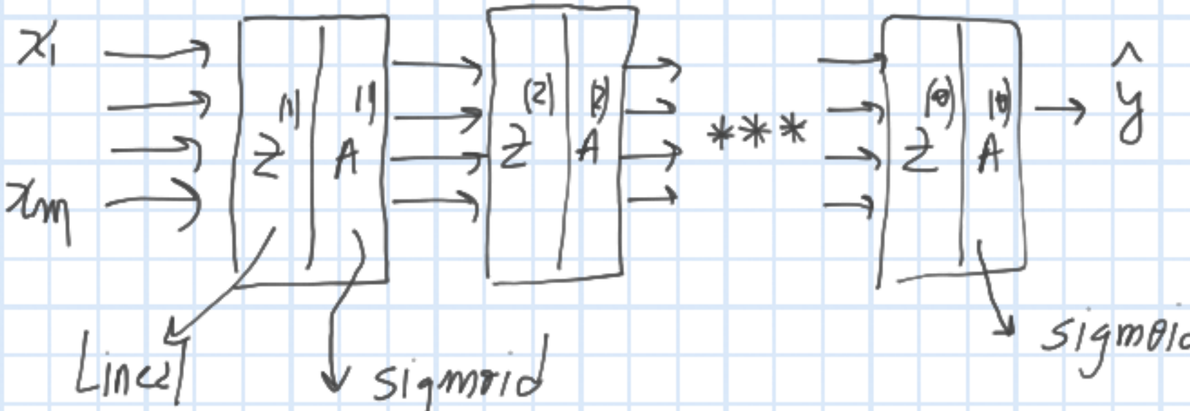
Funciones de activación



***

$\rightarrow \hat{y}$

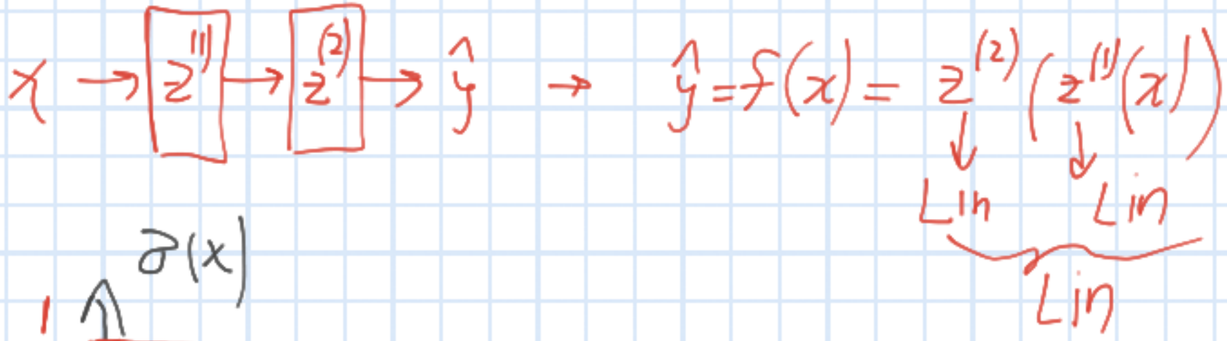
y (labels)

sigmoid → Clasificación Binaria → BCE

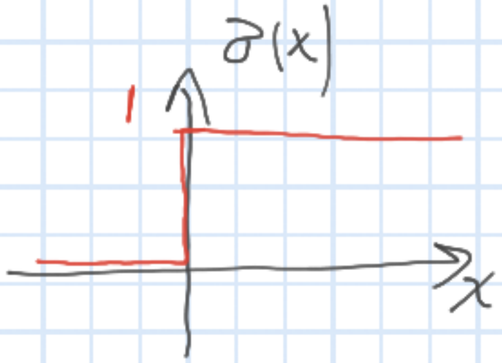
SOFTMAX → Clasificación Multiclase → CE / KL-Divergence

x → Regresión → MSE

(1) No uso función de activación



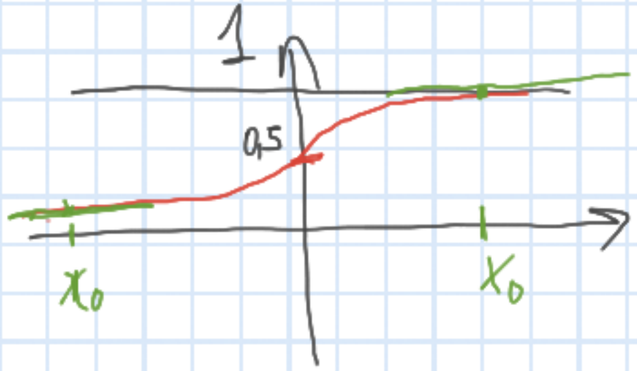
(2)



$$\sigma(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \rightarrow \sigma'(x) = \begin{cases} 0 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

$$w_i \leftarrow w_i - \alpha \frac{\partial L}{\partial w_i}$$

(3) $\sigma(x) = \sigma(x)$ sigmoid

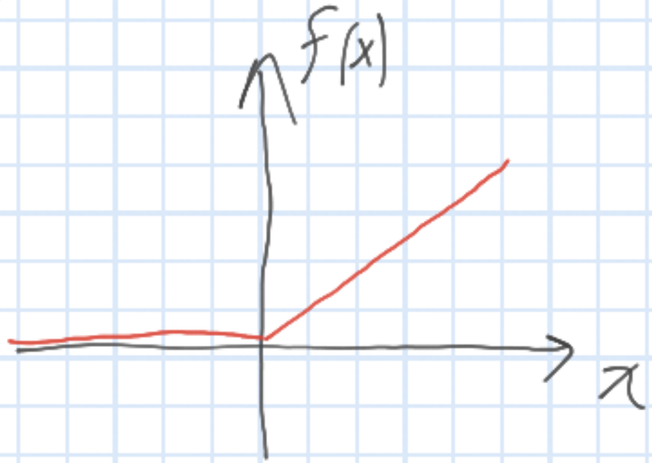


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right)$$

¿Problema? ↔ Optimización lenta
Vanishing Gradients

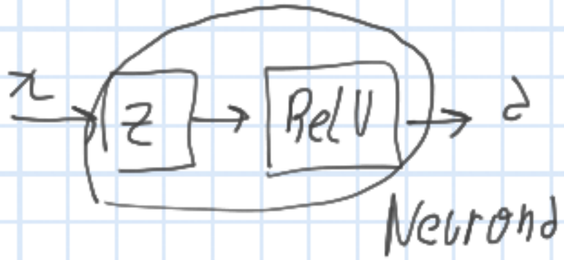
(4) ReLU



$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} = \max(x, 0)$$

$$f'(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

Para valores negativos de x la derivada es cero y mata a la neurona



(5) Leaky ReLU

$$f(x) = \begin{cases} x & x > 0 \\ 0.01x & x \leq 0 \end{cases}$$

a

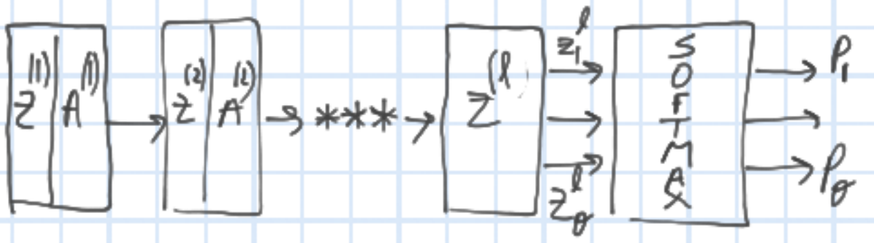
$$f'(x) = \begin{cases} 1 & x > 0 \\ 0{,}01 & x \leq 0 \end{cases}$$

(6) $\tanh_{(x)} = 2\sigma(x) - 1$

Funciones de salida

- Regresión → ϕ (ninguna) ✓
- Clasificación binaria → σ (sigmoid) ✓
- Clasificación multiclase → SOFTMAX

Clasificador de θ clases



$\sum_{i=1}^{\theta} p_i = 1$

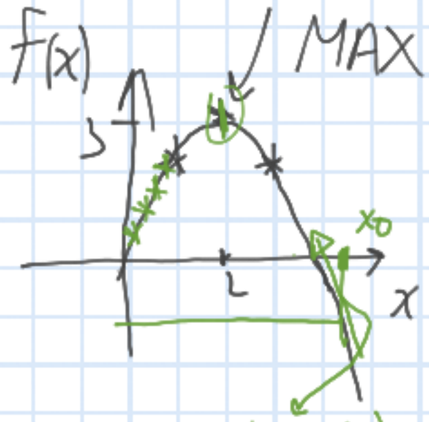
p_i es la dist de probab. de la salida $\hat{y}$

$$p_1 = \frac{e^{z_1}}{\sum_{i=1}^{\theta} e^{z_i}}$$

*
*
*

$$p_\theta = \frac{e^{z_\theta}}{\sum_{i=1}^{\theta} e^{z_i}}$$

Optimización en DL



$$x_0 \leftarrow x_0 + \alpha \frac{\partial f}{\partial x}(x_0)$$

$$f(x) = -(x-2)^2 + 3$$

Implementar GD para encontrar el maximo de $f(x)$

$$\hat{y} = NNET(x)$$

$$L = f(\bar{x}, \bar{\hat{y}}, y)$$

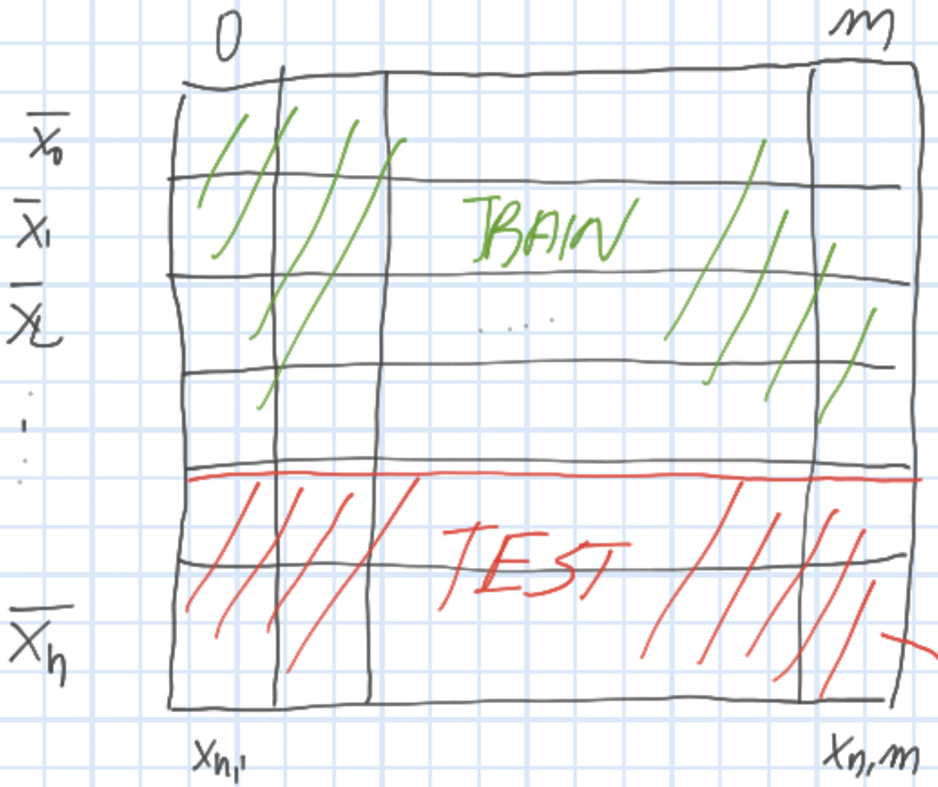
$$\bar{\hat{y}} = \bar{g}(\bar{w})$$ pesos

Optimización en DL

- ¿Cómo se trabaja con el dataset? (1)
- ¿Tengo que normalizar los datos? (2)
- Vanishing gradients (3)
- Exploding gradients (4)
- Algoritmos de optimización (5)
- Regularización (6)
  - L2
  - L1
  - Dropout

} Evitar overfitting

(1) Dataset en deep learning



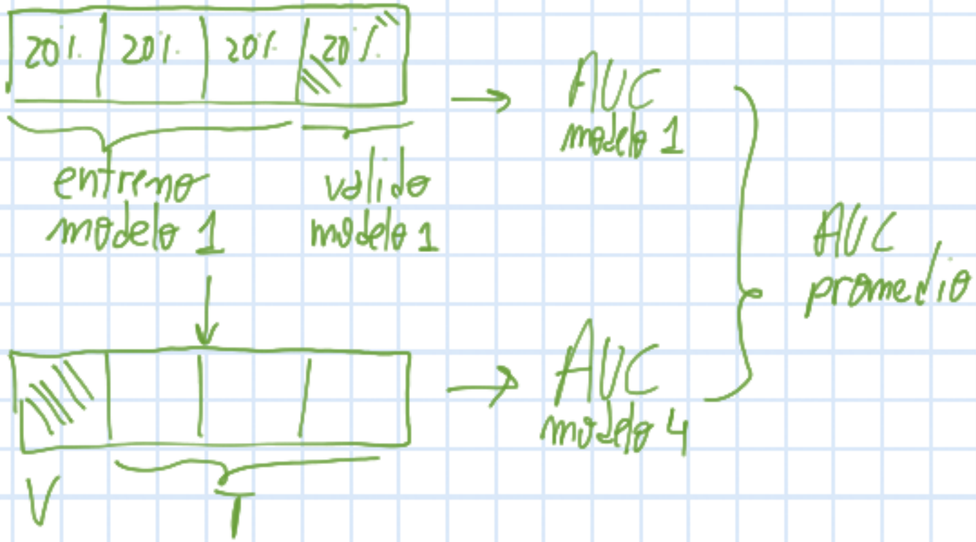
80%

20%

DL → m y n son muy grandes

ML tradicional

Cross Validation → HP search

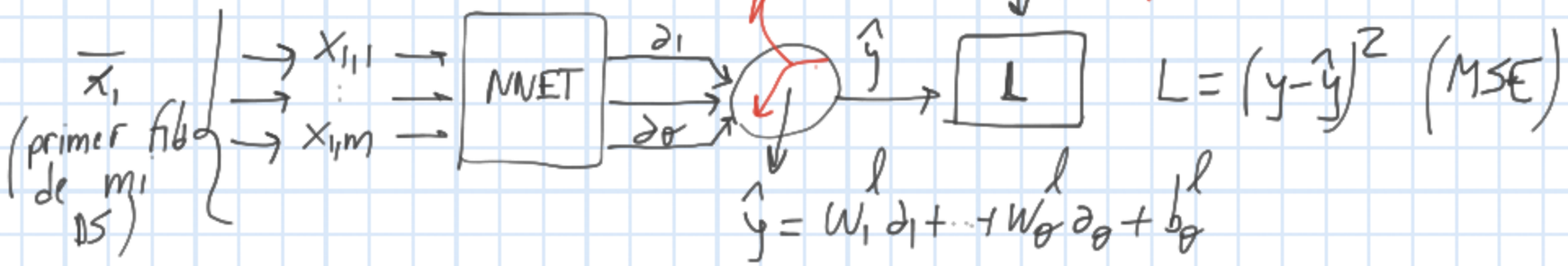


En deep learning

Calculo métricas performance de mi modelo

- → Clasificación → F1, P, R, ACC, AUC
- → Reg → RMSE

(2) ¿ Tengo que normalizar datos en DL?



$L = (y-\hat{y})^2$ (MSE)

$\hat{y} = w_1^l a_1 + \dots + w_\theta^l a_\theta + b_\theta^l$

$$\frac{\partial L}{\partial w_1^l} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_1^l} = -2(y-\hat{y})\, a_1$$

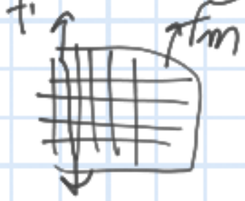
$$w_1^l \leftarrow w_1^l - \alpha\left(-2(y-\hat{y})\right) a_1$$

$$w_1^l + \alpha\, 2(y-\hat{y})\, a_1$$

$$w_2^l \leftarrow w_2^l + \alpha\, 2(y-\hat{y})\, a_2$$

Sol → normalizar datos

Normalizar datos

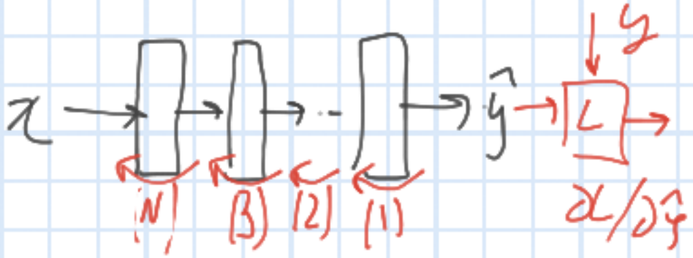


$$f_{1\_nor} = \frac{f_1 - \min(f_1)}{\max(f_1) - \min(f_1)}$$

$$f_1 \in [0,1]$$

(3) y (4)

Vanishing Gradients



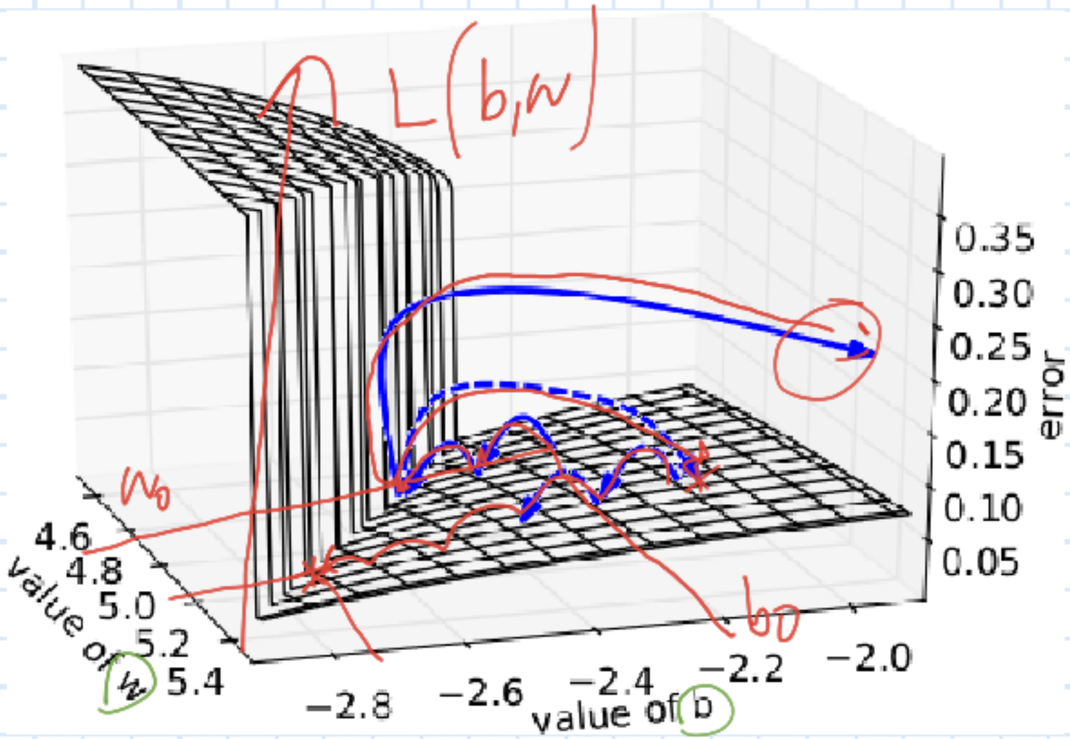
$$\frac{\partial \hat{y}}{\partial w^{(1)}} = \frac{\partial L}{\partial \hat{y}} (1)(2)(3)\cdots(N)$$

Puede ser cero si tengo muchas capas

↓

ReLU

Exploding Gradients

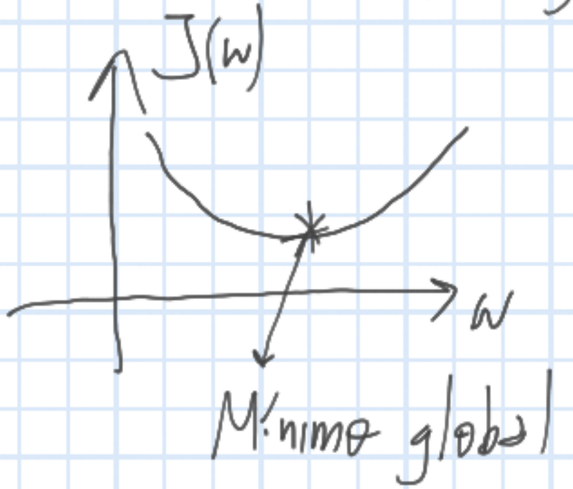


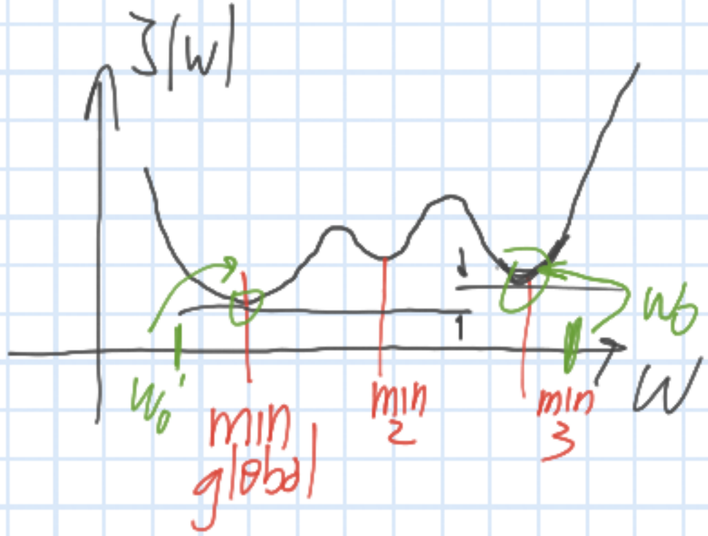
$$\hat{y} = f(x_1, b, w)$$

$$L = f(x_1, x_2, \ldots, x_n, b, w) = \frac{1}{n} \sum_{i=1}^{n} l_i(x_i, b, w)$$

MSE $(y_i - \hat{y})$ ← x_i, b, w

* Algoritmos de optimización





$w_i \sim \text{Uniforme}\left(-\frac{1}{\sqrt{m}}, +\frac{1}{\sqrt{m}}\right)$ → cantidad de datos de entrada de la neurona

* Mini-Batch SGD
  SGD con decaimiento lineal
  momento de primer orden
  AdaGrad
  RMSProp
* → Adam (2014)

} Variantes $\bar{w} \leftarrow \bar{w} - \alpha \nabla J$
Derivada primer orden

* Newton
  BFGS
  L-BFGS

} Derivada de segundo orden
H

* Mini-batch

for e in epochs:
  for b in batches:
    * Forward
    * $\overline{G} = \nabla_{\overline{w}} \left(\frac{1}{b} \sum_{i=1}^{N\_BATCH} \ell\left(mnet(\overline{x}_i, \overline{w}), y_i\right) \right)$
    * $\overline{w} \leftarrow \overline{w} - \alpha \overline{G}$

¿Qué aprende? → $\overline{w}$

¿Quienes son HP? → α, N_BATCH, N_EPOCH

* First order momentum

for e in epochs:
  for b in batches:
    * Forward: $\hat{y} = MNET(\overline{x})$ (para todo el batch)
    * $\overline{G}$ (calculo el gradiente)
    * $\overline{v} \leftarrow \nabla \overline{v} + \alpha \overline{G}$
    * $\overline{w} \leftarrow \overline{w} - \overline{v}$

HP → α, n_epoch, n_batch, ∇

$\nabla = 0$ → Vanila Mini-Batch

$\nabla > 0$

* AdaGrad

for e in epochs:
  for b in batches:
    * Forward
    * $\bar{\square}$
    * $\bar{r} \leftarrow \bar{r} + \bar{G} \odot \bar{G}$
    * $\bar{\Delta} \leftarrow \frac{-\alpha}{\sqrt{\bar{r}}} \odot \bar{G}$
    * $\bar{w} \leftarrow \bar{w} + \bar{\Delta} = \begin{bmatrix} w_1 \\ \vdots \\ w_L \end{bmatrix} - \begin{bmatrix} \alpha_1 \, \partial J/\partial w_1 \\ \vdots \\ \alpha_L \, \partial J/\partial w_L \end{bmatrix}$

element-wise multiplication

$$\begin{bmatrix} \partial J/\partial w_1 & \partial J/\partial w_1 \\ \partial J/\partial w_2 & \partial J/\partial w_2 \\ \vdots & \end{bmatrix}$$

* Adam (2014)

for e in epochs:

for b in batches:

* forward
* $\overline{G}$
* $\overline{v} \leftarrow \rho_1 \overline{V} + (1-\rho_1)\overline{G}$
* $\overline{r} \leftarrow \rho_2 \overline{r} + (1-\rho_2)\,\overline{G} \odot \overline{G}$
* $\overline{\Delta} \leftarrow -\alpha / \sqrt{r}\, V$
* $\overline{w} \leftarrow \overline{w} + \overline{\Delta}$