# Tarea 4-Reconocimiento de Patrones

Felipe Neut

## I. Motivacion

*Deep learning* se ha convertido en uno de los algoritmos más importantes del siglo XXI en el área de computación y muchas innovaciones tecnológica. Es por ello que el estudio de una red neuronal en esta tarea es altamente importante y motivante. A su vez, se verá el uso del clasificador SVM, que posee una gran relevancia dada su elevada velocidad para realizar el *testing* mediante el uso de regresiones.

## II. Solución Propuesta

Como bien se ha estudiado, los datos del conjunto *testing* no pueden ser utilizados en ningún paso del *training*. Es por ello que, para esta tarea se entrenará un conjunto de *training* buscando maximizar el *accuracy* sobre un conjunto *testing* de validación. Esta técnica permite extraer resultados en base a prueba y error sobre que clasificadores y parámetros utilizar posteriormente sobre el conjunto *testing*, para de este modo esperar obtener un *accuracy* en *testing* cercano o mayor al obtenido en validación.

El primer paso utilizado fue usar el comando *Bio_plotfeatures()* del *ToolBox Balu* para analizar como estaban siendo repartidas las 2 clases para las 2 features existentes, entregando como resultado una gran mezcla de las clases unas con otras, lo que incita a usar algoritmos más complejos que algoritmos lineales para el *training*.

→ **Redes Neuronales**: Se utilizó el método descrito en el diagrama de bloques de la Figura 1. Es importante destacar

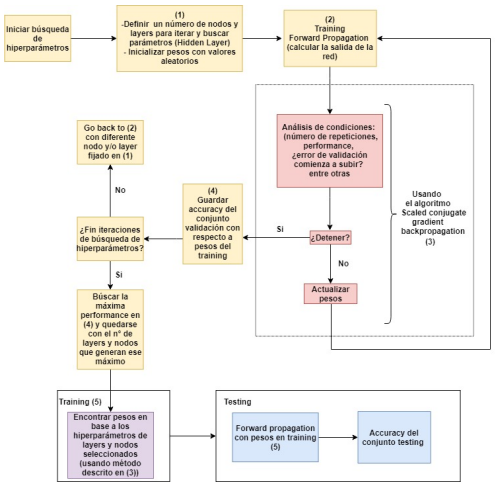


Figura 1. Método redes neuronales (Realizar *zoom*)

que para obtener el *accuracy* en *testing* (A2) se realizó el promedio de 10 iteraciones de la red neuronal formada a partir del número de *layers* y nodos que entregó la etapa de búsqueda de hiperparámetros, lo cual era necesario por la aleatoriedad de los pesos iniciales con los que comienza el algoritmo.

→ **SVM**: Por temas de espacio no se puedo incluir un diagrama de bloques. Para la búsqueda de hiperparámetros se definieron las siguientes funciones *Kernel* del algoritmo SVM para iterar: *linear*, polinomio de grado 3, *rbf* y sigmoide, escogiendo la que reportara el mayor *accuracy* del conjunto validación con respecto a la línea de decisión encontrada por SVM en *training*. Luego, se procedió a realizar el *testing* según la función *Kernel* encontrada en la etapa de hiperparámetros, reportando el *accuracy obtenido*.

## III. Resultados

En la búsqueda de hiperparámetros, se iteró sobre un máximo de 3 *layers*, y un total de 5 a 20 nodos por *layer*, obteniendo el máximo *accuracy* en validación a partir de 3 *layers* y un total de 10, 13 y 9 nodos respectivamente. En base a lo anterior, el *accuracy* obtenido del promedio de 10 iteraciones fue de 0.8740 en validación y 0.8607 en *testing*, valores que fluctuan levemente por la aleatoriedad de los pesos iniciales. Se tuvo especial cuidado de no sobre-entrenar a la red, de modo de detener el algoritmo en el mínimo de error y no permitir que el error de validación comenzara a subir. De la Figura 2 se ve el correcto mínimo global captado.


Figura 2. Mínimo de error captado en redes neuronales

Con respecto a SVM, se obtuvo que la función *Kernel* que reportaba el mayor *accuracy* en validación fue *rbf*, con un valor de 0.88, a diferencia de un: 0.805 con una función lineal, 0.87 con un polinomio de grado 3 y un 0.47 con la función sigmoide como *Kernel* de SVM. Los resultados anteriores llevaron a elegir la función exponencial no lineal *rbf* en SVM para realizar el *training*, cuya línea de decisión reportó un 0.8733 en el conjunto *testing*, con un valor de *gamma* de 0.5.

## IV. Conclusiones

Se pudo ver que el método empleado de utilizar un conjunto de validación para encontrar hiperparámetros que serán utilizados posteriormente para realizar el *testing* resultó ser una muy buena alternativa, reportando el *testing accuracies* muy cercanos a los maximizados en validación. La alta mezcla de las clases, no permitió obtener valores muy elevados de *accuracies*: $\approx 0{,}87$ con redes neuronales y $\approx 0{,}88$ con SVM.