DistLearn: Zero-Order Decentralized Layer-wise Training via Game-Theoretic Block Coordinate Descent

Iván Jourdan[1]

Instituto de Industria. Universidad Nacional de General Sarmiento. Malvinas Argentinas, Prov de Buenos Aires. Argentina. ijourdan@scampus.ungs.edu.ar
https://www.ungs.edu.ar/idei/idei

Abstract. The training of Neural Networks (NNs) via backpropagation faces significant challenges, such as vanishing and exploding gradients. Furthermore, the computation of exact and consistent global gradients incurs a high computational cost as the network deepens. These challenges are exacerbated when aiming to deploy them in ad-hoc edge device architectures for distributed processing. In this work, we introduce *DistLearn*, a Zero-Order layer-wise decentralized training algorithm. By formulating the optimization of the NN layers as an Exact Potential Game, *DistLearn* avoids backward error propagation, encapsulating the layers as independent players. Each layer autonomously updates its parameters, employing particle filtering combined with covariance matrix adaptation evolution strategies (CMA-ES) and surrogate macro-gradients. Experimental evaluations indicate that *DistLearn* is capable of effectively escaping local minima. In the conducted experiments, *DistLearn* achieved an 82% success rate on targets with an $RMSE \leq 0.07$ within a 1-6-4-4-1 topology, compared to the performance of the ADAM optimizer (44%). These results suggest that *DistLearn* could be a viable, gradient-free alternative for the optimization of non-convex problems in deep architectures.

Keywords: zero-order optimization, decentralized training, exact potential games, CMA-ES , backpropagation-free

DistLearn: Entrenamiento de orden cero descentralizado por capas mediante descenso de coordenadas por bloques basado en teoría de juegos

Resumen El entrenamiento de Redes Neuronales (NN) mediante *back-propagation* enfrenta desafíos significativos, tales como el desvanecimiento y la explosión del gradiente. Además, el cálculo de gradientes globales exactos y consistentes presenta un alto costo computacional a medida que se profundiza la red. Estos desafíos se ven acentuados si se propone aplicar en arquitecturas de dispositivos AD-HOC de borde para procesamiento distribuido. En este trabajo, se introduce *DistLearn*, un algoritmo de entrenamiento descentralizado por capas de Orden Cero. Al formular la optimización de las capas de la NN como un *Juego de Potencial Exacto*, *DistLearn* evita la propagación del error hacia atrás, encapsulando a las capas como jugadores independientes. Cada capa actualiza autónomamente sus parámetros, empleando filtrado particular en combinación con estrategias evolutivas basadas en matrices de covarianza (CMA-ES) y macro-gradientes subrogados. Evaluaciones experimentales realizadas indican que DistLearn es capaz de sortear mínimos locales de manera efectiva. En los experimentos realizados, *DistLearn* logró un éxito del 82 % en objetivos con un RMSE ≤ 0.07 en una topología 1-6-4-4-1, en comparación con el desempeño del optimizador ADAM (44 %). Estos resultados sugieren que *DistLearn* pudiera ser una alternativa viable, libre de gradientes, para la optimización de problemas no convexos en arquitecturas profundas.

Keywords: optimización de orden cero, entrenamiento descentralizado, juegos de potencial exacto, CMA-ES , libre de backpropagation

1. Introducción

Los algoritmos de descenso de gradiente subyacen a la arquitectura de los modelos de aprendizaje profundo que definen el estado del arte actual, en particular *Backpropagation* (BP) (Rumelhart et al., 1986). No obstante, este último presenta limitaciones serias, es sensible al desvanecimiento del gradiente y requiere un conocimiento completo del modelo hacia adelante. El algoritmo *forward-forward* (Hinton, 2022) surge como una alternativa para sortear estas limitaciones. Por otra parte, los métodos de optimización de Orden Cero (ZO), son más robustos que BP, pero su aplicación sufre de la *maldición de la dimensionalidad* (Liu et al., 2020; Qian et al., 2016). Existen trabajos recientes que intentan mitigar este problema adoptando diferentes estrategias: reducción de la dimensionalidad mediante proyecciones en subespacios aleatorios de bajo rango (SubZero Yu et al., 2024), fragmentando el objetivo global en sub-pérdidas independientes a través del aprendizaje local (Local Learning Ren et al., 2022), o intentando aproximar BP mediante estimaciones Jacobianas jerárquicas (e.g., HZO Cao et al., 2026). Lamentablemente, estos métodos heredan la acumulación exponencial del error, propia de la regla de la cadena.

En este artículo se presenta *DistLearn*, un marco de entrenamiento de orden cero, descentralizado y por capas, que controla el problema de la varianza mediante la *Descomposición por Coordenadas de Bloques* fundamentada en la *Teoría de Juegos*. En este trabajo se formula teóricamente a *DistLearn* como un Juego de Potencial Estocástico, garantizando la convergencia monótona y se realiza una evaluación preliminar de robustez, donde *DistLearn* es comparado con Backprop Adam (Kingma y Ba, 2017).

2. Aprendizaje distribuido. *DistLearn*

Se puede definir un modelo de redes neuronales de $N \in \mathbb{N}_{\geq 0}$ capas, como una composición de funciones $f_\lambda : \mathbb{R}^{r(\lambda-1)} \to \mathbb{R}^{r(\lambda)}$, $\hat{y} = f(x|\Theta) = (f_N \circ f_{N-1} \circ \cdots \circ f_1)(x) \quad con \quad f_\lambda(x_{\lambda-1}|\theta_\lambda) = g(\theta_\lambda^T x_{\lambda-1})$, siendo $\lambda \in 1 \ldots N$, y $r(\lambda-1), r(\lambda)$ las dimensiones del dominio y co-dominio de las funciones. Finalmente, $\theta_\lambda \in \mathbb{R}^{r(\lambda-1),r(\lambda)}$ son los coeficientes de f_λ, y $g(\cdot)$ una función de activación.

Para entrenar f, se define un espacio $\mathcal{C}$ del que se adquieren muestras de entrenamiento $(x, y) \in \mathcal{C}$, y una métrica de verosimilitud $\mathcal{L}(y, f(x)|\theta) \in (0, 1)$ sobre el error $\varepsilon = \|y - f(x|\theta)\|$ que comete la red al estimar y cuando se proporciona como entrada x. La verosimilitud que posee f se determina evaluando sobre el espacio muestral $\mathcal{C}$: $\mathcal{L}_f(\theta) = E_{(x,y)\in\mathcal{C}}[\mathcal{L}(y, f(x)|\theta)]$. El problema de optimización consiste en encontrar el conjunto de parámetros del modelo que maximice su verosimilitud, $\theta^* = \text{argmax}_\theta [\mathcal{L}_f(\theta)]$.

2.1. Convergencia de *DistLearn*. Un enfoque desde la teoría de juegos.

Pensando al modelo f como una composición de funciones, se tienen N **capas** o **jugadores** f_λ, con $\lambda = \{1, 2, \ldots, N\}$. Los posibles parámetros de cada capa se

representan por un **espacio de acciones** $A_\lambda : \{\theta_\lambda \in \mathbb{R}^{r(\lambda-1)\times r(\lambda)}\}$, asociado a cada jugador, y se define π_λ una política avara (greedy) de selección de acciones pertenecientes al espacio A_λ (ec. 2), mediante la cual se definen los parámetros de la capa f_λ. Para medir si un parámetro es mejor respecto a otro, se adopta una **función de utilidad**, planteada como la verosimilitud del modelo, $\ell_\lambda(\theta_\lambda)$, según la acción o parámetro θ_λ,

$$\ell_\lambda(\theta_\lambda) = E_{\pi_{-\lambda},(x,y)}\left[\mathcal{L}(y, f(x|\theta))|\theta_\lambda\right] \tag{1}$$

$$\pi_\lambda(\theta_\lambda) = \begin{cases} 1 & si\ \theta_\lambda = \theta_\lambda^\star = \underset{\forall\theta\in A_\lambda}{\operatorname{argmax}}\, \ell_\lambda(\theta) \\ 0 & en\ otro\ caso \end{cases} \tag{2}$$

siendo $\pi_{-\lambda}$ la política de las demás capas $(-\lambda)$, de manera tal que cada capa f_λ se configura con los parámetros $\theta_\lambda^\star$ que le permitan maximizar su utilidad. Bajo estas condiciones, cada capa observa en probabilidad la verosimilitud de la red, configurándose en un **juego de potencial exacto**.

$$\begin{aligned} \ell_\lambda(\theta_\lambda^\star) &= E_{\pi_\lambda}\left[\ell_\lambda(\theta_\lambda)\right] = E_{\pi_\lambda,\pi_{-\lambda},(x,y)}\left[\mathcal{L}(y, f(x|\theta))|\theta_\lambda\right] \\ &\overset{p}{=} E_{(x,y)}\left[\mathcal{L}(y, f(x|\Theta^\star))\right] = \mathcal{L}_f(\Theta^\star) \end{aligned}$$

donde $\Theta^\star = \{\theta_1^\star, \ldots, \theta_N^\star\}$ es el conjunto de todos los parámetros seleccionados por las capas. Luego, para cualquier capa λ, y para cualquier par de acciones $\theta_\lambda, \theta_\lambda^\dagger \in A_\lambda$ el cambio de utilidad de un jugador, al modificar unilateralmente de acción, es exactamente igual al cambio en la función de potencial:

$$\ell_\lambda(\theta_\lambda^\dagger|\theta_{-\lambda}) - \ell_\lambda(\theta_\lambda|\theta_{-\lambda}) = \Delta > 0 \implies \mathcal{L}(\theta_\lambda^\dagger, \theta_{-\lambda}) - \mathcal{L}(\theta_\lambda, \theta_{-\lambda}) = \Delta$$

Cada capa de la red, a su turno, busca mejorar su utilidad, y con ello mejora la verosimilitud de la red $\mathcal{L}$. Siendo $\mathcal{L}$ acotada, existe una secuencia finita de cambios de acciones hasta alcanzar el equilibrio de Nash y con ello un máximo local de la función de verosimilitud $\mathcal{L}$ (Monderer y Shapley, 1996).

2.2. Algoritmo. Optimización de orden cero

El entrenamiento se basa en medir la verosimilitud de la red observada por cada capa (ec. 1). Para estimar $\ell_\lambda(\theta_\lambda)$ se utiliza filtrado particulado (Thrun et al., 2005) a partir de un muestreo de P partículas del espacio de acciones A_λ: $\Theta_\lambda : \{\theta_\lambda^1, \theta_\lambda^2, \ldots, \theta_\lambda^P\} \subset A_\lambda$. Al conjunto de partículas Θ_λ se asocia un conjunto de coeficientes convexos, $\Omega_\lambda : \{\omega_\lambda^1, \omega_\lambda^2, \ldots, \omega_\lambda^P\}$ que miden la importancia de una partícula frente a otra, según la verosimilitud observada: $\omega_\lambda^i \propto \ell_\lambda(\theta_\lambda^i)$. Este *enjambre* es un modelo de la verosimilitud $\ell_\lambda(\theta_\lambda)$ que observa la capa (ec. 1), y a partir de este se puede determinar la política de selección de acciones $\pi_\lambda(\theta_\lambda)$ (ec. 2), siendo aquella que selecciona la partícula con mayor peso. El esquema de optimización se describe en el Algoritmo 1.

Algoritmo 1 Entrenamiento de Orden Cero Descentralizado (DistLearn).
$\mathcal{N}_{eff}$ representa el número efectivo de muestras para evitar la pérdida de diversidad (Thrun et al., 2005), η es la tasa de aprendizaje, $\bar{\mu}$ es un vector dirección entre las mejores y peores partículas de la capa. Se considera $\mathcal{N}$ gaussiano. CMA-ES: ver Hansen, 2006

Inicialización: Para cada capa λ, muestrear $\Theta_\lambda \leftarrow \{\theta_\lambda^1, \ldots, \theta_\lambda^P\}$ e inicializar coeficientes $\Omega_\lambda \leftarrow \{\omega_\lambda^1, \ldots, \omega_\lambda^P\}$
repeat
  Seleccionar capa λ para optimizar.
  Fijar las demás capas $-\lambda$ en su mejor partícula: $\theta_{-\lambda}^* \leftarrow \text{argmax}\, \Omega_{-\lambda}$
  for cada partícula $\theta_\lambda^i \in \Theta_\lambda$ **do**
    En épocas de J pasos, evaluar verosimilitud $\mathcal{L}(\theta_\lambda^i, \theta_{-\lambda}^*)$ en el corpus $\mathcal{C}$
    $\omega_\lambda^i \leftarrow \omega_\lambda^i \left(\frac{1}{J} \sum_{j=1}^{J} \mathcal{L}(\theta_\lambda^i, \theta_{-\lambda}^*)_j \right)$ ▷ Actualizar importancia
    $\theta_\lambda^i \leftarrow \theta_\lambda^i + \eta \left(1 - \frac{1}{J} \sum_{j=1}^{J} \mathcal{L}(\theta_\lambda^i, \theta_{-\lambda}^*)_j \right) \mathcal{N}(\bar{\mu}, \sigma)$ ▷ Movimiento (CMA-ES)
  end for
  if $\mathcal{N}_{eff} < P/2$ **then** ▷ Comprobar pérdida de diversidad
    Remuestreo: Generar nuevas partículas Θ_λ e inicializar Ω_λ
  end if
until Convergencia (Equilibrio de Nash)

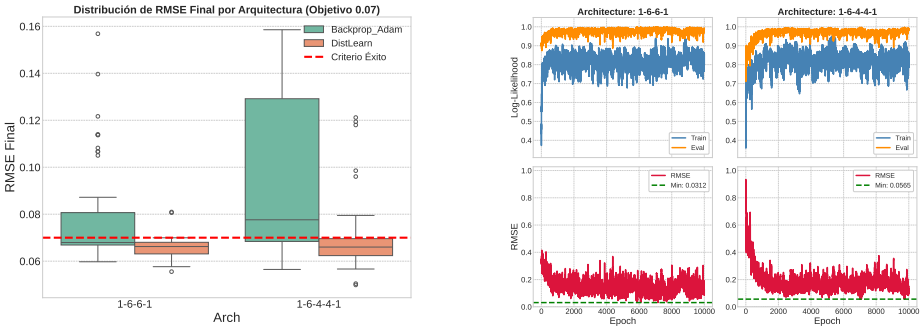
3. Experimentos

Se evaluó empíricamente el desempeño de *DistLearn* frente a Backprop Adam (Kingma y Ba, 2017) utilizando dos topologías de poca profundidad: 1-6-6-1 y 1-6-4-4-1. Ambas redes poseen una cantidad similar de parámetros y abundantes valles locales, resultando aptas para evaluar la convergencia de los optimizadores, al no suavizarse las funciones de sub-pérdida (Choromanska et al., 2015). Se evaluó entonces un problema de regresión para la función $y = -\sin(\pi \times x)$ con $x \in [-1, 1]$. Se configura a *DistLearn*, para el experimento, con $P = 42$ partículas, tasa de aprendizaje $\eta = 0.05$ y un objetivo de convergencia de RMSE ≤ 0.07, para el que se tiene una clara aproximación a la función sinusoidal. Se realizaron 100 repeticiones de entrenamiento por optimizador empleando PyTorch (GPU NVIDIA RTX-2060).

Se muestra en la Tabla 1 los resultados, en ambas topologías. *DistLearn* logra una mayor tasa de éxito y un menor RMSE (global y en fallos), producto de su estrategia evolutiva que explora el espacio de manera más exhaustiva que el método basado en gradiente. En la topología 1-6-6-1, la cantidad de evaluaciones resulta comparable dado que *DistLearn* evalúa las partículas en paralelo (Salimans et al., 2017). Asimismo, al reducir la dimensionalidad de representación en 1-6-4-4-1, la limitada exploración de Backprop Adam provoca que caiga a 44 % la proporción de éxito, frente al 82 % de *DistLearn*, exhibiendo además una variabilidad superior en su error distribuido (Fig. 1a). En consecuencia, *DistLearn* muestra una mayor robustez frente a las variaciones topológicas orientadas a la reducción de dimensionalidad. El método logra mantener el error acotado, incluso

Tabla 1: Comparativa de rendimiento entre DistLearn y Backprop Adam para redes 1-6-6-1 y 1-6-4-4-1. P-valores (Mann-Whitney): para (1-6-6-1) $\rightarrow$ Global=1.28×10^{-4}, Fallos=6.62×10^{-2}; para (1-6-4-4-1) $\rightarrow$ Global=5.97×10^{-7}, Fallos=5.14×10^{-3}.

Topología	Optimizador	Tasa de Éxito	Evaluaciones (Media Éxito)	RMSE Global (Media ± SD)	RMSE Fallos (Media ± SD)
1-6-6-1	DistLearn	**0.96**	150 388	**0.0658** ± 0.0047	**0.0808** ± 0.0002
	Backprop Adam	0.70	130 410	0.0785 ± 0.0221	0.1062 ± 0.0229
1-6-4-4-1	DistLearn	**0.82**	213 116	**0.0695** ± 0.0153	**0.0950** ± 0.0207
	Backprop Adam	0.44	104 400	0.0980 ± 0.0347	0.1227 ± 0.0272



(a) Distribución de RMSE.

(b) Comparativa de entrenamiento.

Figura 1: Resultados del experimento.

en los casos donde no se alcanza el objetivo global, y exhibe una trayectoria de optimización más estable tanto en las métricas de RMSE como de verosimilitud (Fig. 1b)

4. Conclusiones

DistLearn, como algoritmo de orden cero sufre de la *explosión de dimensionalidad*, sin embargo, al optimizar cada capa en forma independiente, en un juego de potencial que alcanza el equilibrio de Nash, se reduce el impacto de esta limitación. Los resultados experimentales muestran que *DistLearn* logra sintonizar dos modelos de poca profundidad, con una mayor tasa de éxito, un menor RMSE y cantidad de evaluaciones comparables a Backprop Adam, optimizando cada capa en forma independiente. Este hecho plantea la posibilidad de utilizar *DistLearn* en conjunto con capas tipo *cajas negras*, donde sólo se tiene acceso al conjunto de parámetros de ajuste, o también capas *"Extreme Learning Machine"* (Huang et al., 2006), que permitirían aumentar la profundidad sin impactar en la dimensionalidad. Siendo este trabajo una primera aproximación, a futuro, se

planea evaluar el desempeño de *DistLearn* en modelos de mayor profundidad y complejidad, explorando sinergias con otros trabajos de optimización de orden cero, particularmente HZO (Cao et al., 2026) y SubZero (Yu et al., 2024), que han demostrado ser promisorios en tareas de optimización de redes neuronales profundas.

Referencias

Cao, S., Ma, Z., & Tian, Y. (2026). Hierarchical Zero-Order Optimization for Deep Neural Networks. https://arxiv.org/abs/2602.10607

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. *Artificial intelligence and statistics*, 192-204.

Hansen, N. (2006). The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, 75-102.

Hinton, G. (2022). The Forward-Forward Algorithm: Some Preliminary Investigations. https://arxiv.org/abs/2212.13345

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*(1-3), 489-501.

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., & Varshney, P. K. (2020). A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, *37*(5), 43-54.

Monderer, D., & Shapley, L. S. (1996). Potential Games. *Games and Economic Behavior*, *14*(1), 124-143. https://doi.org/10.1006/game.1996.0044

Qian, H., Hu, Y.-Q., & Yu, Y. (2016). Derivative-Free Optimization of High-Dimensional Non-Convex Functions by Sequential Random Embeddings. *IJCAI*, 1946-1952.

Ren, M., Kornblith, S., Liao, R., & Hinton, G. (2022). Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533-536.

Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. https://arxiv.org/abs/1703.03864

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).

Yu, Z., Zhou, P., Wang, S., Li, J., & Huang, H. (2024). Subzero: Random subspace zeroth-order optimization for memory-efficient llm fine-tuning.