

Machine learning

Una introducción muy improvisada

Manuel Carlevaro

Grupo de Materiales Granulares - Dpto. Ing. Mecánica, UTN FRLP.

Taller de aprendizaje GMG - ML
11 de agosto, 2020

- Comprender la metodología de ML
- Aprender haciendo (?)
- Aprendizaje colaborativo
- Incorporar la técnica en nuestros trabajos

- Comprender la metodología de ML
- Aprender haciendo (?)
- Aprendizaje colaborativo
- Incorporar la técnica en nuestros trabajos

- Comprender la metodología de ML
- Aprender haciendo (?)
- Aprendizaje colaborativo
- Incorporar la técnica en nuestros trabajos

- Comprender la metodología de ML
- Aprender haciendo (?)
- Aprendizaje colaborativo
- Incorporar la técnica en nuestros trabajos

- Comprender la metodología de ML
- Aprender haciendo (?)
- Aprendizaje colaborativo
- Incorporar la técnica en nuestros trabajos



Algunos recursos:

Physics Reports 810 (2019) 1–124


Contents lists available at ScienceDirect

Physics Reports

journal homepage: www.elsevier.com/locate/physrep

A high-bias, low-variance introduction to Machine Learning for physicists



Pankaj Mehta^a, Marin Bukov^{b,*}, Ching-Hao Wang^a, Alexandre G.R. Day^a, Clint Richardson^a, Charles K. Fisher^c, David J. Schwab^d

^a Department of Physics, Boston University, Boston, MA 02215, USA
^b Department of Physics, University of California, Berkeley, CA 94720, USA
^c Unilearn.AI, San Francisco, CA 94108, USA
^d Initiative for the Theoretical Sciences, The Graduate Center, City University of New York, 365 Fifth Ave., New York, NY 10016, USA

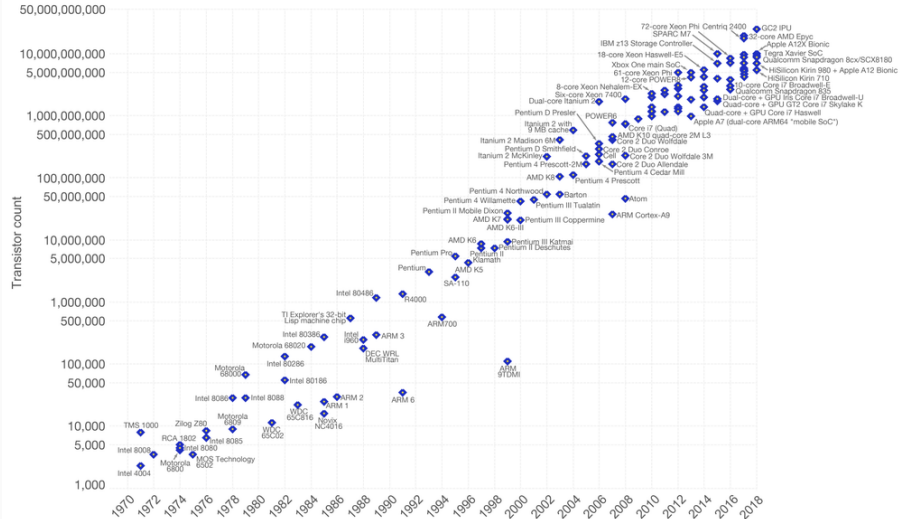
Machine Learning for Physicists

NEURAL NETWORKS AND THEIR APPLICATIONS (SLIDES AND VIDEOS FOR THE LECTURES BY FLORIAN MARQUARDT)



Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

The data visualization is available at [OurWorldinData.org](https://ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

- Aprendizaje supervisado:
 - Aprendizaje a partir de datos etiquetados
 - ▶ Clasificación y regresión
- Aprendizaje no supervisado
 - Búsqueda de patrones y estructuras en datos no etiquetados
 - ▶ Clustering, reducción dimensional, modelado generativo
- Aprendizaje reforzado
 - Aprendizaje a través de la interacción con el entorno
 - ▶ Entrenamiento de robots

- Aprendizaje supervisado:
 - Aprendizaje a partir de datos etiquetados
 - ▶ Clasificación y regresión
- Aprendizaje no supervisado
 - Búsqueda de patrones y estructuras en datos no etiquetados
 - ▶ Clustering, reducción dimensional, modelado generativo
- Aprendizaje reforzado
 - Aprendizaje a través de la interacción con el entorno
 - ▶ Entrenamiento de robots

- Aprendizaje supervisado:
 - Aprendizaje a partir de datos etiquetados
 - ▶ Clasificación y regresión
- Aprendizaje no supervisado
 - Búsqueda de patrones y estructuras en datos no etiquetados
 - ▶ Clustering, reducción dimensional, modelado generativo
- Aprendizaje reforzado
 - Aprendizaje a través de la interacción con el entorno
 - ▶ Entrenamiento de robots

- Dataset $\mathcal{D}(\mathbf{X}, \mathbf{y})$
 - \mathbf{X} : matriz de variables independientes
 - \mathbf{y} : vector de variables dependientes
- Modelo $f(\mathbf{x}, \boldsymbol{\theta})$
 - $f : \mathbf{x} \mapsto \mathbf{y}$
 - f predice una salida a partir de un vector de variables de entrada y parámetros $\boldsymbol{\theta}$
- Función costo $\mathcal{C}(\mathbf{y}, f(\mathbf{X}, \boldsymbol{\theta}))$
 - Cuantifica cuán bien el modelo predice las observaciones \mathbf{y}

- Particionado **aleatorio** de datos en dos conjuntos mutuamente exclusivos:
 - $\mathcal{D}_{\text{train}}$ ($\sim 90\%$ de los datos)
 - $\mathcal{D}_{\text{test}}$ ($\sim 10\%$ de los datos)

- Minimización de la función costo:

$$\hat{\theta} \in \Theta \text{ tal que } \mathcal{C}(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}, \hat{\theta})) \leq \mathcal{C}(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}, \theta)), \forall \theta \in \Theta$$

- Evaluación del desempeño del modelo calculando $\mathcal{C}(\mathbf{y}_{\text{test}}, f(\mathbf{X}_{\text{test}}, \hat{\theta}))$

- Particionado **aleatorio** de datos en dos conjuntos mutuamente exclusivos:

- $\mathcal{D}_{\text{train}}$ ($\sim 90\%$ de los datos)
- $\mathcal{D}_{\text{test}}$ ($\sim 10\%$ de los datos)

- Minimización de la función costo:

$$\hat{\theta} \in \Theta \text{ tal que } \mathcal{C}(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}, \hat{\theta})) \leq \mathcal{C}(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}, \theta)), \forall \theta \in \Theta$$

- Evaluación del desempeño del modelo calculando $\mathcal{C}(\mathbf{y}_{\text{test}}, f(\mathbf{X}_{\text{test}}, \hat{\theta}))$

Errores *in* y *out*

Error in sample: $E_{\text{in}} = \mathcal{C}(\mathbf{y}_{\text{train}}, f(\mathbf{X}_{\text{train}}, \hat{\theta}))$

Error out-of-sample: $E_{\text{out}} = \mathcal{C}(\mathbf{y}_{\text{test}}, f(\mathbf{X}_{\text{test}}, \hat{\theta}))$

$$E_{\text{out}} \geq E_{\text{in}}$$

Proceso probabilístico $x_i \mapsto y_i$:

$$y_i = f(x_i) + \eta_i$$

$$\langle \eta_i \rangle = 0$$

$$\langle \eta_i \eta_j \rangle = \delta_{ij} \sigma^2$$

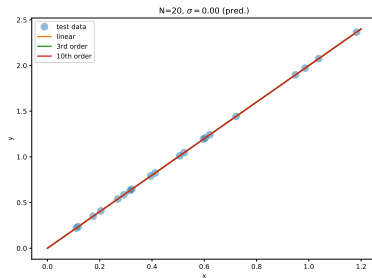
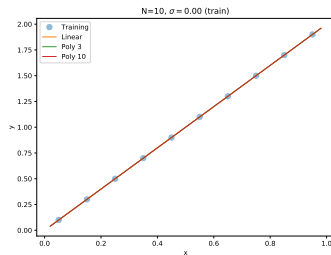
Clase de modelo: $f_\alpha(x; \theta_\alpha)$

- $f_1(x; \theta_1)$ (2 parámetros)
- $f_3(x; \theta_3)$ (4 parámetros)
- $f_{10}(x; \theta_{10})$ (11 parámetros)

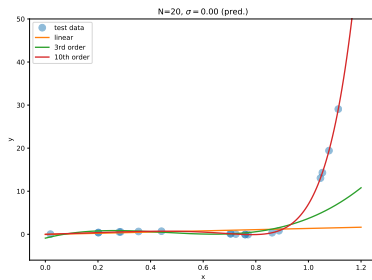
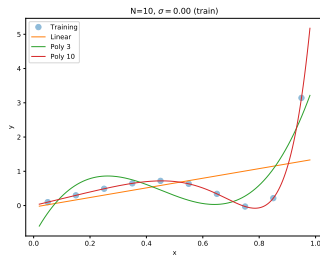
[Ver notebook de Jupyter](#)

$$\sigma = 0$$

$$f(x) = 2x$$

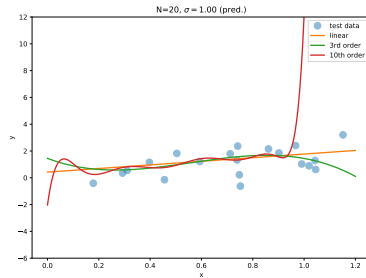
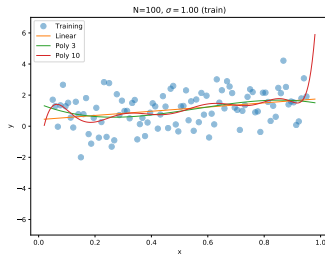


$$f(x) = 2x - 10x^5 + 15x^{10}$$

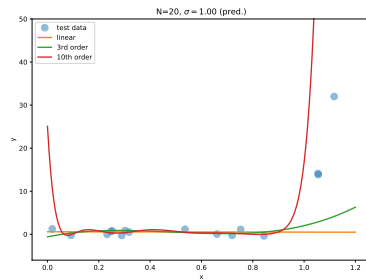
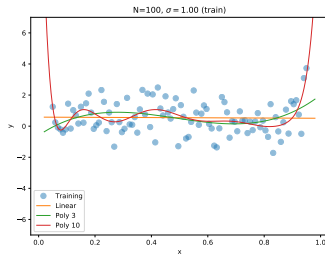


$$\sigma = 1$$

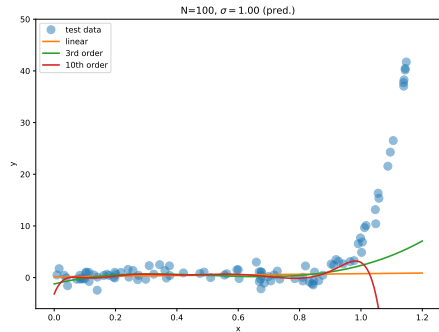
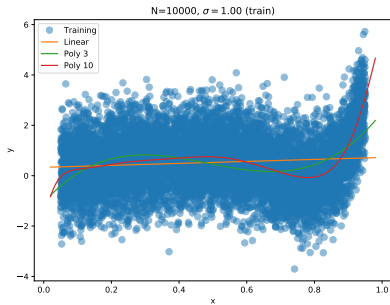
$$f(x) = 2x$$

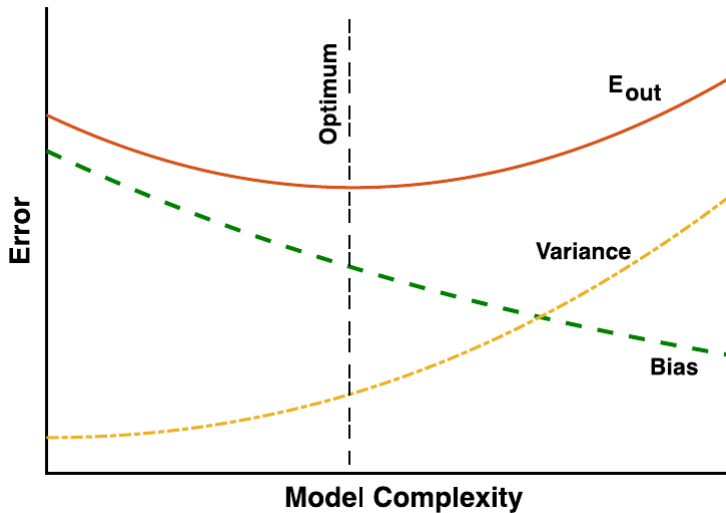


$$f(x) = 2x - 10x^5 + 15x^{10}$$



Compensación sesgo-varianza





- Ajustar no es predecir. Ajustar bien datos existentes es fundamentalmente diferente de hacer predicciones sobre datos nuevos.
- Usar un modelo complejo puede resultar en overfitting. Aumentar la complejidad de un modelo (es decir, el número de parámetros de ajuste) producirá usualmente mejores resultados sobre los datos de entrenamiento. Sin embargo, cuando hay pocos datos de entrenamiento y los datos son ruidosos, esto resulta en overfitting y puede degradar significativamente el poder predictivo del modelo.
- Para datos complejos y conjuntos chicos de entrenamiento, los modelos simples pueden ser mejores para predecir que modelos complejos debido al tradeoff bias-varianza. Se necesita menos información para entrenar un modelo simple que uno complejo. Por lo tanto, aunque se garantiza que el modelo correcto tendrá un mejor rendimiento predictivo para una cantidad infinita de datos de entrenamiento (menos sesgo), los errores de entrenamiento que se derivan del muestreo de tamaño finito (varianza) pueden hacer que modelos más simples superen al modelo más complejo cuando el muestreo está limitado.
- Es muy difícil generalizar más allá de las situaciones encontradas en el conjunto de datos de entrenamiento.

Redes neuronales: motivación¹

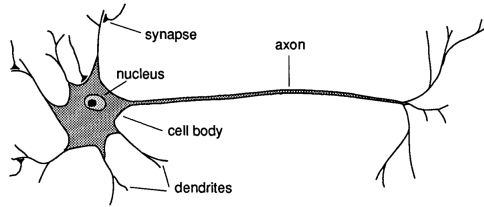


FIGURE 1.1 Schematic drawing of a typical neuron.

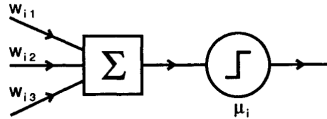
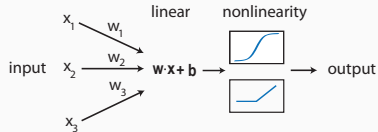


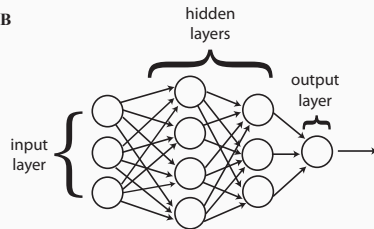
FIGURE 1.2 Schematic diagram of a McCulloch-Pitts neuron. The unit fires if the weighted sum $\sum_j w_{ij} n_j$ of the inputs reaches or exceeds the threshold μ_i .

¹J.A. Hertz, A.S. Krogh y R.G. Palmer. Introduction To The Theory Of Neural Computation. Westview Press, CRC. 1991.

A



B



Neurona i : $\mathbf{x} = (x_1, x_2, \dots, x_d) \mapsto a_i(\mathbf{x})$

Pesos de cada neurona:

$$\mathbf{w}^{(i)} = (w_1^{(i)}, w_2^{(i)}, \dots, w_d^{(i)})$$

Bias: $b^{(i)}$

Entrada por neurona:

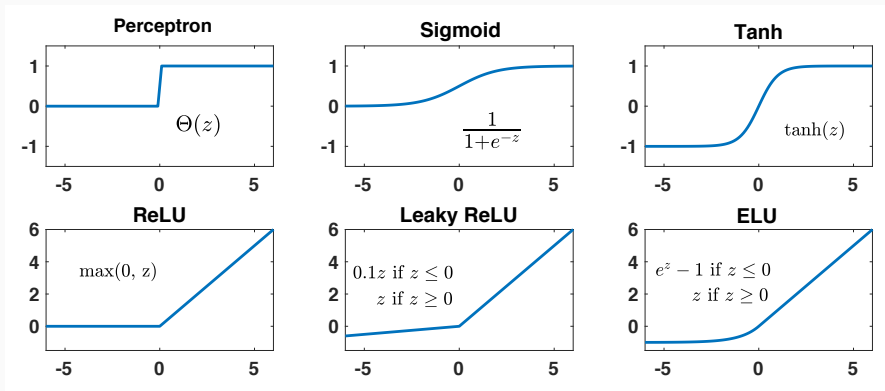
$$z^{(i)} = \mathbf{w}^{(i)} \cdot \mathbf{x} + b^{(i)} = \mathbf{x}^T \cdot \mathbf{w}^{(i)}$$

donde:

$$\mathbf{x} = (1, \mathbf{x})$$

$$\mathbf{w} = (b^{(i)}, \mathbf{w}^{(i)})$$

Activación: $a_i(\mathbf{x}) = \sigma_i(z^{(i)})$



Dado un punto dato: (\mathbf{x}_i, y_i) , $\mathbf{x}_i \in \mathbb{R}^{d+1}$, la red predice $\hat{y}_i(\mathbf{w})$

Datos continuos:

- Error cuadrático medio

$$E(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i(\mathbf{w}))^2$$

- Norma L_1

$$E(\mathbf{w}) = \frac{1}{n} \sum_i |y_i - \hat{y}_i(\mathbf{w})|$$

Datos categóricos:

- $y \in \{0, 1, \dots, M-1\}$

- Para cada punto dato i , definimos y_{im} :

$$y_{im} = \begin{cases} 1, & \text{if } y_i = m \\ 0, & \text{otherwise.} \end{cases}$$

- Probabilidad de asignar una categoría m :

$$\hat{y}_{im}(\mathbf{w}) = p(y_i = m | \mathbf{x}_i; \mathbf{w})$$

- Categorical cross-entropy:

$$E(\mathbf{w}) = - \sum_{i=1}^n \sum_{m=0}^{M-1} y_{im} \log \hat{y}_{im}(\mathbf{w}) \\ + (1 - y_{im}) \log [1 - \hat{y}_{im}(\mathbf{w})]$$

$$a_j^l = \sigma \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right) = \sigma(z_j^l) \quad (1)$$

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l. \quad (2)$$

Error Δ_j^L de la neurona j en la capa L como el cambio en la función costo respecto de z_j^L :

$$\Delta_j^L = \frac{\partial E}{\partial z_j^L}$$

Análogamente, error de la neurona j en la capa l , Δ_j^l , como el cambio de la función costo respecto de z_j^l :

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial a_j^l} \sigma'(z_j^l), \quad (I)$$

Notar que Δ_j^l puede interpretarse como la derivada parcial de la función costo respecto del bias b_j^l :

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial b_j^l} \frac{\partial b_j^l}{\partial z_j^l} = \frac{\partial E}{\partial b_j^l}, \quad (II)$$

Dado que el error depende de las neuronas en la capa l solo a través de la activación de las neuronas de la capa siguiente $l+1$, podemos usar la regla de la cadena:

$$\begin{aligned} \Delta_j^l &= \frac{\partial E}{\partial z_j^l} = \sum_k \frac{\partial E}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \Delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \left(\sum_k \Delta_k^{l+1} w_{kj}^{l+1} \right) \sigma'(z_j^l). \end{aligned} \quad (III)$$

La ecuación final se deriva diferenciando la función costo respecto del peso w_{jk}^l :

$$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial E}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \Delta_j^l a_k^{l-1} \quad (\text{IV})$$

Algoritmo:

1. **Activación en la capa input:** calcular las activaciones a_j^1 de todas las neuronas en la capa input.
2. **Feedforward:** empezando en la primera capa, usar la arquitectura feed-forward a través de la ec. (1) para calcular z^l and a^l en cada capa.
3. **Error en la última capa:** calcular el error de la última capa usando la ec. (I). Esto requiere conocer la expresión para la derivada de la función costo $E(\mathbf{w}) = E(\mathbf{a}^L)$ y de la función de activación $\sigma(z)$.
4. **“Backpropagate” el error:** usar la ec. (III) para propagar el error hacia atrás y calcular Δ_j^l para todas las capas.
5. **Calcular el gradiente:** usar las ecs. (II) y (IV) para calcular $\frac{\partial E}{\partial b_j^l}$ y $\frac{\partial E}{\partial w_{jk}^l}$.

A high-bias, low-variance introduction to Machine Learning for physicists

Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson, Charles K. Fisher, David J. Schwab.

[arXiv:1803.08823v3](https://arxiv.org/abs/1803.08823v3) [physics.comp-ph]

29 de mayo, 2019

En el próximo episodio:
el “¡Hola mundo!” de las redes neuronales,
por Ramiro Irastorza

✉ manuel.carlevaro@gmail.com

🐧 • Lua^ATeX

