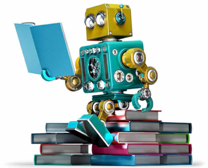
Clasificación Lineal y Evaluación del Desempeño

Inteligencia Artificial



Marco Teran

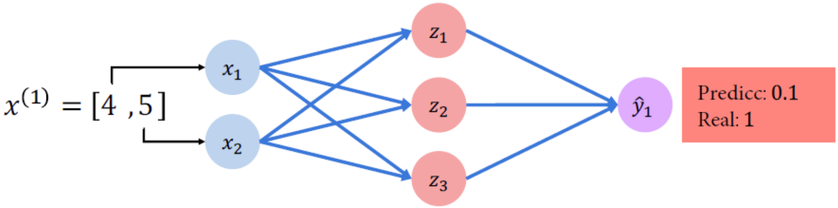
2021 - Bogotá

Contenido

Cuantificación de las pérdidas

Cuantificación de las pérdidas

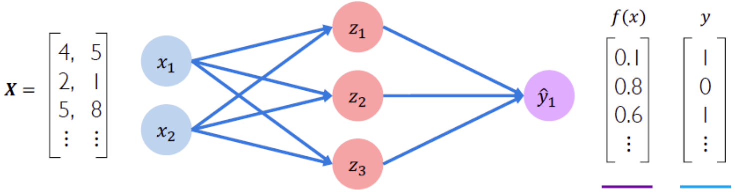
La pérdida (loss) de nuestra red mide el costo incurrido por las predicciones incorrectas



$$\mathcal{L}\left(\underbrace{f\left(x^{(i)};\boldsymbol{W}\right)}_{\text{Predicción}}, \underbrace{y^{(i)}}_{\text{Real}}\right)$$

Pérdidas empíricas

La pérdida empírica mide la pérdida total en todo nuestro conjunto de datos



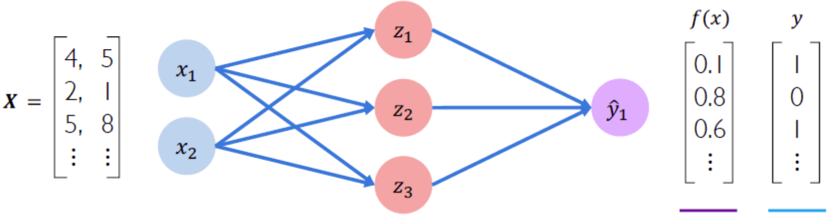
$$J(W) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\Big(\underbrace{f\big(x^{(i)}; W\big)}_{\text{Predicción}}, \underbrace{y^{(i)}}_{\text{Real}}\Big)$$

También conocida:

- Función objetivo
- Función de costo
- Riesgo empírico

Entropía cruzada binaria

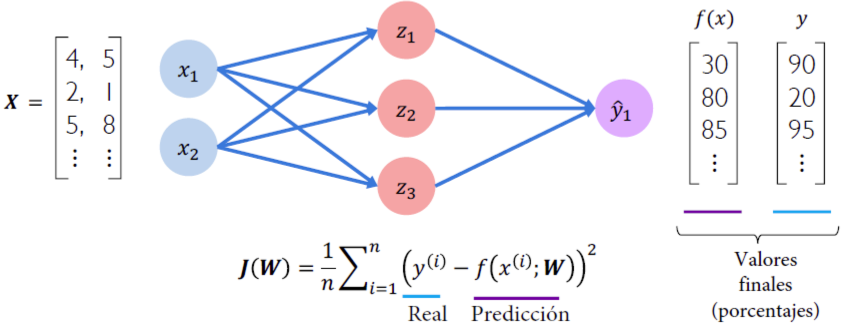
Binary Cross Entropy Loss La pérdida de entropía cruzada puede utilizarse con modelos que arrojan una probabilidad entre 0 y 1



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Real}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicción}}\Big) + (1-\underbrace{y^{(i)}}_{\text{Real}})\log\Big(1-\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicción}}\Big)$$

Perdida del Error cuadrático medio

Mean Squared Error Loss La pérdida media de error al cuadrado se puede usar con modelos de regresión que producen números reales continuos



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{y^{(i)}}_{\text{Real}} - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicción}}\Big)^2$$

Entrenamiento

Optimización de la perdida

Queremos encontrar los pesos de la red que **logren la menor pérdida**

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f(x^{(i)}; \mathbf{W}), y^{(i)}\right)$$

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\, J(\mathbf{W})$$

Optimización de la perdida

Queremos encontrar los pesos de la red que **logren la menor pérdida**

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f(x^{(i)}; \mathbf{W}), y^{(i)}\right)$$

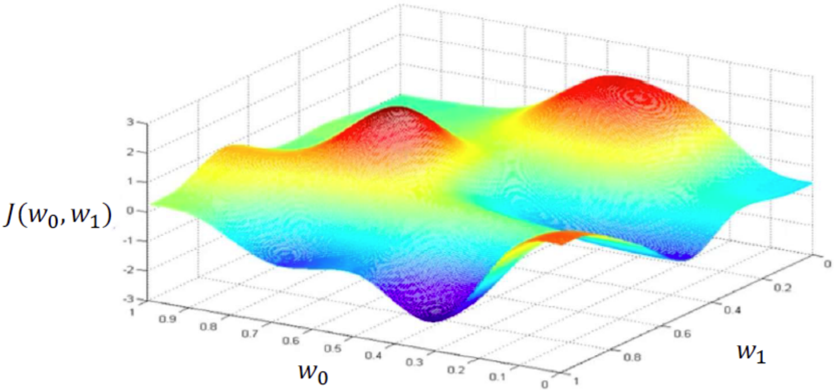
$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{W})$$

Recuerda:

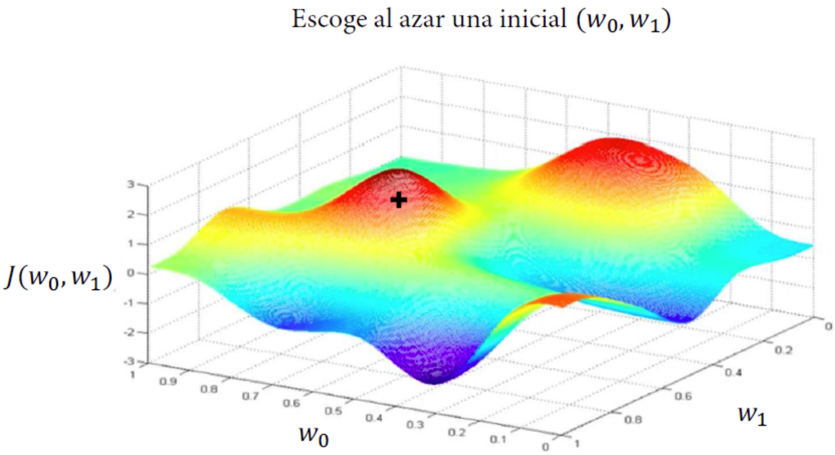
$$\mathbf{W} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \ldots\}$$

Optimización de la perdida

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\, J(\mathbf{W})$$

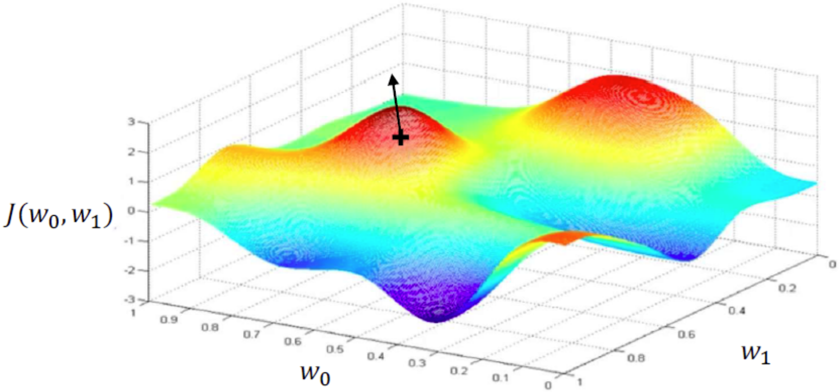


Optimización de la perdida



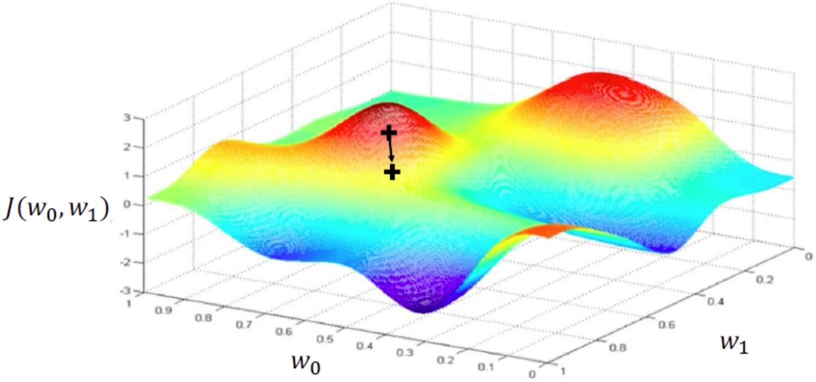
Optimización de la perdida

Calculo del gradiente, $\dfrac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$



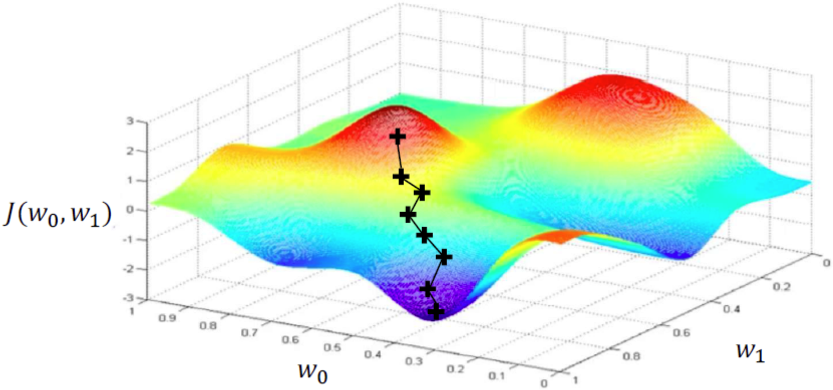
Optimización de la perdida

De un pequeño paso en dirección opuesta al gradiente



Gradiente descendente

Repita hasta la convergencia



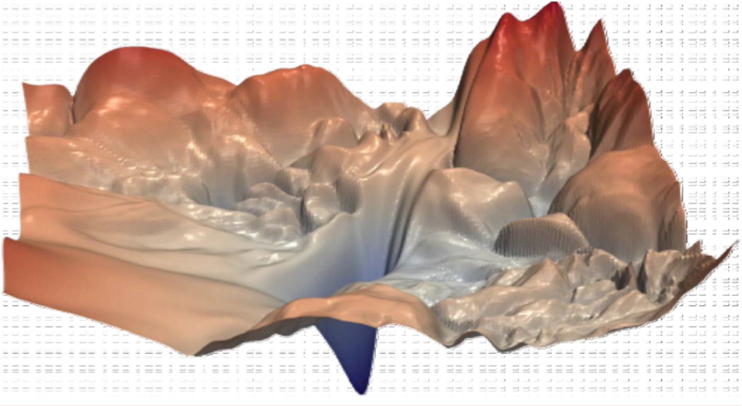
Gradiente descendente

Algoritmo:

- Iniciar los pesos al azar $\sim \mathcal{N}(0, \sigma^2)$
- Bucle hasta la convergencia:
  - Calcular el gradiente $\dfrac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
  - Actualizar los pesos $\mathbf{W} \leftarrow \mathbf{W} - \eta \dfrac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Devuelve los pesos

Optimización

El entrenamiento es complejo



"Visualizando el paisaje de pérdida de las redes neuronales". Diciembre de 2017.

Las funciones de pérdida pueden ser difíciles de optimizar

Recuerde: Optimización a través del descenso del gradiente

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

Las funciones de pérdida pueden ser difíciles de optimizar

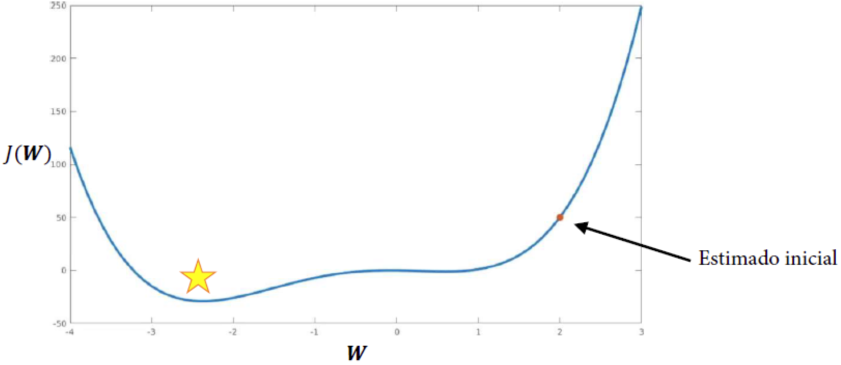
Recuerde: Optimización a través del descenso del gradiente

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$$

¿Cómo podemos establecer la tasa de aprendizaje? *(learning rate)*

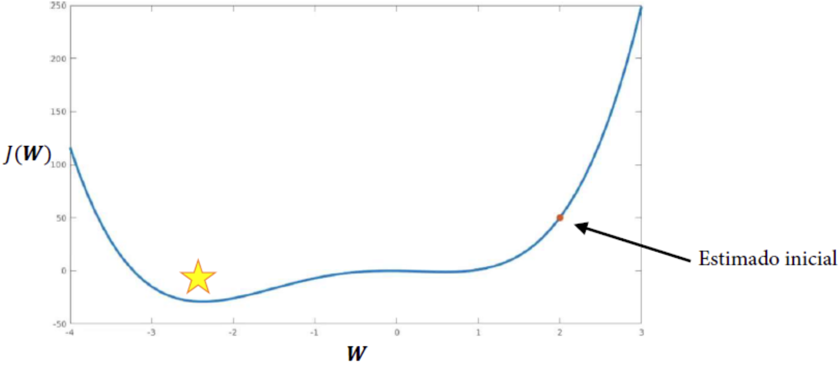
Ajuste de la tasa de aprendizaje

Recuerde: Una pequeña tasa de aprendizaje converge lentamente y se atasca en falsos mínimos locales



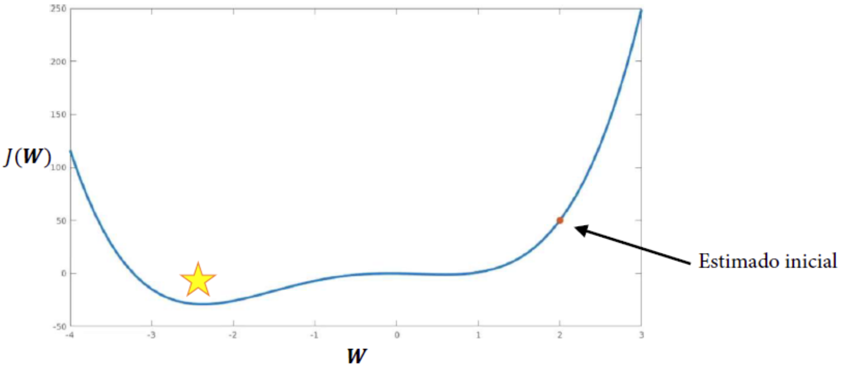
Ajuste de la tasa de aprendizaje

Recuerde: Las grandes tasas de aprendizaje se sobrepasan, se vuelven inestables y divergen



Ajuste de la tasa de aprendizaje

Recuerde: Las tasas de aprendizaje estables convergen sin problemas y evitan los mínimos locales



¿Cómo se puede hacer frente a esto?

- **Idea 1:** Intentar muchas tasas de aprendizaje diferentes y ver cuál funciona ”bien”.
- **Idea 2:** ¡Haz algo más inteligente! Diseñar una tasa de aprendizaje adaptativo que se ”adapte” al paisaje

Tasas de aprendizaje adaptativas

- Las tasas de aprendizaje ya no son fijas
- Pueden hacerse más grandes o más pequeñas dependiendo de:
    - de cuán grande sea el gradiente
    - lo rápido que se está aprendiendo
    - tamaño de pesos particulares
    - etc...

Algoritmos de tasas de aprendizaje adaptativas

- Momentum
- Adagrad
- Adadelta
- RMSProp

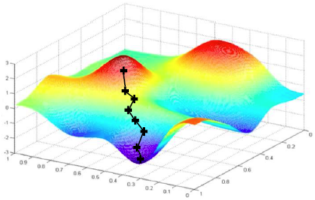
Detaller adicionales: http://ruder.io/optimizing-gradient-descent/

Mini-lotes

Gradiente descendente

Algoritmo:

- Iniciar los pesos al azar $\sim \mathcal{N}(0, \sigma^2)$
- Bucle hasta la convergencia:
  - Calcular el gradiente $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
  - Actualizar los pesos $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Devuelve los pesos



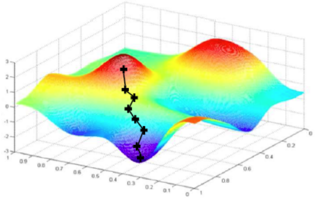
Difícil de calcular $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$

Gradiente descendente estocástico

Algoritmo:

- Iniciar los pesos al azar $\sim \mathcal{N}(0, \sigma^2)$
- Bucle hasta la convergencia:
  - Tomar un solo punto i
  - Calcular el gradiente $\frac{\partial J_i(\mathbf{W})}{\partial \mathbf{W}}$
  - Actualizar los pesos $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Devuelve los pesos

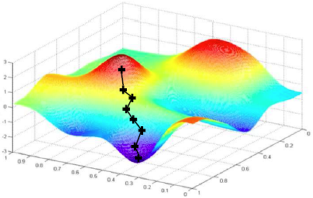
Fácil de calcular $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$ pero muy ruidoso (estocástico)



Gradiente descendente estocástico

Algoritmo:

- Iniciar los pesos al azar $\sim \mathcal{N}(0, \sigma^2)$
- Bucle hasta la convergencia:
  - Tomar un lote de puntos B
  - Calcular el gradiente $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial J_k(\mathbf{W})}{\partial \mathbf{W}}$
  - Actualizar los pesos $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
- Devuelve los pesos



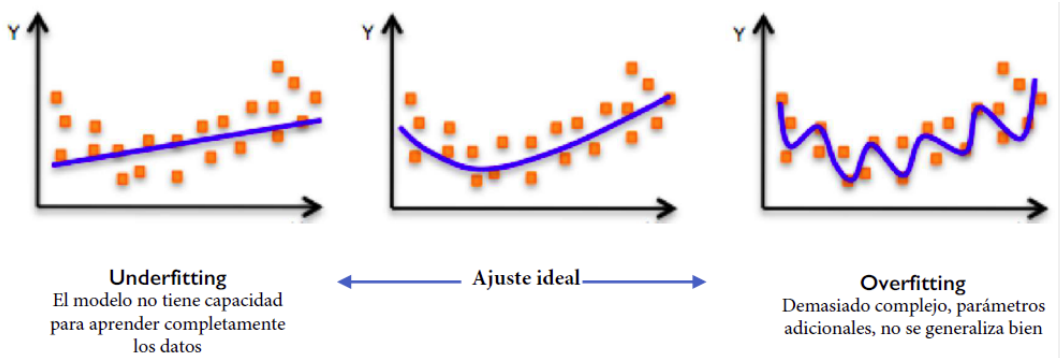
Rápido de calcular y una estimación mucho mejor del verdadero gradiente

Mini-batches durante el entrenamiento

- Estimación más precisa del gradiente:
  - Una convergencia más suave
  - Permite mayores tasas de aprendizaje
- Los mini lotes conducen a un rápido entrenamiento
  - Puede paralelizar la computación + lograr aumentos significativos de velocidad en las GPU

Sobreajuste (overfitting)

El problema del sobreajuste



Regularización

Regularización

- **¿Qué es?** Técnica que limita nuestro problema de optimización para no incentivar la generación de modelos complejos
- **¿Por qué lo necesitamos?** Mejorar la generalización de nuestro modelo sobre datos no vistos

Regularización: Parada temprana

Early Stopping

- Detener el entrenamiento antes de empezar a sobreajustar...

Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...

Regularización: Parada temprana

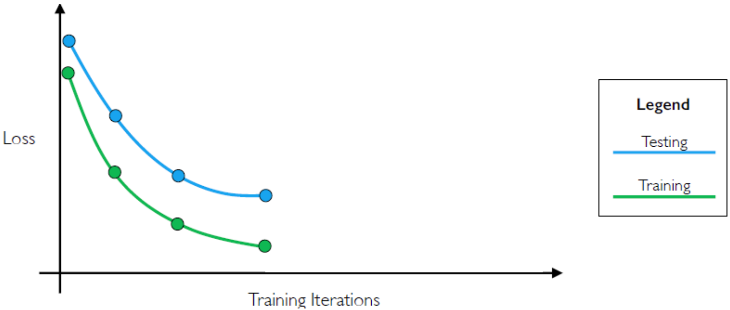
- Detener el entrenamiento antes de empezar a sobreajustar...

Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...

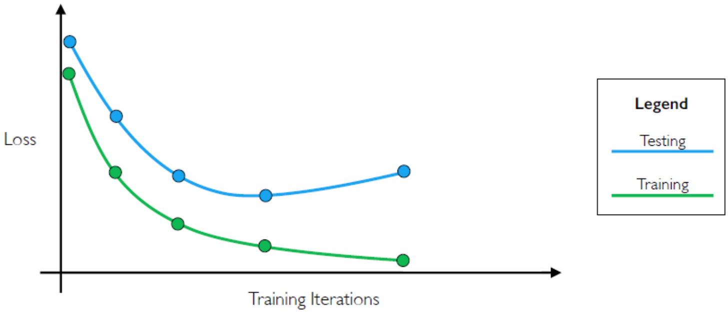
Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...



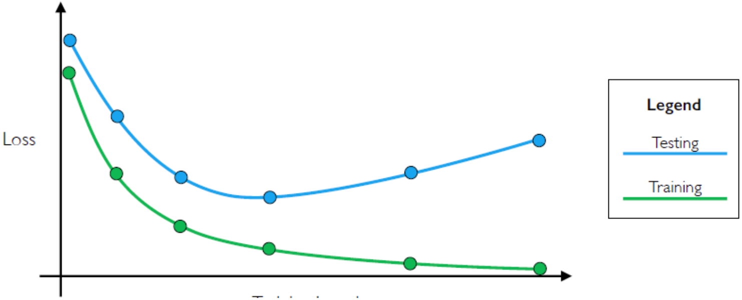
Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...



Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...



Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...

Regularización: Parada temprana

- Detener el entrenamiento antes de empezar a sobreajustar...

Muchas gracias por su atención

¿Preguntas?

Contacto: Marco Teran
webpage: marcoteran.github.io/