

# Tarea 4

SVM y Random Forest

**Profesor: Pablo Estévez**

Auxiliar: Pablo Cornejo

Ayudantes: Diego Castillo, Andrés González, Sebastián Guzmán, Francisco Soto

## Instrucciones generales:

- **Importante:** Todos los plots realizados **deben** tener una etiqueta en el eje X y en el eje Y, además de un título. Las etiquetas y los títulos son libres a ser determinados por cada uno, sin embargo, deben estar relacionados con el contexto del gráfico.
- Por favor comentar los códigos. Si hay subsecciones se recomienda comentar los lugares donde comienza cada subsección.
- Utilicen nombres significativos para variables, funciones y clases. Esto les ayudará principalmente a ustedes cuando lean sus códigos en un futuro y también al resto de la gente con la que trabajen luego.

## Instrucciones específicas:

- Se deberá subir a u-cursos un informe con los resultados y figuras obtenidos, además del notebook utilizado en la tarea.
- Se recomienda que la extensión del informe no supere las 12 páginas.
- Los modelos y datasets reutilizan los nombres de las variables. Revise que los resultados que esté observando correspondan al modelo y datos correctos. Sea ordenado al programar.
- Se recomienda la utilización de python local en su computador en vez de Google Colaboratory, ya que en este último no existe soporte para la última versión de scikit-learn, la que es necesaria para esta tarea.

## Parte Teórica

1. Imagine que está entrenando un modelo *Random Forest*. Dada una característica elegida para dividir el espacio ¿qué criterio se utiliza para colocar el umbral de clasificación? ¿Cómo se decide qué característica se utilizará para dividir el conjunto de muestras en un nodo?
2. En los modelos *Random Forest* cada árbol de decisión clasifica utilizando un subconjunto de las características disponibles. Explique por qué se hace esto.
3. ¿Qué son los métodos de kernel? ¿Qué ventaja tiene llevar los puntos a un espacio distinto? ¿Es necesario conocer la función que mapea los puntos al nuevo espacio para aplicar los métodos de kernel?

## Parte experimental

En esta tarea usted deberá resolver dos problemas de clasificación binaria utilizando los conjuntos de datos *two moons* y *Covertypes dataset*. Para esto usted utilizará (a) Support Vector Machines con algoritmo C-SVM y (b) Random Forest. En el caso de SVM se pide que compare SVM lineal y SVM no-lineal con kernel Gaussiano. Adjunto a esta tarea encontrará un Jupyter notebook (EL4106\_Tarea4.ipynb). Utilícelo para responder los puntos mencionados a continuación.

### 1. Two Moons

1. Importe las librerías necesarias corriendo el bloque B.1. No es necesario ejecutar este bloque nuevamente mientras el notebook esté activo.
2. Genere el conjunto *two moons* usando el bloque B.2. Esto creará conjuntos de entrenamiento y validación de manera automática.
3. Para el caso de SVM, usted debe comparar el caso SVM lineal (donde sólo debe calibrar  $C$ ) y el caso de SVM con kernel Gaussiano. Para ajustar  $C$  y  $\gamma$  del kernel Gaussiano, comience fijando  $C = 1$  y pruebe con  $\gamma \in \{0.1, 1, 10, 100\}$ . Luego, fijando  $\gamma$  en el mejor caso anterior, ajuste  $C \in \{0.1, 1, 10, 100\}$ . El parámetro  $\gamma$  corresponde al inverso del radio de influencia de las muestras (ver fórmula en la API de **sklearn**). Grafique el AUC versus el valor de los hiperparámetros probados, tanto para el kernel lineal como Gaussiano. Elija los parámetros que maximizan el AUC de validación. Describa el efecto que tiene la elección de kernel y los hiperparámetros en el hiperplano separador.
4. Fije el número de árboles del modelo *Random Forest* en 1 (`n_estimators=1`). Compare cualitativamente la forma en que el modelo *Random Forest* divide el espacio al cambiar la profundidad máxima del árbol entre los valores 3 y 10. Comente respecto al *trade-off* entre simplicidad y sobreajuste.
5. Evalúe las 4 combinaciones de número de árboles y profundidad máxima en  $\{7, 50\} \times \{3, 10\}$ . Presente los valores de AUC en validación mediante una matriz y elija la mejor combinación como modelo final.

## 2. Covertypes dataset

*Covertypes dataset* es una base de datos que contiene 581,012 observaciones del tipo de vegetación dominante en diversas áreas del Roosevelt National Forest en Colorado, USA. El objetivo es predecir la clase de vegetación en diversos lugares de entre las 7 clases existentes, solo a partir de información cartográfica, para lo cual se proveen 54 características por lugar.

1. Cargue la base de datos de *Covertypes dataset* ejecutando el bloque B.5. Comente acerca del desbalance entre clases que presenta la base de datos.
2. Una técnica para lidiar con el desbalance es asignar diferentes pesos a cada muestra, de forma inversamente proporcional a la frecuencia de la clase a la que pertenece. Evalúe el efecto de la corrección de desbalance por ponderación que tiene *Random Forests*. Para esto, cambie el flag `class_weight` entre `balanced` y `None`. Use 30 árboles con profundidad máxima 10. ¿Cómo cambia la cantidad de ejemplos clasificados correctamente en validación? ¿Cuál de estas configuraciones recomendaría usted?
3. Usando el modelo *Random Forest* con corrección por desbalance, mida qué valor de profundidad máxima en el conjunto {10, 30, 50} entrega mejores resultados en validación.
4. Dado el mejor modelo hallado, ¿cuáles son las características más importantes según *Random Forest*? ¿Qué porcentaje de las características se necesitan para “capturar” el 80 % de la importancia?

## Programación

Compare el mejor resultado obtenido por el modelo *Random Forest* con un C-SVM de acuerdo con los siguientes pasos:

1. Construya un subconjunto del dataset *Covertypes* con 1000 muestras por cada clase tomadas al azar.
2. Particione dicho subconjunto en un conjunto de entrenamiento y validación, con proporciones 70 % y 30 % respectivamente. Utilice la función `train_test_split` de `sklearn` con el parámetro `stratify`.
3. Normalice cada característica restando la media y dividiendo por la desviación estándar. Puede utilizar el `StandardScaler` de `sklearn`. Recuerde computar la media y desviación estándar utilizando sólo datos del conjunto de entrenamiento.
4. Entrene una C-SVM multiclase del tipo *one-versus-all* (`decision_function_shape="ovr"`). Utilice kernel Gaussiano, probando al menos 5 combinaciones de los hiperparámetros `C` y `gamma`.
5. Muestre la tasa de acierto y el *recall* promedio en validación para el mejor modelo. Grafique la matriz de confusión normalizada de la mejor SVM encontrada. Compare con el *Random Forest*.
6. Incluya las secciones más relevantes del código escrito en su informe.