EST-25134: Aprendizaje Estadístico

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Redes Neuronales.
Objetivo: Que veremos.
Lectura recomendada: Referencia. *Disclaimer*: Todas las figuras han sido tomadas de [1].

1. INTRODUCCIÓN

- La primera publicación relacionada es la de Rosenblatt [2].
- Las redes neuronales se volvieron populares en los 80s (con impacto limitado).
- A partir de ahí, se vio un auge en otros modelos predictivos.
- A partir de 2010, surge *Deep Learning* con resultados impresionantes.
- Mejora en poder de cómputo, *software* y datos.

	1970	1980	1990	2000	2010	2020
Data (samples)	10^2 (e.g. iris)	10^3	10^4 OCR	10^{7-8} web	10^{10} advertising	10^{12} social nets
RAM	1kB	100kB	10MB	100MB	1GB	100GB
CPU	100kF (8080)	1MF (80186)	10MF (80486)	1GF (Intel Core)	100GF NVIDIA	>1PF (8xP3 Volta)

Figura 1. *Imagen tomada de [4].*

- El avance y adopción se debe a diversos investigadores en varios ámbitos de lo que rodea *Deep Learning*.
- Por ejemplo (time-line de eventos importantes):
  - Rumelhart et al. [3]: Redescubren la regla de la cadena (`backpropagation`).
  - Yann LeCun, **Corinna Cortes** y Christopher Burges hacen pública la base de datos de `MNIST` (1998).
  - Un equipo de investigadores liderado por **Fei-Fei Li** publica la base de `imageNet` para concurso (2009).

  - Un equipo de investigadores en Google Deepmind logra vencer a jugadores profesionales en Go (2016).
- Otros perfiles de gente impresionante haciendo investigación en el área se puede encontrar aquí.

2. PRECURSOR: PERCEPTRÓN

- Objetivo: resolver el problema de clasificación binaria por medio de separaciones lineales.
- Es decir, poder encontrar una ω tal que

$$\langle \omega, x \rangle \geq 0, \qquad \text{si } y = 1\,, \tag{1}$$
$$\langle \omega, x \rangle < 0, \qquad \text{si } y = -1\,. \tag{2}$$

- Por lo tanto, lo que queremos es un predictor de la forma

$$\hat{y} = \mathsf{signo}(\langle \omega, x \rangle)\,. \tag{3}$$

2.1. Algoritmo:

- Empezamos con $\omega^{(1)} = 0$.
- Para cada iteración $t = 1, \ldots, T$:
  - Buscamos un elemento mal clasificado:

$$y_i \cdot \langle \omega^{(t)}, x_i \rangle < 0\,, \tag{4}$$

    dentro de observaciones.
  - Actualizamos por medio de:

$$\omega^{(t+1)} = \omega^{(t)} + y_i \cdot x_i\,. \tag{5}$$

- Nos detenemos cuando todas las observaciones están bien clasificadas.

2.2. Observaciones

- El perceptrón funciona cuando las clases son separables.
- El perceptrón no convergerá cuando las clases no son separables.
- La pérdida asociada a este algoritmo considera términos individuales

$$\text{máx}[0, -y\langle \omega, x \rangle]\,. \tag{6}$$

- La idea que utilizaremos: el perceptrón emite una señal si $\langle \omega, x \rangle \geq 0$,
  ¿qué tal que utilizamos un conjunto de perceptrones y los combinamos linealmente?

3. RED NEURONAL DE UNA CAPA

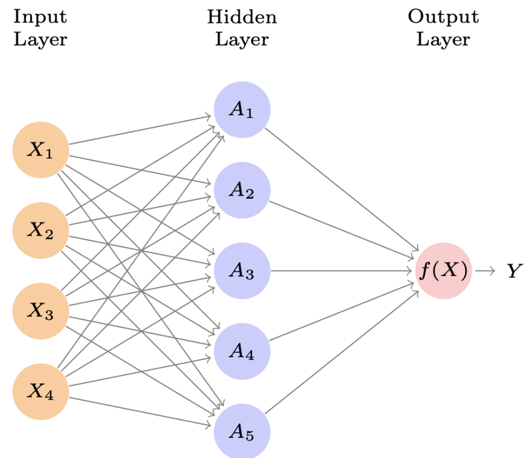
Consideramos el modelo predictivo de la forma

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X)\,, \tag{7}$$

donde los términos h_k son `transformaciones no-lineales` de `combinaciones lineales de los atributos`. Es decir,

$$h_k(X) = g\left(\omega_{k0} + \sum_{j=1}^{p} \omega_{kj} X_j\right)\,, \tag{8}$$

donde $g(\cdot)$ es una transformación no-lineal de $\mathbb{R}$ a $\mathbb{R}$.



3.1. Detalles

- En la figura anterior tenemos que $A_k = h_k(X) = g\left(\omega_0 + \sum_{j=1}^{p} \omega_{kj} X_j\right)$.
- La función $g(\cdot)$ se denomina `función de activación`.
- Las opciones mas populares son: `ReLU` o `sigmoide`.
- Si no utilizamos funciones de activación no-lineales, entonces el modelo seguiría siendo lineal.
- La salida de las funciones de activación son interpretadas como atributos .
- El modelo se entrena (en regresión) minimizando

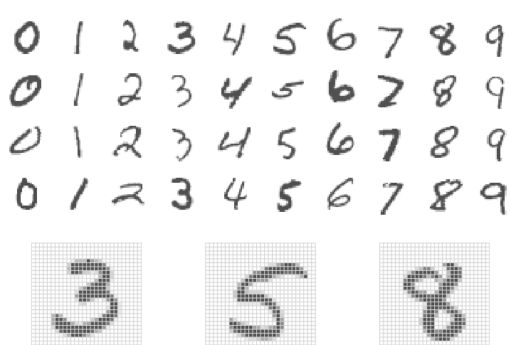
$$\sum_{i=1}^{n} (y_i - f(x_i))^2 \,. \tag{9}$$

- La solución aprende representaciones de los atributos que pueden servir para predecir.

3.2. Ejemplo: clasificación multi-clase (`MNIST`)

Tenemos imágenes de 28×28 pixeles en escala de grises. Tenemos $60K$ datos de entrenamiento y $10K$ datos de validación. Podemos pensar que cada imagen es un vector de 784 dimensiones. Las etiquetas son los dígitos del 0 al 9.

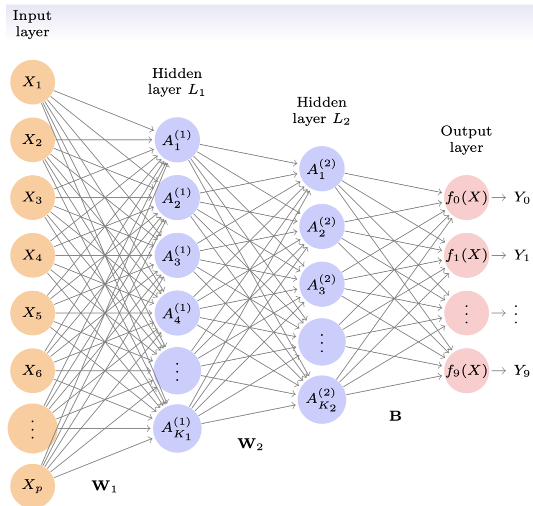
Objetivo: Predecir la clase de la imagen basada en los valores de los pixeles.



3.3. El modelo

Se utiliza una red neuronal de dos capas. La estructura (arquitectura) es 256 unidades en la primera capa, 128 unidades en la capa intermedia y 10 unidades de salida.

3.3.1. ¿Cuántos parámetros tiene este modelo?



3.4. La capa de salida

- Denotemos por

$$Z_m = \beta_{m0} + \sum_{\ell=1}^{K_2} \beta_{m\ell} A_\ell^{(2)}\,, \tag{10}$$

las m combinaciones lineales de las unidades que salen de la segunda capa.
- Denotamos por m es el número de unidades en la capa de salida.
- Para obtener *probabilidades* usamos la función `softmax` como función de activación en la última capa

$$f_m(X) = \mathbb{P}(Y = m|X) = \frac{\exp(Z_m)}{\sum_{\ell=0}^{9} \exp(Z_\ell)}\,, \tag{11}$$

donde entrenamos el modelo minimizando

$$-\sum_{i=1}^{n} \sum_{m=0}^{9} y_{im} \log(f_m(x_i))\,, \tag{12}$$

la cual llamamos `entropía cruzada`.

- y_{im} tomará el valor de 1 en la clase que a la que pertenezca la observación i ésima. Todos los demás valores son 0 (`one-hot encoding`).

3.5. Detalles

- La pérdida de `entropía relativa` corresponde a un modelo multinomial de K clases:

$$\mathbb{P}(y|x) = \prod_{k=1}^{K} p_k(x)^{y_k}\,, \tag{13}$$

- Considerando la función de `softmax` entonces la función de pérdida (individual) queda

$$\ell(y, \hat{y}) = -\left[\sum_{k=1}^{K} y_k \log\left(\frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}\right)\right]. \tag{14}$$

- La cual se puede simplificar

$$\ell(y,\hat{y}) = -\sum_{k=1}^{K} y_k z_k + \log\left(\sum_{k=1}^{K}\exp(z_k)\right) . \tag{15}$$

- Lo cual es muy útil para métodos iterativos de optimización

$$\frac{\partial \ell}{\partial z_j} = \mathsf{softmax}(z_j) - y_j . \tag{16}$$

3.6. Regularización

- Con tantos parámetros en los modelos resulta indispensable *regularizar* nuestro problema de entrenamiento.
- Consideremos el problema de clasificar imágenes de perros y gatos.
- Las imágenes son tomadas con nuestras cámaras (12Mp) lo cual se traduce en 12×10^6 píxeles.
- Un modelo de una capa con mil unidades tiene entonces (apróx.) 36×10^9 parámetros.
- Según una búsqueda en Google (datos de 2019), tenemos una población de 471M perros y 373M gatos.
  - Esto es (apróx) 0.844×10^9 imágenes.
- Necesitaríamos $36/.844 \approx 42.65$ más datos para tener una relación 1 a 1 de parámetros con datos.

Los métodos usuales de regularización son (mas adelante veremos detalles de esto):

1. Regularización en coeficientes matrices W_k.
2. Regularización *dropout*.

- Resultados en MNIST son:

Method	Test Error
Neural Network + Ridge Regularization	2.3%
Neural Network + Dropout Regularization	1.8%
Multinomial Logistic Regression	7.2%
Linear Discriminant Analysis	12.7%

FIGURA 2. *Resultados de generalización obtenidos por distintos modelos en el conjunto de datos de* `MNIST`*, fuente: [1].*

- A la fecha, los mejores resultados reportan un error de generalización de menos del 0.5 %
- El error de personas en este conjunto de datos es de 0.2 %,

4. MODELOS CONVOLUCIONALES

- Historia de éxito para problemas de visión por computadora.

5. MODELOS RECURRENTES

6. CASOS DE USO

7. AJUSTE Y REGULARIZACIÓN

8. SOFTWARE

REFERENCIAS

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer US, New York, NY, 2021. 1, 5

[2] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471. . 1

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. ISSN 1476-4687. . 1

[4] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. In print, 2021. 1