

## Tarea 4: Máquinas de soporte vectorial

**Profesor:** Felipe Tobar

**Auxiliares:** Mauricio Araneda, Alejandro Cuevas, Mauricio Romero

**Consultas:** Todo el cuerpo docente.

**Fecha entrega:** 23/6/2019

**Formato entrega:** Entregue un informe en formato PDF con una extensión de a lo más 2 páginas presentando y analizando sus resultados, detalle la metodología utilizada y adicionalmente debe entregar un jupyter notebook con los códigos que creó para resolver la tarea.

*Esta es una de las 2 versiones de la tarea4, deben elegir solo una de las versiones (SVM o NN).*

### P1. Clasificación con SVM

Se pretende realizar la tarea de clasificación sobre el dataset MNIST, usando Support Vector Machines (SVM) como clasificador, para esto:

- a) (1 pts.) Cargue los datos (Vea el demo para cargarlos) y sepárelos de forma aleatoria en conjuntos de entrenamiento 5/7, validación 1/7 y prueba 1/7 (si carga con pytorch el conjunto de prueba ya está definido).

Normalice de forma estándar los datos de modo que tengan media 0 y varianza unitaria, es decir,  $\mathbf{x}_{\text{new}} = \frac{\mathbf{x} - \mu}{\sigma}$  donde  $(\mu, \sigma)$  es la media y desviación estándar del conjunto de entrenamiento por dimensión.

*En caso de cargar los datos con Pytorch:* Transforme sus datos de tal forma que cada imagen de  $28 \times 28$  sea un vector de tamaño 784.

**Observación:** Le puede ser útil `train_test_split` y `StandardScaler` de `sklearn`.

- b) (2 pts.) Visualice el conjunto de datos sobre un gráfico de dos dimensiones para el conjunto de entrenamiento mediante PCA. Debe marcar con un color distinto los elementos dependiendo de la clase a la que corresponden. ¿Se distinguen tendencias para la separación del conjunto en sus clases? ¿Qué clases son más fáciles de distinguir de otras?
- c) (3 pts.) Implemente tres funciones de kernel a utilizar en SVM, una de ellas debe ser el kernel Gaussiano o RBF, las otras 2 son a elección.
- Para esto implemente una función que tome como argumentos los conjuntos de samples  $X_1$  y  $X_2$  y retorne una matriz de  $n_{\text{samples}1} \times n_{\text{samples}2}$  con la evaluación del kernel, es decir el elemento  $(i, j)$  de dicha matriz es igual a  $k(X_1(i), X_2(j))$  la evaluación del kernel del elemento  $i$  con el elemento  $j$ .
- d) (3 pts.) Entrene un clasificador SVM con 5 distintos valores de  $C$  ( $\star$ ) usando su kernel RBF implementado. Muestre los errores totales (i.e., datos mal clasificados) que comete su clasificador SVM en función de distintos valores de  $C$ , utilizando un kernel de su elección, tanto para los datos de entrenamiento como los de validación. Vea para qué valor de  $C$  se obtiene el mínimo número de errores tanto en los datos de entrenamiento y validación, luego evalúe en el conjunto de test. Comente sobre las diferencias de los resultados obtenidos.
- e) (3 pts.) Entrene un SVM con kernel los tres kernel previamente definidos (NO puede usar los kernels predefinidos de SVC), considerando el mejor  $C$  encontrado en la parte d). Analice sus resultados comparando el desempeño de los distintos kernels mediante la *accuracy* y matriz de confusión sobre el conjunto de validación y test. Discuta sus resultados.

**Observaciones:** Recomendaciones generales para valores de parámetros y documentación extra:

- \* Le puede ser útil la documentación de sklearn <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.
- En la parte **c)** le puede ser útil considerar alguno de los kernels definidos en el siguiente enlace: <http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/>
- En la parte **c)** En su implementación **evite hacer uso de ciclos for**, aproveche la notación matricial, de otro modo su código puede ser muy ineficiente, para generar la matriz de distancias le puede servir [https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance\\_matrix.html](https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance_matrix.html).
- El tiempo de entrenamiento para el clasificador debería tomar al rededor de 2 a 6 minutos cada vez que se instancie.
- En la parte **d)** considere  $C \in [0, 1]$ .