar la frontera de decisión y las curvas de vectores soportes sobre un `scatter`.

**2.11** 📖 La cromatografía de ultra alta performance acoplada a espectrometría de masas de alta resolución permite el diagnóstico del cáncer de próstata. El archivo `prostata.csv` posee datos de la abundancia de concentración de diferentes compuestos químicos y el resultado del diagnóstico: sano, cáncer, benigno y post-cirugía. Se desea predecir el diagnóstico en función del resto de los indicadores.

(a) Las muestras sin etiquetas representan errores de medición. Construir un conjunto de datos con las muestras válidas.


(b) Utilizando `cost_complexity_pruning_path` (sklearn) y utilizando la entropía como impureza, calcular todos los  $\alpha$  relevantes para la poda de un árbol de decisión.

(c) Utilizando `GridSearchCV` (sklearn) optimizar el valor de  $\alpha$  para un 4-fold, utilizando como métrica la  $F_1$  macro. Graficar los valores de  $F_1$  cross-validada en función de  $\alpha$ . ¿Qué  $\alpha$  maximiza dicha métrica?

(d) Utilizando `plot_tree` (sklearn) graficar el árbol podado. Indicar la cantidad de nodos y hojas.

(e) Encontrar los 5 *features* más relevantes según la *Gini importance*.

🔗: Es importante resolver todo el ejercicio utilizando solo 1 `fit`.

**2.12**  [ver **Ejercicio 2.6**] La base de datos FASHION-MNIST posee la mismas características que la MNIST pero para clasificar 10 tipos de ropa. Se desea entrenar un clasificador que a partir de una imagen prediga que dígito aparece en ella.

(a) Cargar la base de datos utilizando `tensorflow.keras.datasets.fashion_mnist.load_data`. Utilizando `imshow` (matplotlib) represente 10 muestras del conjunto de testeo elegidas al azar. ¿Que tipo de prenda representa cada categoría?

(b) Utilizando `RandomForestClassifier` (sklearn), entrenar un bosque aleatorio de 100 árboles con impureza *Gini*. Indicar el *accuracy* de entrenamiento y testeo.

(c) Utilizando `ConfusionMatrixDisplay` (sklearn) represente la matriz de confusión normalizada (testeo) para mostrar la probabilidad de cada predicción para cada clase con 3 decimales.

(d) Graficar en una imagen los 100 píxeles más relevantes según la *Gini importance*. Indique con un punto negro los mencionados píxeles y deje el resto en blanco.

---



## Guía 3

### A. Aprendizaje No Supervisado

**3.1** Dos variables físicas  $(X, Y)$  poseen una densidad de probabilidad conjunta de la forma

$$p_{XY}(x, y) = \frac{e^{-(2x + \frac{y}{4x+2})}}{2x + 1} \cdot \mathbb{1}\{x > 0, y > 0\}.$$

Encontrar un mecanismo simple que permita generar una de las variables aleatorias en función de la otra y un ruido independiente. Sugiera que variable posiblemente sea la causa y cual el efecto.

🔗: Notar que si  $T \sim \mathcal{E}(\lambda)$ , entonces  $kT \sim \mathcal{E}(\lambda/k)$  con  $k > 0$ .

**3.2** 📖 [ver **Ejercicio 2.12**] Utilizando la base de datos FASHION-MNIST, se desea entrenar un algoritmo de PCA.

(a) Utilizando `linalg.eig` (numpy), calcular y graficar el porcentaje de energía en función del número de componentes principales.

(b) Graficar el error cuadrático medio de testeo en función del número de componentes principales.

🔗: Mucha atención a no repetir cálculos. Es decir, aprovechar la reconstrucción con  $k$  componentes principales para el cálculo de  $k + 1$  componentes.

(c) Graficar 10 imágenes reconstruidas de testeo (elegidas al azar) utilizando 1, 100 y 784 componentes principales.

(d) Graficar un `scatter` de las dos primeras componentes principales de las imágenes de testeo, indicando en diferentes colores las clases.

(e) Se desea evaluar el desempeño del algoritmo de PCA como detector de anomalías. Para ello, construir una base de datos combinando el conjunto de datos de testeo con el conjunto de datos de testeo de la base de datos MNIST (dígitos).

(f) Diseñar un detector de anomalías comparando el error cuadrático contra un umbral. Graficar la curva ROC y marcar el *equal error rate* para 1, 100 y 784 componentes principales. Interpretar resultados.

**3.3** 📖 [ver **Ejercicio 2.7**] Utilizando los dos primeros formantes de la base de datos `formantes.txt`:

(a) Implementar K-means para 3 clusters. Utilizar, como condición de parada, tanto cantidad de iteraciones (100 por ejemplo) como convergencia.

(b) Graficar un `scatter` de la clasificación final de los datos de entrenamiento, resaltando los centroides.

(c) Graficar las fronteras de decisión, superpuestos a un `scatter` con las verdaderas etiquetas.


**3.4** 📖 Se desea comprimir la imagen `pikachu_vs_charmander.jpeg` a 16 colores, utilizando `cluster.KMeans` (sklearn).

- (a) Tomando cada pixel como muestras diferentes, implementar un K-means de 16 clusters.
- (b) Utilizar los centroides como diccionario, para convertir cada pixel en un centroide (utilizando el algoritmo previamente entrenado). Utilizar `imshow` (matplotlib) para graficar la imagen ya codificada.
- (c) Calcular la cantidad de bits necesarios para guardar la imagen antes y después de comprimirla (teniendo en cuenta el etiquetado y los centroides).

**3.5** Los habitantes de *Smallville* pueden ser considerados *trabajador registrado*, *trabajador informal* o *desempleado* con probabilidades  $\frac{\theta}{2}$ ,  $\frac{1-\theta}{2}$  y  $\frac{1}{2}$  respectivamente, donde  $0 \leq \theta \leq 1$ . El municipio posee 10.000 habitantes y cuenta con 4.000 trabajadores registrados.

- (a) Estimar  $\theta$  por máxima verosimilitud.
- (b) Deducir matemáticamente una recursión, vía algoritmo EM, que permita estimar  $\theta$ .
- (c) Se denomina *puntos fijos* a los valores de  $\theta$  que no varían al iterar un paso del algoritmo. Encontrar los puntos fijos del problema de recursión definido por EM.
- (d) ¿A que punto converge el problema si  $\theta_0 = 0.99$ ? Analizar resultado.

**3.6** Se desea utilizar el algoritmo EM para aproximar la distribución de una variable aleatoria a una mezcla de gaussianas.

- (a) Generar 100 muestras de una mezcla de gaussianas con pesos 0.1, 0.4, 0.2, 0.3, medias  $-4, 0, 4, 5$  y varianzas 1, 1.96, 1.44, 1 respectivamente.
- (b) Implementar el algoritmo EM para entrenar una mezcla de 6 gaussianas a partir de los datos generados. Inicializar el algoritmo a partir de K-Means.  Si bien es evidente que los centroides representan las medias de las gaussianas, no es tan claro pensar como inicializar los pesos y varianzas. Justificar su criterio de inicialización.
- (c) Repetir el inciso anterior utilizando `GaussianMixture` (sklearn).
- (d) Graficar las dos densidades de probabilidad aprendidas, la densidad con que se inicializan (K-Means) y la verdadera en un mismo gráfico.

**3.7** La base de datos `fetch_olivetti_faces` (sklearn) contiene 400 imágenes de rostros.

- (a) Construir un código que cargue los datos de sklearn, cuando estos estén disponibles, y en caso contrario los cargue del archivo `olivetti.npy`. Elegir 6 imágenes al azar y graficarlas.
- (b) Encontrar las relaciones matemáticas que definen el algoritmo EM. Puede deducirlas (recomendado) o buscarlas en la bibliografía. En caso de elegir la segunda, explicar detalladamente como se implementaría.
- (c) Utilizando *FactorAnalysis* (sklearn) reducir la dimensión a un espacio latente normal de dimensión 2. Inicializar la matriz  $\Psi$  con la diagonal de la matriz de covarianza empírica de los datos y utilizar la implementación *lapack*.

(d) Se desea explorar el manifold del algoritmo entrenado previamente. Defina una grilla regular de  $10 \times 10$  entre  $[-2, 2]$ . Cada punto de esa grilla (100 en total) debe ser reconstruido en una imagen (utilizando el decoder entrenado) y mostrar los 100 rostros reconstruidos en una grilla representativa del espacio latente.

## B. Extracción de Features en aplicaciones específicas

---

**3.8** 📖 La base de datos `fetch_20newsgroups` (sklearn) posee textos sobre 20 tópicos diferentes.

(a) Construir un código que cargue los datos de sklearn, cuando estos estén disponibles, y en caso contrario los cargue del archivo `20news.pkl`. Cargar los conjuntos de entrenamiento y testeo omitiendo *headers*, *footers* y *quotes* de las categorías `alt.atheism`, `talk.religion.misc`, `comp.graphics`, `sci.space`.

(b) Utilizando `TfidfVectorizer` (sklearn) pre-procesar los datos.

🔗: Se recomienda convertir todo a minúsculas, utilizar como *Stop Words* las estándar del idioma inglés, descartar el 80 % de las palabras más frecuentes y utilizar la transformación tf-idf.

(c) Utilizando `LogisticRegression`(sklearn), entrenar un clasificador logístico y evaluar el *accuracy* con el conjunto de testeo.

---

**3.9** 📖 Se desea estudiar relaciones entre países y sus capitales. El archivo `country-list.csv` contiene la información correspondiente.

(a) Descargar las representaciones pre-entrenadas *FastText* en idioma inglés.

🔗: Con el siguiente código shell (linux) puede descargar el modelo:

```
wget https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.en.300.bin.gz
gzip -d cc.en.300.bin.gz
```

(b) Utilizando `FastText` (pyfasttext), cargar las representaciones pre-entrenadas.

(c) Utilizando `most_similar` (pyfasttext), armar una función que conteste a la pregunta *A es a B como C es a ...* dando 3 opciones posibles.

🔗: Ej. *France* es a *Paris* como *Brazil* es a ... (Brasilia).

(d) Combinar los países y las capitales de `country-list.csv` para entrenar un algoritmo PCA que reduzca la dimensión a 2, utilizando `decomposition.PCA` (sklearn). Graficar las representaciones reducidas de *Italy*, *Rome*, *France*, *Paris*, *Germany* y *Berlin* uniendo los países con sus capitales.


---


**3.10** 📖 Se desea incursionar en la temática de clasificación de género musical.

(a) Utilizando `load` (librosa) cargar los primeros 20 segundos del archivo `mi_perro_dinamita.mp3`. Graficar la señal temporal en función del tiempo (en segundos). Repetir con los primeros 20 segundos del archivo `exclusive.mp3`.

(b) Reproducir los audio utilizando `Audio` (IPython).

(c) Extraer los 12 primeros MFCC de cada señal utilizando `mfcc` (librosa). Concatenarlos y normalizar en media y varianza utilizando `StandardScaler` (sklearn).

- (d) Generar etiquetas que indiquen de que archivo provenía cada frame e implementar una clasificación SVM con Kernel gaussiano utilizando `svm.SVC` (sklearn) con  $C = 1$ .  El método aquí descrito está tratando los datos de cada ventana como muestras iid.
- (e) Construir una base de datos de testeo con los siguientes 30 segundos de cada audio. Evaluar el *accuracy* del clasificador en la nueva base de datos.
- (f) Repetir el ejercicio agregando los coeficientes  $\Delta$  y  $\Delta\Delta$  utilizando `delta` (librosa).
- (g) Repetir el ejercicio para diferentes duraciones de entrenamiento (variar entre 5 y 120 segundos). Graficar el *accuracy* de testeo en función de la duración, tanto usando los coeficientes dinámicos como sin usarlos. Para todas las duraciones usar el mismo conjunto de testeo (30 segundos de cada audio a partir de los 120 segundos). Relacionar los resultados con el concepto de *overfitting*.



**3.11**  En el archivo `instrumentos.zip` encontrará audios de diferentes instrumentos musicales. Se desea clasificar entre las 5 clases de instrumentos presentes en la base de datos. Reservando el último archivo de cada instrumento para el conjunto de testeo, se desea entrenar un clasificador adaptando el algoritmo EM como se menciona a continuación.


- (a) Extraer 13 primeros MFCC de cada señal y normalizar en media y varianza a partir del conjunto de datos de entrenamiento.
- (b) Entrenar 5 mezclas de 16 gaussianas diagonales cada una (una por cada instrumento).
- (c) Para todas las combinaciones de desea evaluar que tan verosímil es que las muestras de la clase  $i$ -ésima correspondan al modelo  $j$ -ésimo. Indicar las *log-verosimilitud* correspondientes en un cuadro de doble entrada.
- (d) Asumiendo una probabilidad de cada clase proporcional a la cantidad de muestras de entrenamiento de cada instrumento, calcular las probabilidades a posteriori. Es decir, indicar en un cuadro de doble entrada la probabilidad de la clase  $j$ -ésima para las muestras correspondientes al instrumento  $i$ -ésimo  $P(j|\mathcal{D}_i)$ .
- (e) Sea  $x(t)$  la señal correspondiente al audio de testeo del saxofón. Graficar  $P(j|x(t))$  en función del tiempo, para cada uno de los 5 instrumentos (indexados por  $j$ ).
- (f) ¿Como cambia la performance si se utilizando covarianzas full? Relacionar con el concepto de *overfitting*.

## Guía 4

---


**4.1** Lucas dispara a un blanco y el disparo impacta en un punto aleatorio  $(X, 0)$  con  $X$  (en decímetros) una variable aleatoria con distribución normal de media nula y varianza  $1/\tau$ , donde  $\tau$  representa la precisión de Lucas. A priori la precisión  $\tau$  tiene una distribución chi-cuadrado de 8 grados de libertad. Lucas tiro 10 veces al blanco y observó  $\sum_{i=1}^{10} x_i^2 = 17$ . En virtud a la información muestral,


- (a) Hallar la distribución *a posteriori*.
  - (b) Hallar la distribución predictiva. : Mirar con cariño la distribución t-student.
  - (c) Estimar la probabilidad de que su próximo tiro se aleje del centro en menos de 2.1 decímetros. : Se recomienda calcular el cuantil con un software.
- 

**4.2**  [ver **Ejercicio 2.7**] Utilizando los dos primeros formantes de la base de datos `formantes.txt`:

- (a) Utilice el comando `train_test_split` (sklearn) para definir dos conjuntos con las proporciones 70 % y 30 %.
  - (b) Implementar un modelo GNB. Graficar las muestras en un `scatter`, las fronteras de decisión y resaltar medias y covarianzas de cada gaussiana (una curva de nivel) del modelo.
  - (c) Repetir el inciso anterior para LDA y QDA, programando su propio código.
  - (d) Clasificar el conjunto de testeo con cada método. Indicar la porcentaje de acierto.
- 


**4.3**  [ver **Ejercicio 3.8**] Utilizando `fetch_20newsgroups` (sklearn):

- (a) Cargar los conjuntos de entrenamiento y testeo omitiendo *headers*, *footers* y *quotes*.
  - (b) Utilizando `CountVectorizer` (sklearn) pre-procesar los datos.  
: Se recomienda convertir todo a minúsculas, utilizar como *Stop Words* las estándar del idioma inglés, descartar el 95 % de las palabras más frecuentes y descartar las palabras vistas 1 sola vez.
  - (c) Bajo las hipótesis de *Multinomial Naive Bayes* con  $\alpha = (1, 1, \dots, 1)$ , hallar los estimadores bayesianos puntuales para las probabilidades de cada palabra.
  - (d) A partir de la estimación anterior, evaluar el *accuracy* con el conjunto de testeo.
- 

**4.4**  [ver **Ejercicio 3.6**] Utilizando los datos del ejercicio **Ejercicio 3.6**, implementar un Variational Bayes Gaussiano. Suponer *a priori*  $m = 0$ ,  $\delta = \nu = \beta = 0.05$  y  $\alpha = (1, 1, 1, 1, 1, 1)$ , y utilizar el algoritmo EM para inicializar las probabilidades.

- (a) Con la distribución *a posteriori* generar 3 muestras de parámetros y graficar la densidad de  $X|\mu, \lambda, \pi$  para cada uno de esos conjuntos de parámetros. Comparar con la densidad verdadera y con la estimada por el algoritmo EM.

(b) Graficar la densidad *predictiva*. Comparar con la densidad verdadera y con la estimada por el algoritmo EM.


: Las funciones `gamma` y `digamma` (scipy) pueden ser útiles.


**4.5**  Estimar mediante un Monte-Carlo de 20 puntos  $\int_{-1}^2 \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt$  si

(a) Las variables aleatorias se sortean de forma uniforme.

(b) Las variables aleatorias se sortean de forma normal.

Repetir el experimento 100 veces e indicar el error promedio en cada caso.

**4.6**  [ver **Ejercicio 4.4**] Utilizando los datos del ejercicio **Ejercicio 3.6**, entrenar un modelo *Markov Chain Monte-Carlo* con `pymc`. Suponer *a priori*  $m = 0$ ,  $\delta = \nu = \beta = 0.05$  y  $\alpha = (1, 1, 1, 1, 1, 1)$ , y simular 3 cadenas de 500 experimentos cada una. Graficar la densidad *predictiva* para cada cadena. Comparar con la densidad verdadera, con la estimada por el algoritmo EM y con la predictiva estimada utilizando Bayes Variacional.

**4.7**  Se desean modelar los resultados de un partido de futbol. Para esto, se propone asumir que la cantidad de goles que marca un equipo es una función de su potencia ofensiva combinada con la fuerza defensiva del rival. También se debe tener en cuenta la ventaja de la localía. El modelo se verá así:

- potencial local = ventaja localía + ataque local + defensa visitante
- potencial visitante = ataque visitante + defensa local




Notar que mientras tener alto ataque es positivo, tener alta defensa es negativo. Cada equipo tendrá su propio *rating* de ataque y defensa, pero estás dependerán de una distribución común (si bien cada equipo de fútbol es claramente diferente entre sí, todos juegan en la misma liga). Vamos a suponer que el ataque y defensa (inicial) de cada equipo son variables aleatorias normales de media nula. *A priori*, la precisión (inversa de la varianza) de ataque y la de defensa (comunes a todos los equipos) serán variables aleatorias gamma de parámetros 0.1 y 0.1.

Dado que existen innumerables combinaciones de parámetros que potencialmente podrían darnos los mismos resultados, se propone restar las calificaciones medias (es decir, el promedio de todos los equipos) tanto de ataque como defensa para garantizar la identificabilidad. Esto obliga al modelo a brindarnos un resultado reproducible y mantiene los parámetros en un rango realista. De esta manera se construye el ataque y defensa final de cada equipo.

Una vez definidos los potenciales tanto del local como del visitante, es necesario modelar la cantidad de goles de cada equipo. La cantidad de goles de cada equipo se supondrán Poisson de media  $e^{\text{potencial}}$  (para que sean positivos).

Por último, para la ventaja de localía se supondrá una *flat-distribution prior* (buscar información al respecto).

(a) La base de datos `liga_arg.csv` contiene todos los resultados del futbol argentino desde el profesionalismo. Armar un conjunto de datos de entrenamiento con todas las competiciones iniciadas en el 2020 o posterior.

- (b) Construir la arquitectura descripta. Utilizando `model_to_graphviz` (pymc) mostrar el grafo del modelo. : En el modelo aquí descripto no tiene mucha relevancia las muestras predcitivas, sino que el foco debe estar sobre las posteriors.
- (c) Entrenar el modelo bayesiano con los datos del inciso (a). Utilizando `plot_posterior` (pymc) graficar la distribución a posteriori de la ventaja de localía.
- (d) Reportar los 5 equipos con mejores ataques esperados. Repetir con los de mejores defensas.
- (e) Diseñar una función que, dado un equipo local y uno visitante, estime la cantidad de goles esperados por cada equipo en un partido. Estimar el valor esperado del resultado global en un cruce ida y vuelta entre River y Boca (dos partidos, uno local River y el otro local Boca). : Si  $X|A = a \sim \text{Poi}(e^a)$ , entonces  $\mathbb{E}[X] = \mathbb{E}[e^A]$ .
- (f) Diseñar una función que, dado un equipo local y uno visitante, estime la probabilidad que gane el local, empate y gane el visitante. : Si  $X|A = a \sim \text{Poi}(e^a)$  y  $Y|B = b \sim \text{Poi}(e^b)$ , entonces  $\mathbb{P}(X > Y) = \sum_{i=0}^{\infty} \sum_{j=0}^{i-1} \mathbb{E}[\mathbb{P}(X = i, Y = j|A = a, B = b)]$ . Para su implementación numérica asuma que la máxima cantidad de goles por un equipo es 10 (a la hora de implementarlo la serie se aproxima con una sumatoria hasta 10).
- (g) Construir un conjunto de datos de testeo con la última fecha jugada del campeonato local al momento de resolver el ejercicio (solo tener en cuenta los partidos donde ambos equipos aparezcan en el conjunto de datos de entrenamiento). Utilizar como predicción *hard* la máxima probabilidad estimada dentro de las 3 clases definidas en el inciso anterior. Reportar el porcentaje de acierto.

## BIBLIOGRAFÍA SUGERIDA

1. “Pattern Recognition and Machine Learning”, C. Bishop.
2. “The Elements of Statistical Learning: Data Mining, Inference, and Prediction”, J. Hastie, T. Tibshirani, R. Friedman.
3. “Machine Learning: A Probabilistic Perspective”, K. Murphy.
4. “Introduction to Machine Learning with Python: A Guide for Data Scientists”, A. Müller, S. Guido.
5. “Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference”, C. Davidson-Pilon.
6. “Pattern Classification”, R. Duda, P. Hart, D. Stork.
7. “Deep Learning”, I. Goodfellow, Y. Bengio, A. Courville.
8. “Elements of Information Theory”, T. Cover, J. Thomas.
9. “Elements of Causal Inference: Foundations and Learning Algorithms”, J. Peters, D. Janzing, B. Schölkopf.
10. “Foundations of Machine Learning”, M. Mohri, A. Rostamizadeh, A. Talwal-kar.
11. “Data Analysis: A Bayesian Tutorial”, D. Sivia and J. Skilling.
12. “The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation”, C. Robert.