# Introducción a las Redes Neuronales Artificiales y aplicaciones en análisis de imágenes

Martín Makler

Desafíos: Carla Bonifazi, João França, Lucas Bignone, Manuel Szewc, Martín Belzunce

ICAS

Instituto de Ciencias Físicas
ICIFI-ECYT_UNSAM-CONICET

CONICET

1983 - 2023
40 AÑOS DE DEMOCRACIA

# ¿Por qué Redes Neuronales Artificiales (ANNs)?

ICAS
icas.unsam.edu.ar

No son los métodos de Aprendizaje Automático más simples

No necesariamente son los mejores ni los más rápidos

Primer sistema verdaderamente de Inteligencia Artificial

Versátiles: no necesitan tener un modelo predefinido, aprende las características

Aún ampliamente utilizadas en las más diversas aplicaciones

Incorporan la mayoría de las características del Aprendizaje Automático

Ejemplo de cómo modelos no lineales simples llevan a comportamientos complejos

Todavía se pueden programar desde cero (+ disponibles en varias bibliotecas)

Puerta de entrada al Aprendizaje Profundo



Hoy: ¿Qué son las Redes Neuronales Artificiales y cómo funcionan?

# Preámbulos

Antes de arrancar con las RNA, vamos a recapitular conceptos importantes:

- Datos y modelos
- Ajuste de funciones / Minimización
- Sobre/sub ajuste, hiperparámetros
- Entrenamiento y predicción
- Conjuntos de entrenamiento, validación y prueba
- Regularización

# Datos y modelos

Normalmente los datos son discretos:

- Ejemplo: pares de números
  pueden visualizarse como un gráfico de dispersión

Frecuentemente los modelos vienen dados por funciones continuas

- Ej.: función continua de una variable ***f(x)***
- Puede representarse en un gráfico ***y = f(x)***

variable

- Ejemplo de modelo: **y = a x + b**

parámetros

- Parámetros determinados a partir de los datos

# Ajustando una función a los datos

Encontrar un modelo (función) que represente (aproximadamente) los datos

Minimizar la distancia de un modelo a los datos

**En este caso**

$$f(a, b; x) = ax + b$$

**Distancia entre la previsión del modelo y el dato:**

$$f(a, b; x_1) - y_1$$

$$= ax_1 + b - y_1$$

***x***

# Ajustando un modelo a los datos

Encontrar un modelo (función) que represente (aproximadamente) los datos

Minimizar la distancia de un modelo a los datos

**Distancia cuadrática entre todos los puntos y la predicción:**

$$(f(a,b;x_1) - y_1)^2 +$$
$$(f(a,b;x_2) - y_2)^2 +$$
$$(f(a,b;x_3) - y_3)^2 +$$
$$\cdots$$
$$= D(a,b)$$

x

# Ajustando un modelo a los datos

Encontrar un modelo (función) que represente (aproximadamente) los datos

Minimizar la distancia de un modelo a los datos

**El modelo que mejor representa los datos (dentro de esa categoría de modelos) es el que minimiza la función** $D(a,b)$

**Una vez que obtenemos *a* y *b*, podemos hacer predicciones para nuevos valores de *x***

**En el caso simple de la función *y = a x + b***
**es muy fácil encontrar *a* y *b* a partir del conjunto de todos los** $x_i$ e $y_i$ **(derivadas, etc.)**
**ver expresión en el notebook**

**Distancia cuadrática entre todos los puntos y la predicción:**

$$
\begin{aligned}
&(f(a,b;x_1)-y_1)^2 + \\
&(f(a,b;x_2)-y_2)^2 + \\
&(f(a,b;x_3)-y_3)^2 + \\
&\cdots \\
&= D(a,b)
\end{aligned}
$$

***x***

# Ejemplo: regresión por mínimos cuadrados

"Algoritmo más sencillo de aprendizaje automático"

- Encontrar un modelo (función) que represente (aproximadamente) los datos
- Minimizar la distancia de un modelo a los datos

$$D^2(a,b;\{y\},\{x\}) = \sum_i \left(f(a,b;x_i) - y_i\right)^2$$

# Regresión por mínimos cuadrados

"Algoritmos más sencillos de aprendizaje automático"

- Minimizar la distancia de un modelo a los datos

$$D^2(a, b; \{y\}, \{x\}) = \sum_i \left(f(a, b; x_i) - y_i\right)^2$$

- Ejemplo: relación lineal y = a x + b
- Cálculo (derivadas) + matrices (inversa, transpuesta): < 10 líneas de código, encuentra a y b

***x***

¡Esto es un aprendizaje automático a partir de datos!

# Recordando: Maximización y minimización

Minimizar (o maximizar) una función (por ejemplo, el beneficio o la distancia cuadrática):

- Puntos extremos
- Pendientes nulas en todas las direcciones (gradiente)
- Muy utilizado en aprendizaje automático
- Encontrar máximos y mínimos en muchas dimensiones puede ser muy complicado
- En este caso de ajuste de función, las dimensiones son los parámetros del modelo

$x$

# ¡Aprendiendo de los datos!

A partir de los datos, obtuvimos un **modelo** que los representa:
el que tiene los parámetros que minimizan la distancia cuadrática promedio a los datos (por eso se llama de "método de mínimos cuadrados")
El modelo **generaliza** una relación y permite hacer **predicciones**

Se puede pensar que el modelo se **entrenó** con los dados (**fiteando**) y ahora puede **predecir** un valor de *y* para cualquier *x*

¡Un algoritmo y un ejemplo simple de aprendizaje automático a partir de datos!

***x***

# ¿Cómo esto se generaliza?

Aquí el modelo era lineal, $y = ax + b$ (por eso se le llama **regresión linear**)
Podemos tener funciones más complejas, como cuadrática, polinómica, etc.

Vector de coeficientes

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

En general tienen más parámetros (= más libertad para ajustar los datos)
¡Hay que minimizar en muchas dimensiones!

# ¿Cómo esto se generaliza?

Aquí el modelo dependía de una única variable, *x*, y queremos predecir un único valor, *y*.

En general la dependencia puede ser en muchas variables

- Ejemplo: valor del inmueble puede depender del área, número de ambientes, región, edad, etc.
- x1, x2, x3, etc. (u otras notaciones)

Podemos querer modelizar/predecir várias cantidades

- Ejemplo: altura y peso en función de la edad (y otras variables)

En vez de valores precisos, muchas veces queremos predecir/entender distribuciones de probabilidad

¡Abordage bayesiano!

***x***

# Aplicaciones más avanzadas

ICAS
icas.unsam.edu.ar

Ejemplo con una función lineal, baja dimensionalidad de los datos (2) y pocos parámetros (2)

Algunos sistemas son altamente no lineales, tienen una alta dimensionalidad y un número extremadamente grande de parámetros.

Por ejemplo, las redes neuronales convolucionales estándar pueden tener 100.000.000 de parámetros libres.

Un poco más complicado....

$x$

Antes de arrancar con las redes neuronales, veamos esos conceptos en la práctica...

Vamos al notebook!

**`RegresionPolinomial.ipynb`**

# Un poco de história

**1943. McCulloch and Pitts.** Un modelo computacional simplificado de cómo las neuronas biológicas podrían trabajar juntas en cerebros de animales para realizar cálculos complejos utilizando lógica proposicional. Esta fue la primera arquitectura de red neuronal artificial.

**1957. Rosenblatt.** Uno de los primeros algoritmos basados en el comportamiento de las neuronas físicas. Consulta "The Organization of Behavior" de Donald Hebb.

Implementado como un programa, pero luego convertido en una máquina.

**1969, Minsky y Papert**. En su libro "Perceptrons" demuestran que el perceptrón no puede aprender la función XOR. Comienza la "edad oscura" de las redes neuronales.

**Años 1980**. Primer retorno de las Redes Neuronales, pero rápidamente superada por otros algoritmos como las Máquinas de Soporte Vectorial (SVM).

**Años 2000**. Segundo retorno, impulsada por el aumento en la capacidad computacional y la disponibilidad de datos, junto con técnicas de entrenamiento eficientes.

# Redes neuronales biologicas

Ramón y Cajal

- Image by Bruce Blaus (Creative Commons 3.0). Reproduced from

# El perceptron

Bloque de construcción de las Redes Neuronales Artificiales (ANN).
Comprender: percibir
"Unidad básica de percepción" en las ANN.
1957: Rosenblatt (primero en software, luego en hardware).

“the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.” [The NYT, 1958]

"El embrión de una computadora electrónica que [la Armada] espera que pueda caminar, hablar, ver, escribir, reproducirse a sí misma y ser consciente de su existencia". [The New York Times, 1958].

# El perceptron

Biological neuron

From Stanford cs231n lecture notes

El perceptron

Imitar el cerebro: recibir señales de entrada producir una señal de salida.

# El perceptron

$$z = \sum_i x_i \times w_i$$

clasificación

salida

Ejemplo de una respuesta binaria (si/no): la función escalón

# Regresión logística

$$z = \sum_i x_i \times w_i$$

clasificación

Decisión
Si/No

Para asignar una "probabilidad"
(y para suavidad/diferenciabilidad):
función sigmoide

# Regresión lineal

$$z = \sum_i x_i \times w_i$$

Nota: para una función de activación lineal, $g(z) = z$, la salida es equivalente a la regresión regresión lineal

Permite combinar diferentes entradas y proporciona una salida

El problema: la combinación lineal. Esencialmente, permite trazar líneas (o hiperplanos) que dividen los datos (o ajustar datos por una recta o hiperplanos)

# Redes neuronales artificiales

- Añadir complejidad: más neuronas (idénticas)
- Cada una con sus propios pesos
- Diferentes salidas para cada una

# Redes neuronales artificiales

- Una capa: hiperplanos que dividen los puntos de datos
- Redes neuronales: imitan al cerebro y conectan las "neuronas"
- Una capa: lineal, más capas: ¡no lineal!

# Redes neuronales artificiales

- Combinar las salidas en una sola neurona para generar la salida final
- ¡Hemos construido una red neuronal!

# Funciones lineales combinadas con activación no lineal

Feed-forward fully-connected

$$y = f\left(w_0^{(2)} + \sum_{j=1}^{M} w_j^{(2)} z_j\right)$$

where

$$z_j = h\left(w_{j0}^{(1)} + \sum_{i=1}^{D} w_{ji}^{(1)} x_i\right)$$

# ¿Como entrenar una RNA?

| Problema | Capa de salida: Tamaño | Capa de salida: Activación | Capa de salida: Error |
|---|---|---|---|
| Regresión | $N$ | $f(x) = x$ | MSE<br>RMSE |
| Clasificación Binaria | $1$ | $f(x)$ = sigmoide | Cross-entropy |
| Clasificación Multi-class | $K$ | $f(x)$ = softmax | Multiclass Cross-entropy |

**Gradient descent**

- El entrenamiento es equivalente a minimizar la función de pérdida.
- Debido al gran número de parámetros en el modelo, a menudo se utiliza regularización en los pesos de las capas ocultas (L2, L1 o ElasticNet).
- Función compleja y no convexa.
- Muchos mínimos locales, puntos de silla, etc.
- El gradiente se puede calcular eficientemente utilizando la retropropagación del error (también conocida como regla de la cadena).
- A menudo se utilizan métodos de gradiente (ver ejemplo de Descenso de Gradiente o SGD).
- La complejidad de la función requiere trucos adicionales (por ejemplo, una tasa de aprendizaje programada) y algoritmos específicos.

# ¿Como entrenar una RNA? Retropropagación

ICAS
icas.unsam.edu.ar

- Entrenar/aprender es obtener los pesos $w_j$ : los que minimizan la función de pérdida
- Necesita las direcciones para mover $w_j$ para disminuir la pérdida (recordar las derivadas, etc...)
- Técnicamente es muy difícil minimizar la función de pérdida: muchos parámetros, altamente no lineal
- Una función de activación suave ("diferenciable") permite el uso directo del descenso del gradiente y otros algoritmos de optimización para el ajuste de los pesos

- Elegante truco matemático: la retropropagación (*backpropagation*, 1986)
- Se parte de la pérdida, se calculan los cambios (derivadas) de los nodos de cada capa "hacia la izquierda" hasta la entrada. Regla de la cadena, producto de matrices
- Reajustar los pesos y calcula de nuevo la pérdida
- Y luego hacia atrás de nuevo, hasta la convergencia.

# ¿Cómo entrenar una RNA?

- Después de cada ciclo (iteración) se actualizan los pesos
- La tasa de aprendizaje determina el "tamaño del paso" (sobrepasar x atascado en un mínimo local)
- Después de muchas iteraciones, el proceso debería converger
- ¡Su red neuronal está finalmente entrenada!

# ¿Cuándo parar? Evitar el sobreajuste

ICAS

- *Holdout*: dividir los datos de entrenamiento en una muestra de entrenamiento y otra de validación (por ejemplo, 80% + 20%)
- Semejante a validación cruzada, pero se hace una sola vez
  Se puede usar **durante** el entrenamiento.
  Ejemplo: EarlyStopping (un tipo de *callback*)
- La pérdida en la muestra de validación debe ser comparable a la del entrenamiento, de lo contrario estamos sobreajustando
- Otras técnicas de regularización: *Lasso*, *Ridge*, *Dropout*
- De ser posible: hacer validación cruzada:
  Repetir la división varias veces, seleccionando aleatoriamente las submuestras: robustez de la RNA y estimación de las barras de error

# Tipos e importancia de las funciones activación

| Capa de salida | | | |
|---|---|---|---|
| **Problema** | **Tamaño** | **Activación** | **Error** |
| Regresión | $N$ | $f(x) = x$ | MSE<br>RMSE |
| Clasificación Binaria | $1$ | $f(x)$ = sigmoide | Cross-entropy |
| Clasificación Multi-class | $K$ | $f(x)$ = softmax | Multiclass Cross-entropy |

$$s(z_i) = \frac{\exp(z_i)}{\sum_{k=1}^{K} \exp(z_j)}$$

Capas internas: ReLU (o SeLU, GeLU)
(problema de los gradientes evanescentes)

# Construyendo una Red Neuronal Artificial en Keras

ICAS
icas.unsam.edu.ar

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
# Multilayer Perceptron model
model = Sequential()
model.add(Dense(input_dim=6, activation="relu", units=4))
model.add(Dense(activation="relu", units=2, kernel_initializer="normal"))
model.add(Dense(activation="sigmoid", units=1, kernel_initializer="normal"))
model.compile(optimizer=SGD(lr=0.01), loss='mean_squared_error', metrics=['accuracy'])
model.summary()
```

RNA con la misma arquitectura que el diagrama de ejemplo mostrado anteriormente

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 4)                 28
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 10
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 3
=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
_________________________________________________________________
```

Además de los pesos $w_j$, cada neurona tiene también un parámetro de sesgo añadido al producto (por lo tanto, un parámetro adicional por neurona)

# Resumen

RNA son versátiles: no importa cuál sea la estructura de los datos o el modelo subyacente

Red neuronal definida por la arquitectura (número de capas, número de neuronas en cada capa, conectividad, etc.), por las funciones de activación y por los pesos $w_j$

Dada una arquitectura, una función pérdida, un optimizador, una regularización, encontrar los pesos mediante retropropagación (definir épocas y tasa)

Posibilidad/necesidad de ajustar los hiperparámetros para mejorar el rendimiento

¡¡Ya estás listo para utilizar tu RNA para predecir salidas!!

Como siempre hay dos etapas:

Entrenamiento: el par entrada-salida es fijo, mientras que los pesos varían, de forma a minimizar la función pérdida

Evaluación: los pesos son fijos, mientras que las entradas varían (y la salida objetivo puede ser desconocida)

# Comentarios finales - parte 1

ICAS
icas.unsam.edu.ar

Para entender las RNA se usan todos los conceptos del aprendizaje automático supervisado

No son el método más simple de ML, ni necesariamente el mejor, pero son muy versátiles

Precio a pagar por la versatilidad (muchos parámetros):

Hambrienta de datos (ok, big data)

Intensiva en términos de computación (ok, GPUs)

Hoy: red neuronal artificial de conexión directa (feedforward), totalmente conectada

Camino hacia el aprendizaje profundo y redes neuronales más especializadas

# Comentarios finales - parte 1

ICAS
icas.unsam.edu.ar

Las redes neuronales son métodos bastante "sencillos", pero potentes

Los avances técnicos han permitido el aprendizaje profundo (por ejemplo, la retropropagación, la activación ReLU, el uso de GPU, etc.)

Hoy en día son triviales de codificar mediante bibliotecas (ej. Keras, PyTorch)

Sin embargo, hay que tener en cuenta su complejidad: arquitectura, función de activación, función de pérdida, tamaño del lote/velocidad de aprendizaje/épocas, optimizadores, etc.

Hoy vamos a ver algunos ejemplos de la construcción y el entrenamiento de una RNA y seguiremos con más detalles en la próxima clase

# Para divertirse y aprender sobre redes neuronales sin código

ICAS
icas.unsam.edu.ar

http://playground.tensorflow.org/

http://ai-friendly.com/

# Para divertirse y aprender sobre redes neuronales sin código

ICAS
icas.unsam.edu.ar

http://playground.tensorflow.org/

# Para divertirse y aprender sobre redes neuronales sin código

ICAS

du.ar

**Upload an excel file to teach "something" to the Neural Network, and then upload another excel file to take profit of what the Machine has learned!**

Let's try! Customize Network

[Help] [About]

por Cesar Miquel y Ezequiel Alvarez (+G. Mazzei)

Vamos al notebook!

**`Notebook_RedesNeuronales.ipynb`**

# Métricas de clasificación

## Matriz de confusión

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix}$$

elementos relevantes

## Métricas

$$\text{precision} = \frac{TP}{TP + FP} =$$

$$\text{recall} = \frac{TP}{TP + FN} =$$

## Taza de éxitos

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$