Computação Evolutiva no treinamento de Redes Neurais

Euclides Carlos Pinto Neto
David Fernandes Neves Oliveira
Macilio da Silva Ferreira

PSI5886 - Prof. Emilio Del Moral Hernandez

Agenda

- Objetivo
- Otimização
- Método
- Datasets
- Modelos
- Resultados

Grupo

- Euclides Carlos Pinto Neto 
- David Fernandes Neves Oliveira 
- Macilio da Silva Ferreira 

Grupo

- **Euclides Carlos Pinto Neto**
  - Computação evolutiva, Integração e experimentos
- **David Oliveira**
  - Datasets, Integração e experimentos
- **Macilio Ferreira**
  - Integração e experimentos

Otimização

- **Marc Toussaint [1]**
  - **Gradient**-based optimization **(1st order methods)**
    - Plain grad., steepest descent, conjugate grad., Rprop, **stochastic grad**.
  - **Black box optimization** (“**0th** order methods”)
    - **Evolutionary** Algorithms
    - "**Blackbox optimization** is often related to **learning**”
    - e.g. **PSO**
      - (**Tendência**) convergência mais rápida [2] [3] [4]

[1] Marc Toussaint. Optimization Course SS 13 U Stuttgart. https://ipvs.informatik.uni-stuttgart.de/mlr/marc/teaching/13-Optimization/.

[2] MOHAGHEGI, S. et al. A comparison of pso and backpropagation for training rbf neural networks for identification of a power system with statcom. In: IEEE. Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. [S.l.], 2005. p. 381–384

[3] GUDISE, V. G.; VENAYAGAMOORTHY, G. K. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: IEEE. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No. 03EX706). [S.l.], 2003. p. 110–117.

[4] Yinyu Ye. Zero-Order and First-Order Optimization Algorithms I. https://web.stanford.edu/class/msande311/lecture10.pdf

Objetivo

- **Comparar** a **performance** de **um modelo** de Rede Neural em diferentes problemas de **regressão**
- **Treinamento**:
    - **SGD**
    - **Evolutionary Algorithms ("Swarm-Based") + SGD**

Otimização

- **Particle Swarm Optimization (PSO)[1][11]**

```
BEGIN
  Initialize agents
  Find current best
  Set global best = current best
  FOR i= 0 : number of iterations
    Calculate particle velocity
    Chage particles velocity
    Update particles positions
    Select new agents according to the selection strategy
    IF current best better than global best
      SET global best to current best
    END IF
  END FOR
  save global best
END
```

1 https://github.com/SISDevelop/SwarmPackagePy

[11] Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *evolutionary computation, 2001. Proceedings of the 2001 Congress on* (Vol. 1, pp. 81-86). IEEE.

Otimização

- **Whale Swarm Algorithm[1] [12]**

```
BEGIN
 Initialize agents
 Find current best
 global best = current best
 FOR t = 0 : number of iterations
   FOR each agent
     find better and nearest
     IF  Exists
       move current agent in direction of its better and nearest
     END IF
     find current best
     IF current best better than global best
       SET global best to current best
     END IF
   END FOR
  Save golobal best
END
```

1 https://github.com/SISDevelop/SwarmPackagePy

[12] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67.

Otimização

- **Firefly Algorithm[1][13]**

```
Objective function f(x), x=(x_1, x_2, ... , x_d)^T
Initialize a population of fireflies x_i(i = 1, 2, ... , n)
Define light absorption coefficient gamma
WHILE count < MaximumGeneratons
  FOR i = 1 : n (all n fireflies)
    FOR j = 1 : i
    Light intensity Ii at x_i is determined by f(x_i)
    IF I_i > I_j
        Move firefly i towards j in all d dimensions
      ELSE
        Move firefly i randomly
      END IF
      Attractiveness changes with distance r via exp[-γ r_2]
    Determine new solutions and revise light intensity
    END FOR j
  END FOR i
  Rank the fireflies according to light intensity and find the current best
END WHILE
```

1 https://github.com/SISDevelop/SwarmPackagePy

[13] Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. *arXiv preprint arXiv:1308.3898*.

Otimização

- **Harmony Search[1] [14]**

Step 1. Randomly generate initial harmony in the domain D {$h_i \in D$}.
Step 2. On each iteration generate new zero harmony h_{new}
Step 3. For each component of a new harmony generate a random number ε from 0 to 1.
Step 4. IF $f(h_{new})$ is better than Gbest, $h_{new} = h_{Gbest}$

1 https://github.com/SISDevelop/SwarmPackagePy

[14] Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *simulation*, *76*(2), 60-68.

Otimização

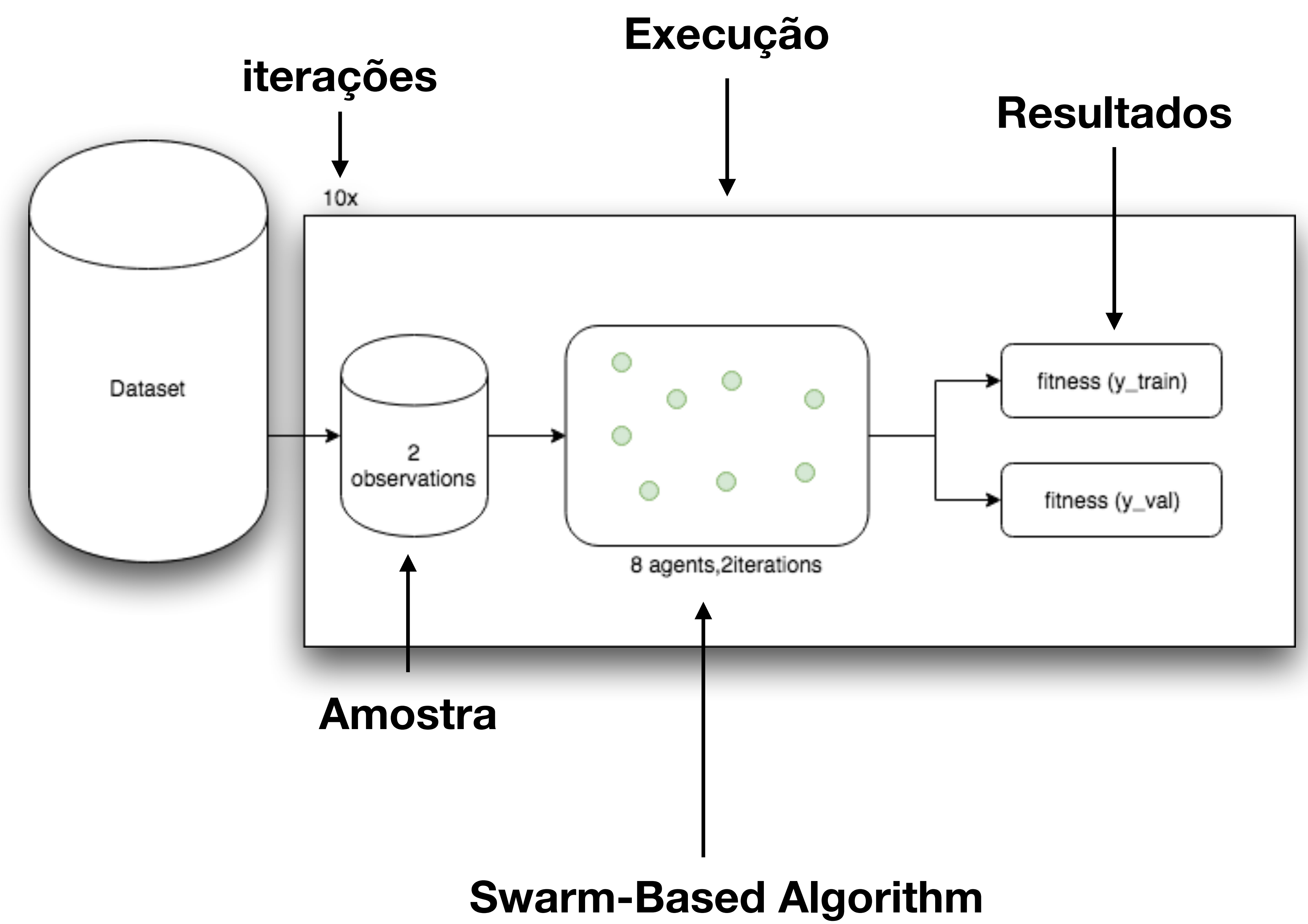
- **Bat Algorithm[1] [15]**

```
BEGIN
Objective function f(x), x=(x₁, ...,x_d)^T
Initialize the bat population x_i (i= 1, 2, ..., n) and v_i
Define pulse frequency f_i at x_i
Initialize pulse rates r_i and the loudness A_i
  WHILE count < max number of iterations
    Generate new solutions by adjusting frequency, and updating velocities and locations/solutions
    IF rand > r_i
      Select a solution among the best solutions
      Generate a local solution around the selected best solution
    END IF
    Generate a new solution by flying randomly
    IF rand < A_i AND f(x_i) < f(x_*)
      Accept the new solutions
      Increase r_i and reduce A_i
    END IF
    Rank the bats and find the current best x_*
  END WHILE
  Postprocess results and visualization
```

1 https://github.com/SISDevelop/SwarmPackagePy

[15] Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74). Springer, Berlin, Heidelberg.

Computação Evolutiva



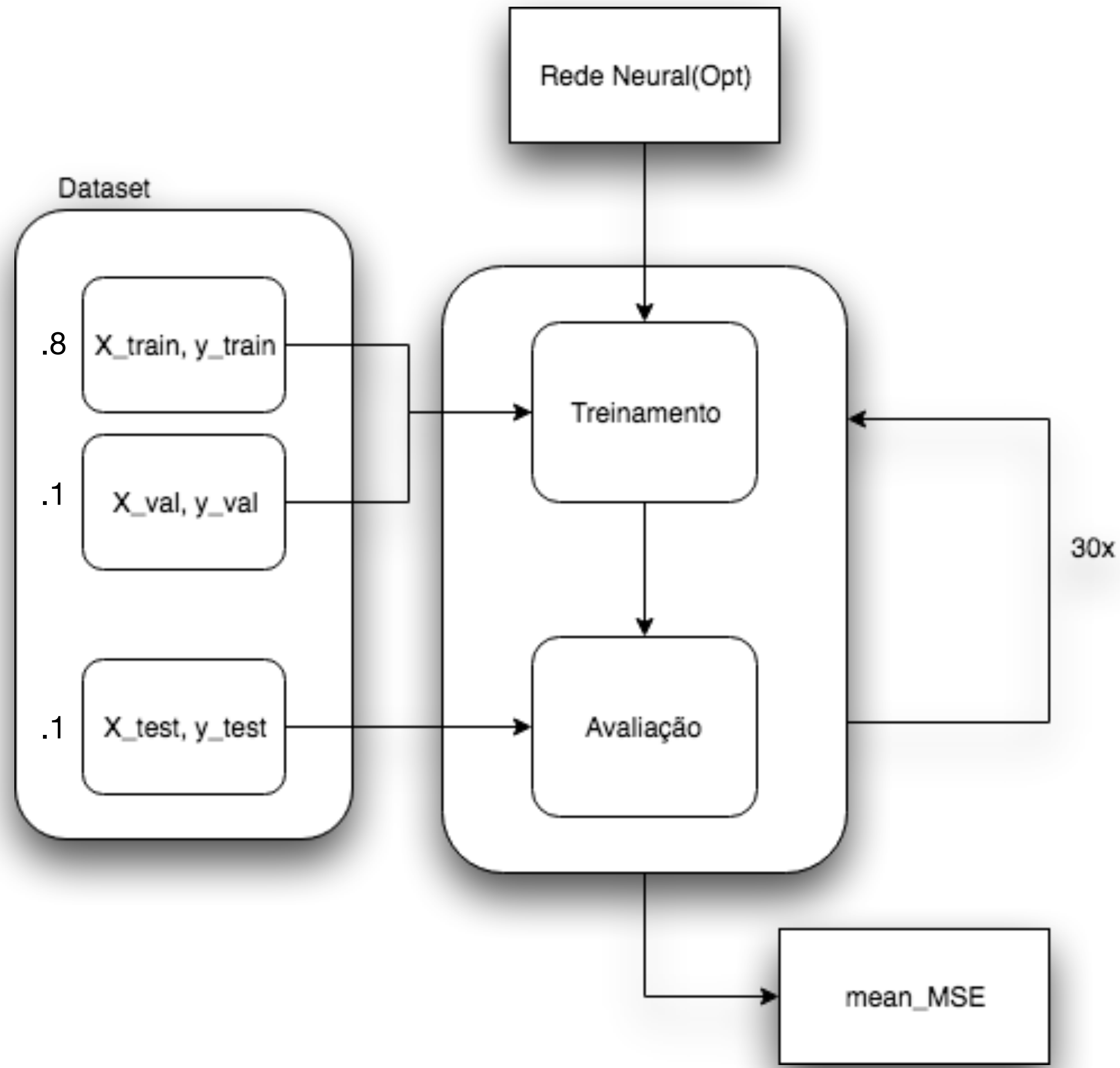
Otimização

SwarmPackagePy [1]

SwarmPackagePy is a Library of swarm optimization algorithms. It includes 14 optimization algorithms and each can be used for solving specific optimization problem. You can find the principles they operate on and pseudo codes below.

1 https://github.com/SISDevelop/SwarmPackagePy

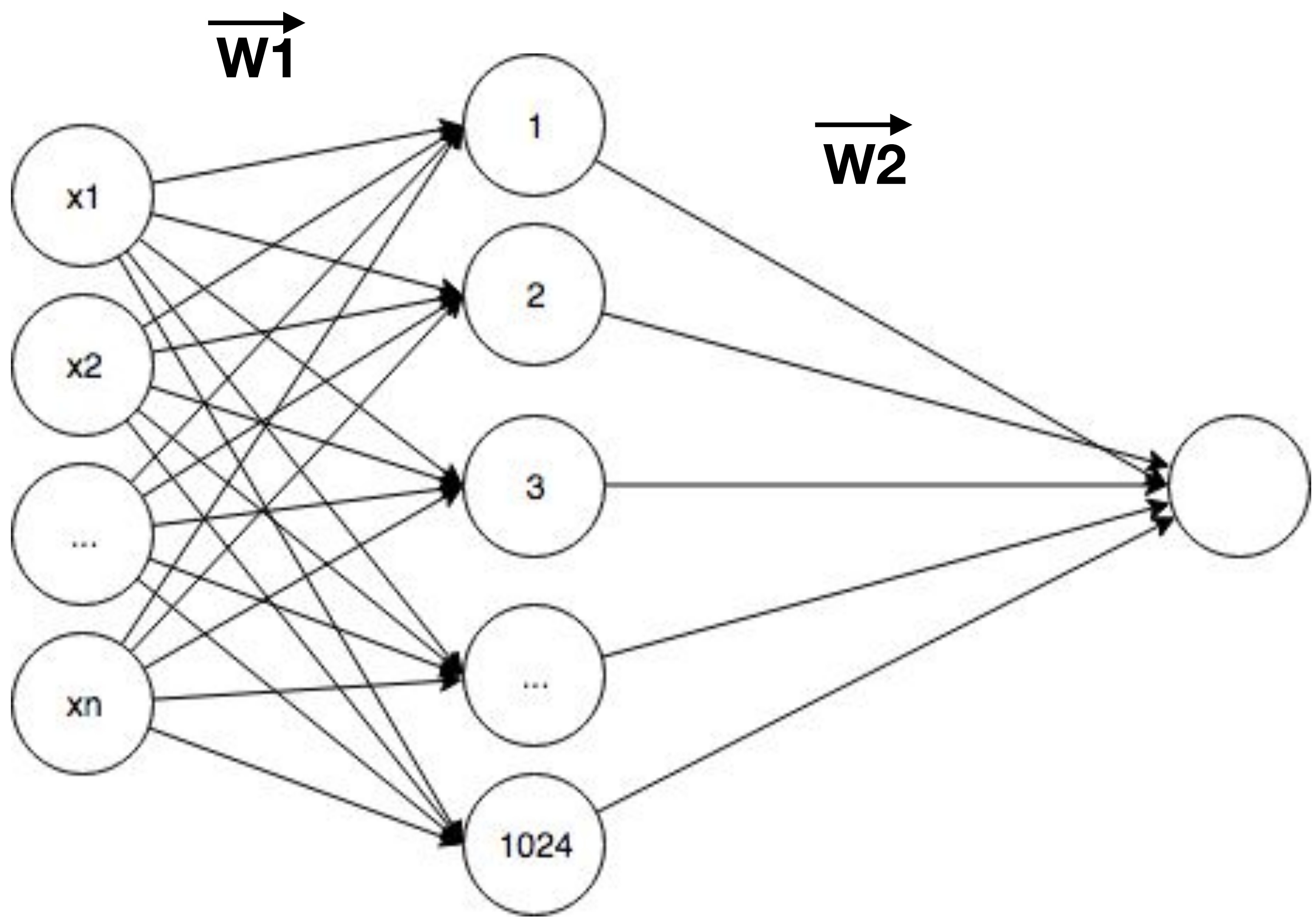
Método


Rede Neural(Opt)
Dataset
.8
X_train, y_train
.1
X_val, y_val
.1
X_test, y_test
Treinamento
Avaliação
30x
mean_MSE

Problemas de Regressão

	Atributos	Tamanho	Objetivo	Fonte
Abalone [5]	8	4177	Rings	UCI: https://archive.ics.uci.edu/ml/datasets/abalone
Bank [6]	33	8192	rej	Delve: https://www.cs.toronto.edu/~delve/data/bank/desc.html
Boston [7]	14	506	Median value of owner-occupied homes (MEDV)	Delve: https://www.cs.toronto.edu/~delve/data/boston/desc.html
Forward Kinematics [8]	33	8192	y	Delve: https://www.cs.toronto.edu/~delve/data/kin/desc.html
Computer System Activity [9]	22	8192	% CPU usage	Delve: https://www.cs.toronto.edu/~delve/data/comp-activ/desc.html
add10 [10]	11	9792	y	Delve: https://www.cs.toronto.edu/~delve/data/add10/desc.html

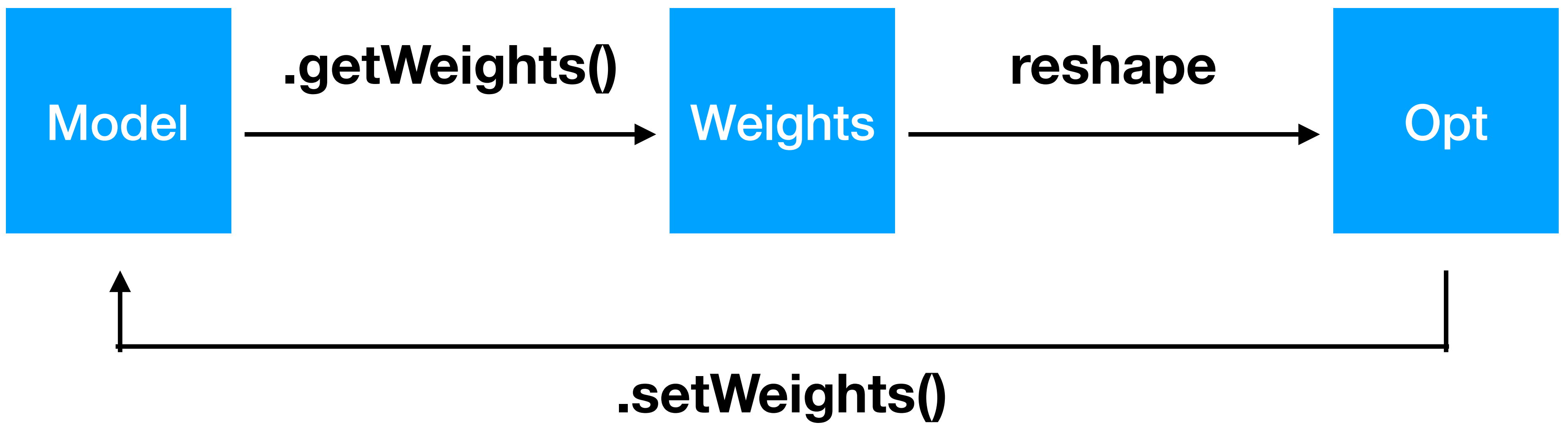
Keras



Bias, sigmoid

Implementação

- **Keras**



Implementação

```
In [18]: model = Sequential()
         model.add(Dense(2, input_shape = X.shape[1:]))
         model.add(Activation('sigmoid'))
         model.add(Dense(1))
         model.compile(loss='mse', optimizer='sgd', metrics=['mse'])

In [19]: model.get_weights()

Out[19]: [array([[ 0.69727975, -0.39692822],
                 [-0.41678765,  0.27229577],
                 [-0.13999617,  0.05079371],
                 [ 0.5199296 , -0.6908521 ],
                 [-0.33442158, -0.32437125],
                 [-0.47489583, -0.43757632],
                 [-0.5660593 ,  0.01423705],
                 [-0.6334289 ,  0.20981878],
                 [ 0.03570968,  0.64675194],
                 [-0.16250408, -0.42136058]], dtype=float32),
          array([0., 0.], dtype=float32),
          array([[0.6775118],
                 [0.7084166]], dtype=float32),
          array([0.], dtype=float32)]

In [ ]: model.set_weights(parsed_weights)
```

Modelos

	Epochs
SGD	200
SGD	150
SGD	100
PSO+SGD	(60) 90
BA+SGD	(60) 90
FA+SGD	(60) 90
WSA+SGD	(60) 90
HS+SGD	(60) 90

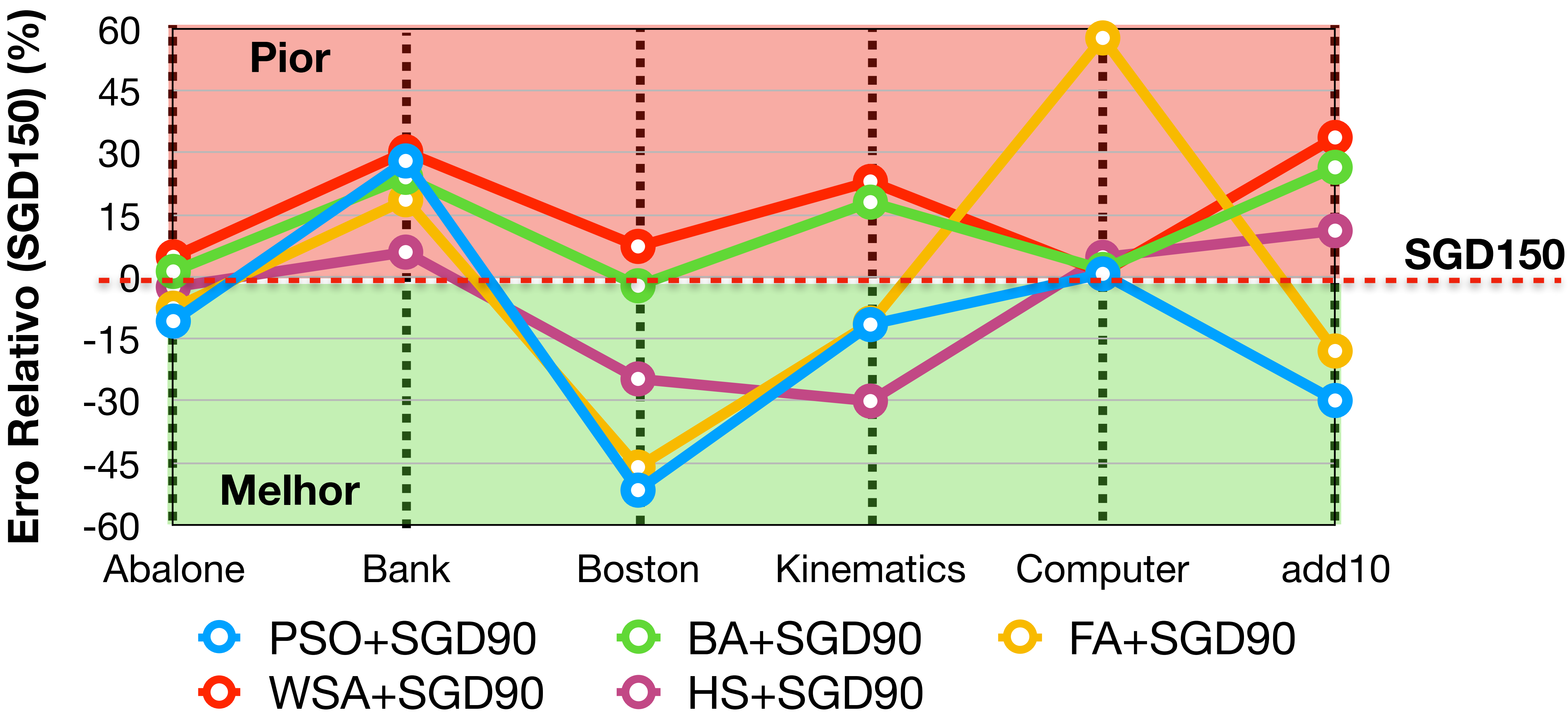
Resultados

	Mean Squared Error (Mean of 30 iterations)					
	Abalone	**Bank**	**Boston**	**Kinematics**	**Computer**	**add10**
SGD200	5.753459	0.1299616	99.67435	0.1500777	7.89021	5.2173023
SGD150	5.5822	0.16364995	104.866486	0.17441125	7.716035	5.3457494
SGD100	5.861112	0.1916133	111.10657	0.1919048	17.278103	6.9409275
PSO+SGD90	4.982256	0.2274761	50.66765	0.1542164	7.7706504	3.7434628
BA+SGD90	5.655402	0.2155763	102.58357	0.2128790	7.855001	7.2727294
FA+SGD90	5.1570597	0.2011744	56.51917	0.1548593	18.30432	4.384577
WSA+SGD90	5.8652616	0.2347182	113.18724	0.2495655	7.7908316	8.067269
HS+SGD90	5.442809	0.1740435	78.90492	0.1174007	8.097605	6.017419

Resultados Relativos

Diminuição do Erro | Aumento do Erro

Relative Mean Squared Error (%) (SGD150)						
	Abalone	Bank	Boston	Kinematics	Computer	add10
SGD150	0.	0.	0.	0.	0.	0.
PSO+SGD90	-10.75	+28.05	-51.68	-11.58	+0.7	-29.97
BA+SGD90	+1.29	+24.08	-2.18	+18.07	+1.77	+26.5
FA+SGD90	-7.62	+18.65	-46.1	-11.21	+57.84	-17.98
WSA+SGD90	+4.83	+30.27	+7.35	+23.08	+0.96	+33.74
HS+SGD90	-2.5	+5.97	-24.75	-30.11	+4.71	+11.16

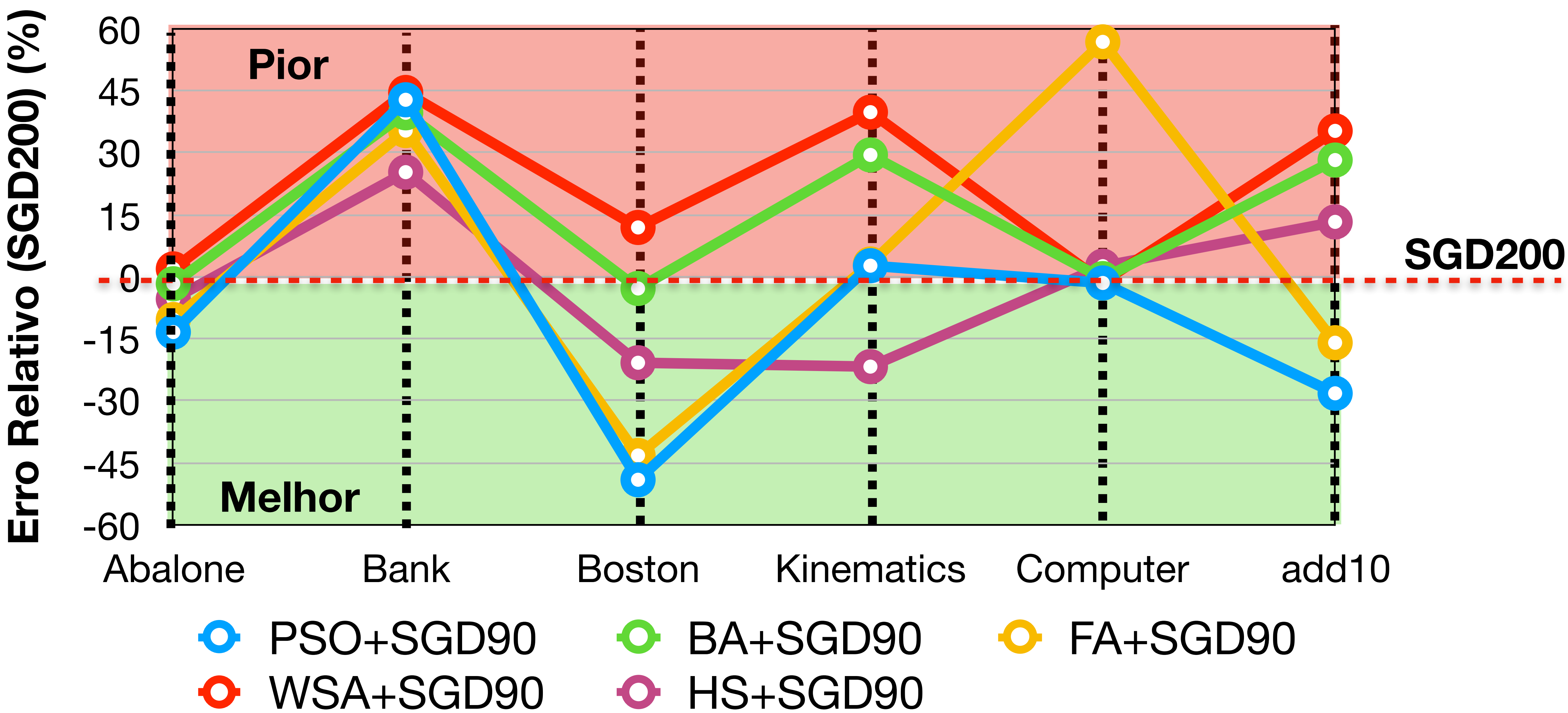


Resultados Relativos

Diminuição do Erro

Aumento do Erro

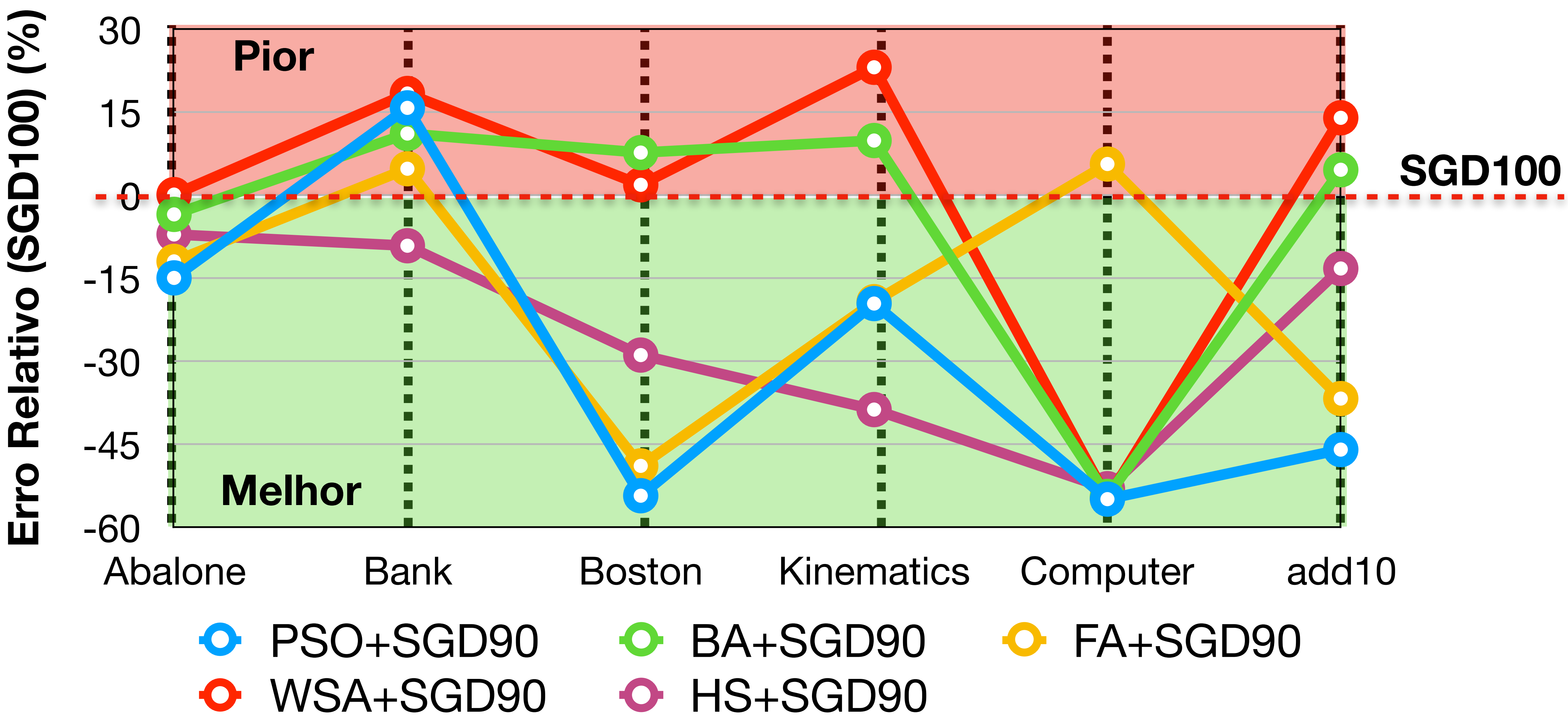
	Relative Mean Squared Error (SGD200) (%)					
	Abalone	Bank	Boston	Kinematics	Computer	add10
SGD200	0.	0.	0.	0.	0.	0.
PSO+SGD90	-13.40	+42.86	-49.17	+2.68	-1.51	-28.24
BA+SGD90	-1.7	+39.71	-2.84	+29.5	-0.45	+28.26
FA+SGD90	-10.36	+35.40	-43.29	+3.09	+56.89	-15.98
WSA+SGD90	+1.9	+44.63	+11.94	+39.86	-1.26	+35.33
HS+SGD90	-5.4	+25.33	-20.83	-21.77	+2.56	+13.29



Resultados Relativos

Diminuição do Erro | Aumento do Erro

	Relative Mean Squared Error (SGD100) (%)					
	Abalone	Bank	Boston	Kinematics	Computer	add10
SGD100	0.	0.	0.	0.	0.	0.
PSO+SGD	-14,994	+15.76	-54,4	-19,64	-55,02	-46.06
BA+SGD	-3,51	+11.11	-7,67	+9.85	-54,53	+4.56
FA+SGD	-12,01	+4.75	-49	-19,3	+5.6	-36.83
WSA+SGD	+0.0707	+18.36	+1.84	+23.10	-54,9	+13.96
HS+SGD	-7,14	-9,18	-28,96	-38,82	-53,13	-13.30



Diminuição do Erro   Aumento do Erro

	Relative Mean Squared Error (SGD100) (%)					
	Abalone	Bank	Boston	Kinematics	Computer	add10
SGD100	0.	0.	0.	0.	0.	0.
PSO+SGD	-14,994	+15.76	-54,4	-19,64	-55,02	-46.06
BA+SGD	-3,51	+11.11	-7,67	+9.85	-54,53	+4.56
FA+SGD	-12,01	+4.75	-49	-19,3	+5.6	-36.83
WSA+SGD	+0.0707	+18.36	+1.84	+23.10	-54,9	+13.96
HS+SGD	-7,14	-9,18	-28,96	-38,82	-53,13	-13.30

	Relative Mean Squared Error (SGD200) (%)					
	Abalone	Bank	Boston	Kinematics	Computer	add10
SGD200	0.	0.	0.	0.	0.	0.
PSO+SGD90	-13.40	+42.86	-49.17	+2.68	-1.51	-28.24
BA+SGD90	-1.7	+39.71	-2.84	+29.5	-0.45	+28.26
FA+SGD90	-10.36	+35.40	-43.29	+3.09	+56.89	-15.98
WSA+SGD90	+1.9	+44.63	+11.94	+39.86	-1.26	+35.33
HS+SGD90	-5.4	+25.33	-20.83	-21.77	+2.56	+13.29

Referências

- [1] Marc Toussaint. Optimization Course SS 13 U Stuttgart. https://ipvs.informatik.uni-stuttgart.de/mlr/marc/teaching/13-Optimization/.
- [2] MOHAGHEGI, S. et al. A comparison of pso and backpropagation for training rbf neural networks for identification of a power system with statcom. In: IEEE. Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. [S.l.], 2005. p. 381–384
- [3] GUDISE, V. G.; VENAYAGAMOORTHY, G. K. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: IEEE. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706). [S.l.], 2003. p. 110–117.
- [4] Yinyu Ye. Zero-Order and First-Order Optimization Algorithms I. https://web.stanford.edu/class/msande311/lecture10.pdf
- [5] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [https://archive.ics.uci.edu/ml/datasets/abalone]. Irvine, CA: University of California, School of Information and Computer Science.
- [6] Delve. "Bank family of Datasets". Disponível em: https://www.cs.toronto.edu/~delve/data/bank/desc.html. Acessado em Dezembro de 2018.
- [7] Delve. "boston dataset". Disponível em: https://www.cs.toronto.edu/~delve/data/boston/desc.html. Acessado em Dezembro de 2018.
- [8] Delve. "Kin family of datasets". Disponível em: https://www.cs.toronto.edu/~delve/data/kin/desc.html. Acessado em Dezembro de 2018.
- [9] Delve. "comp-activ dataset". Disponível em: https://www.cs.toronto.edu/~delve/data/comp-activ/desc.html. Acessado em Dezembro de 2018.
- [10] Delve. "add10 dataset". Disponível em: https://www.cs.toronto.edu/~delve/data/add10/desc.html. Acessado em Dezembro de 2018.
- [11] Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *evolutionary computation, 2001. Proceedings of the 2001 Congress on* (Vol. 1, pp. 81-86). IEEE.
- [12] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67.
- [13] Yang, X. S., & He, X. (2013). Firefly algorithm: recent advances and applications. *arXiv preprint arXiv:1308.3898*.
- [14] Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *simulation*, *76*(2), 60-68.
- [15] Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74). Springer, Berlin, Heidelberg.