# 1C 2019 - IA

# UTN FRBA

# 5 May 2019

Inteligencia Artificial

Ingeniería en Sistemas de Información

Año 2019

|  |
| --- |
| Trabajo Práctico N° 2  Entrega Final |

|  |  |
| --- | --- |
| **GRUPO N°6** | |
|  |  |
| **Apellido y Nombres** | **Dirección de E-Mail** |
| Gabriel Arce | [arce.gerardogabriel@gmail.com](mailto:arce.gerardogabriel@gmail.com) |
| Rocio Filgueira | [we.are.torchwood@gmail.com](mailto:we.are.torchwood@gmail.com) |
| Lilian Patricia Luque Duarte | [lilieclipse@gmail.com](mailto:lilieclipse@gmail.com) |
| Francisco Patalagoity | [frp2007.2008@gmail.com](mailto:frp2007.2008@gmail.com) |
| Axel Furlan | [axel.furlan95@gmail.com](mailto:axel.furlan95@gmail.com) |

|  |  |
| --- | --- |
| **Fecha de Presentación** | 19/05/2019 |

Universidad Tecnológica Nacional

Facultad Regional Buenos Aires



Aplicación de redes neuronales artificiales para la predicción de enfermedad cardiaca en individuos

Axel Furlan, Gabriel Arce, Rocio Filgueira, Lilian Patricia Luque Duarte, Francisco Patalagoity

Universidad Tecnológica Nacional. Facultad Regional Buenos Aires.

Buenos Aires. Argentina

Resumen

En el siguiente trabajo práctico. Utilizaremos el dataset de [enfermedad cardiaca proporcionado](https://www.kaggle.com/ronitf/heart-disease-uci) por Kaggle. Nuestro objetivo es analizarlo a grandes rasgos y construir una Red Neuronal Artificial para predecir la presencia de enfermedad cardiaca en futuros individuos, a través de mediciones y datos relacionados a la salud de la persona.

Para esto, utilizaremos Python 3 como lenguaje y [sklearn](https://scikit-learn.org/) como librería para hacer feature-engineering en el dataset y construir la red.

Como añadido a este trabajo, incluimos junto al mismo un documento generado mediante [Jupyter](https://jupyter.org/), un proyecto usado extensamente en los estudios estadísticos mediante código, que permite hacer Explanatory Data Analysis con código y explicación escrita. Consideramos esto valioso para poder explicar paso a paso qué es lo que hacemos en cada instrucción de código.

Palabras clave: redes neuronales, enfermedad cardiaca, salud, aprendizaje automático, inteligencia artificial, sistemas inteligentes, backpropagation

Introducción

Elegimos el presente tema porque presenta una problemática preocupante en la salud de las personas pasados aproximadamente los 45 años (según indican los datos). De acuerdo a la World Health Organization, 17.9 millones de personas mueren cada año por enfermedades cardiovasculares, representando un 31% de las muertes alrededor del mundo [[1]](#footnote-1). A su vez, el 85% de las muertes ocasionadas por CVD (*Cardio Vascular Disease)* son por paros cardiacos y ataques al corazón. Las CVDs (sus siglas en inglés) engloban desordenes del corazón y vasos sanguíneos e incluyen la enfermedad cardiaca coronaria, accidentes cerebro vasculares (ACV), enfermedad cardiaca reumática entre otras condiciones. Los individuos en riesgo de CVD pueden demostrar un incremento en su presión sanguínea, en glucosa y en lípidos, asimismo padecer sobrepeso y/o obesidad.

Datos

Contamos con 303 ejemplos, con 13 dimensiones y 1 variable objetivo. Dejamos la descripción que nos proporciona Kaggle:

* age edad en años
* sex (1 = hombre; 0 = mujer)
* cp tipo de dolor de pecho (de 0 a 3)
* trestbps presión sanguínea en descanso (en mm Hg)
* chol colesterol en mg/dl
* fbs(azúcar en sangre en ayunas > 120 mg/dl) (1 = verdadero; 0 = falso)
* restecg resultados electrocardiogáficos en descanso
* thalach ritmo cardiaco máximo obtenido
* exang dolor en el pecho inducido por ejercicio (1 = si; 0 = no)
* oldpeak *ST depression[[2]](#footnote-2)* inducido por ejercicio relativo al descanso
* slope pendiente de ejercicio máximo de *ST segment[[3]](#footnote-3)*
* ca número de vasos sanguíneos (0-3) colorados por flourosopía
* thal 3 = normal; 6 = defecto fijo; 7 = defecto reversible
* target posee o no enfermadad cardiaca (1 = si ; 0 = no)

Desarrollo

# Librerías y herramientas

Para la generación de la red neuronal, utilizamos el modelo de Multi Layer Perceptron de la librería Scikit Learn[[4]](#footnote-4). Sklearn––su acrónimo––nos permite modificar varios atributos de la red neuronal.

# Topología y entrenamiento de la red neuronal

En nuestro caso, establecemos únicamente la cantidad de *capas ocultas* y la cantidad de neuronas por capa, este parámetro es hidden\_layer\_sizes (también descrito en el notebook de Jupyter) y recibe como tipo de dato una tupla de 2 elementos: (cantidad\_de\_neuronas\_para\_capa, cantidad\_de\_capas). A modo de prueba y error, desarrollamos código para iterar y probar diferentes configuraciones de capas y redes.

El método de entrenamiento de la red neuronal es Backpropagation—tal cual está establecido por Sklearn [aquí](http://www.apple.com).

La clase que utilizaremos— MLPClassifier—utiliza el método fit()que recibe una lista de cantidad de ejemplos y cantidad de atributos como primer parámetro y una lista con elementos y 1 sola dimensión—la variable objetivo *y*.

# Cálculo de error y análisis de resultados

Para la prueba del modelo, utilizamos una técnica que consiste en dividir el dataset en 2, un dataset para entrenamiento del modelo y otro para pruebas. De esta forma, contamos con datos *supervisados* para poder probar la efectividad de nuestro modelo. Como es aclarado en el código, utilizamos un 20% del dataset para pruebas: 61 ejemplos.

Comprobamos el error y la precisión de cada iteración de configuración de la red utilizando la definición de *precisión*[[5]](#footnote-5):

[[6]](#footnote-6)

A modo de analizar mejor los resultados obtenidos y los errores de Tipo I como de Tipo II, utilizamos la llamada *matriz de confusión*[[7]](#footnote-7) que nos permite ver a muy simple vista, el número de errores y aciertos de nuestro clasificador.

Resultados

Probamos diferentes configuraciones de la red, que van de 1 capa escondida con 3 neuronas a 5 capas con 12 neuronas cada una. En total son 50 combinaciones. A continuación los resultados:

| neurons | layers | score |
| --- | --- | --- |
| 10.0 | 3.0 | 0.901639 |
| 11.0 | 3.0 | 0.885246 |
| 12.0 | 1.0 | 0.885246 |
| 8.0 | 2.0 | 0.868852 |
| 7.0 | 2.0 | 0.868852 |
| 5.0 | 2.0 | 0.868852 |
| 9.0 | 5.0 | 0.852459 |
| 12.0 | 4.0 | 0.836066 |
| 3.0 | 4.0 | 0.836066 |
| 5.0 | 3.0 | 0.836066 |
| 12.0 | 5.0 | 0.836066 |
| 6.0 | 5.0 | 0.836066 |
| 9.0 | 1.0 | 0.836066 |
| 10.0 | 5.0 | 0.836066 |
| 11.0 | 4.0 | 0.819672 |
| 3.0 | 5.0 | 0.819672 |
| 7.0 | 5.0 | 0.819672 |
| 9.0 | 4.0 | 0.819672 |
| 11.0 | 2.0 | 0.819672 |
| 5.0 | 4.0 | 0.819672 |
| 3.0 | 3.0 | 0.819672 |
| 4.0 | 3.0 | 0.819672 |
| 7.0 | 3.0 | 0.819672 |
| 7.0 | 4.0 | 0.803279 |
| 11.0 | 5.0 | 0.803279 |
| 10.0 | 4.0 | 0.803279 |
| 8.0 | 3.0 | 0.803279 |
| 4.0 | 2.0 | 0.803279 |
| 5.0 | 1.0 | 0.803279 |
| 6.0 | 3.0 | 0.803279 |
| 4.0 | 1.0 | 0.803279 |
| 10.0 | 2.0 | 0.803279 |
| 6.0 | 2.0 | 0.803279 |
| 9.0 | 2.0 | 0.786885 |
| 6.0 | 1.0 | 0.786885 |
| 8.0 | 1.0 | 0.786885 |
| 9.0 | 3.0 | 0.786885 |
| 6.0 | 4.0 | 0.786885 |
| 12.0 | 3.0 | 0.786885 |
| 8.0 | 5.0 | 0.770492 |
| 7.0 | 1.0 | 0.770492 |
| 3.0 | 2.0 | 0.754098 |
| 12.0 | 2.0 | 0.754098 |
| 8.0 | 4.0 | 0.737705 |
| 11.0 | 1.0 | 0.737705 |
| 5.0 | 5.0 | 0.737705 |
| 4.0 | 5.0 | 0.704918 |
| 4.0 | 4.0 | 0.672131 |
| 10.0 | 1.0 | 0.557377 |
| 3.0 | 1.0 | 0.557377 |

Como podemos ver, la configuración que nos generó mejores resultados es de 3 capas ocultas con 10 neuronas cada una, con un increíble % de precisión. La que posee peor *scoring* es 1 sola capa con sólo 3 neuronas, con un decepcionante %. Esto es entendible ya que considerando que poseemos 13 *features,* hacer una capa oculta con sólo 3 neuronas ocasionará que se “junten” los coeficientes *W* propagando el error para todas las predicciones.

La matriz de confusión para esa configuración presenta 25 verdaderos negativos y 30 verdaderos positivos, con 4 falsos negativos (error Tipo II) y 2 falsos positivos (error Tipo I).

array([[25, 2],

[ 4, 30]])

Podemos concluir que utilizando una configuración de 3 capas ocultas con 10 neuronas, tenemos grandes probabilidades de predecir correctamente la presencia—o falta—de enfermedad cardiaca en individuos a través de datos simples de su salud—como mencionamos en la sección [Datos](#Datos).

Conclusiones

Mediante este trabajo, podemos ver la eficacia que tienen los métodos de clasificación basados en redes neuronales para resolver problemáticas importantes como la prevención de muerte de seres queridos. En este caso, una aplicación que utilice una red neuronal de estas características, podría ayudar mucho a centros de salud para priorizar pacientes con alto riesgo de padecer un CVD antes de que sea demasiado tarde. Otro caso de aplicación es una app de celular que a través de datos biométricos de la persona, pueda informarle si se encuentra en riesgo de padecer una enfermedad cardiaca y sugerirle que visite inmediatamente un hospital cercano para ser revisado por un profesional.

1. <https://www.who.int/cardiovascular_diseases/en/> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/ST_depression> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/ST_segment> [↑](#footnote-ref-3)
4. <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html> [↑](#footnote-ref-4)
5. <https://en.wikipedia.org/wiki/Accuracy_and_precision> [↑](#footnote-ref-5)
6. TP = True positive; FP = False positive; TN = True negative; FN = False negative [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Confusion_matrix> [↑](#footnote-ref-7)