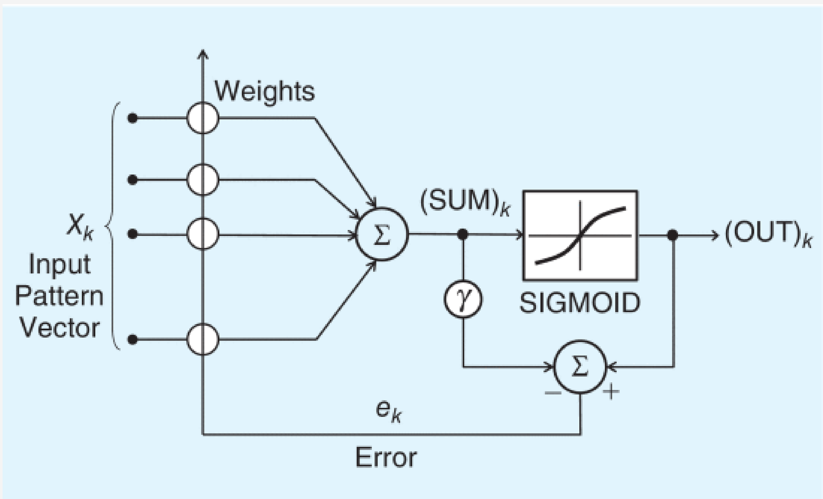
% LMS: A stochastic gradient descent algorithm inspired by neurobiology

Abhipray Sahoo, Jake Kaplan, Stephane Remigereau

Stanford University 2019

Motivation

Dr. Bernard Widrow proposed a linear neuron model, Hebbian-LMS, that learns via % LMS.

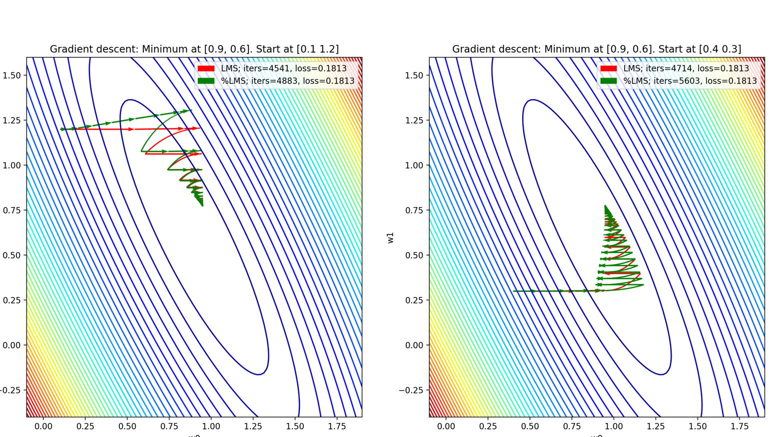


Inputs $\to$ firing rate of pre-synaptic neurons
Weights $\to$ number of neuroreceptors at the dendrite of a living neuron
Output $\to$ firing rate
Training $\to \min \mathbb{E}[e_k^2]$

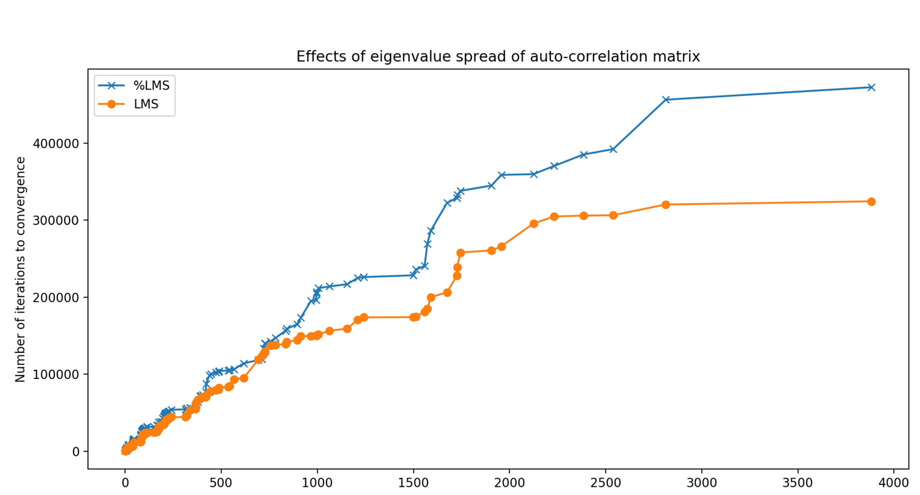
Concentration of neuroreceptors in living neurons changes via synaptic scaling; weights of the neuron model $\uparrow$ or $\downarrow$ as a multiplicative factor instead of additively leading to the %-LMS update rule. We explore properties, performance and extensions.

Convergence

Does it converge?



How does it do with different shapes of the quadratic cost function?



Observation: Cost Function

A neuron updates its weight θ to minimize:

$$\min d(\theta_{k+1}, \theta_k) + \alpha J(\theta_{k+1}) \quad (1)$$

Set the gradient w.r.t θ_{k+1} to 0 after assuming

$$\nabla_{\theta_{k+1}} J(\theta_{k+1}) \sim \nabla_{\theta_k} J(\theta_k)$$

$$\nabla_{\theta_{k+1}} d(\theta_{k+1}, \theta_k) + \alpha \nabla_{\theta_k} J(\theta_k) = 0 \quad (2)$$

$$J(\theta_k) = \text{MSE} = ||y_k - \theta_k^T x_k||_2^2 \quad (3)$$

LMS $\to d(\theta_{k+1}, \theta_k) = ||\theta_{k+1} - \theta_k||_2^2$

%LMS $\to d(\theta_{k+1}, \theta_k) = \Sigma_{j=1}^{|\theta|} \frac{(\theta_{k+1,j} - \theta_{k,j})^2}{\theta_{k,j}}$

%LMS updates to minimize relative change in weights. Big weights adapt faster.

Observation: Non-negativity

Weights have to be non-negative because

1. Negative weights grow more and more negative.
2. Zero weights stop changing due to multiplication.

To prevent the weight becoming negative, the learning rate is bounded:

$$\alpha_k \leq \frac{1}{\epsilon_k x_k} \quad \forall k \quad (4)$$

% LMS

$$\theta_{k+1} = \theta_k + \alpha \epsilon_k x_k \circ \theta_k$$
$$= (1 + \alpha \epsilon_k x_k) \circ \theta_k$$

$x_k \to$ input, $\theta_k \to$ weight vector, $\alpha \to$ learning rate, $\epsilon_k \to$ the error $(y_k - \theta^T x_k)$, $\circ \to$ element-wise product

Generalized Algorithm and Variance

Extend % LMS for negative weights:

$$\theta_{k+1} = (1 + \alpha \epsilon_k x_k \text{sign}(\theta_k)) \circ (\theta_k) \quad (5)$$

$$\text{sign}(\theta_k) = \begin{cases} 1 & \theta_k \geq 0 \\ -1 & \theta_k < 0 \end{cases} \quad (6)$$

Add noise to prevent convergence to 0:

$$\theta_{k+1} = (1 + \alpha \epsilon_k x_k \text{sign}(\theta_k) + \vec{g}(\theta_k)) \circ (\theta_k) \quad (7)$$

$$g_i(\theta_k) = \begin{cases} z \sim \mathcal{N}(0, \varepsilon^2) & -\varepsilon^2 < (\theta_k)_i < \varepsilon^2 \\ 0 & \text{else} \end{cases} \quad (8)$$

1. Might not converge to small θ^* if ε too large.
2. False convergence to 0 if ε too small.

Solution: Set $\varepsilon_k^2 = |\alpha \nabla J(\theta_k)| = |\alpha \epsilon_k x_k|$.
Results: All % LMS models showed convergence improvement; $\varepsilon^2 = \nabla J$ minimized loss deviation.

Variance Experimental Results

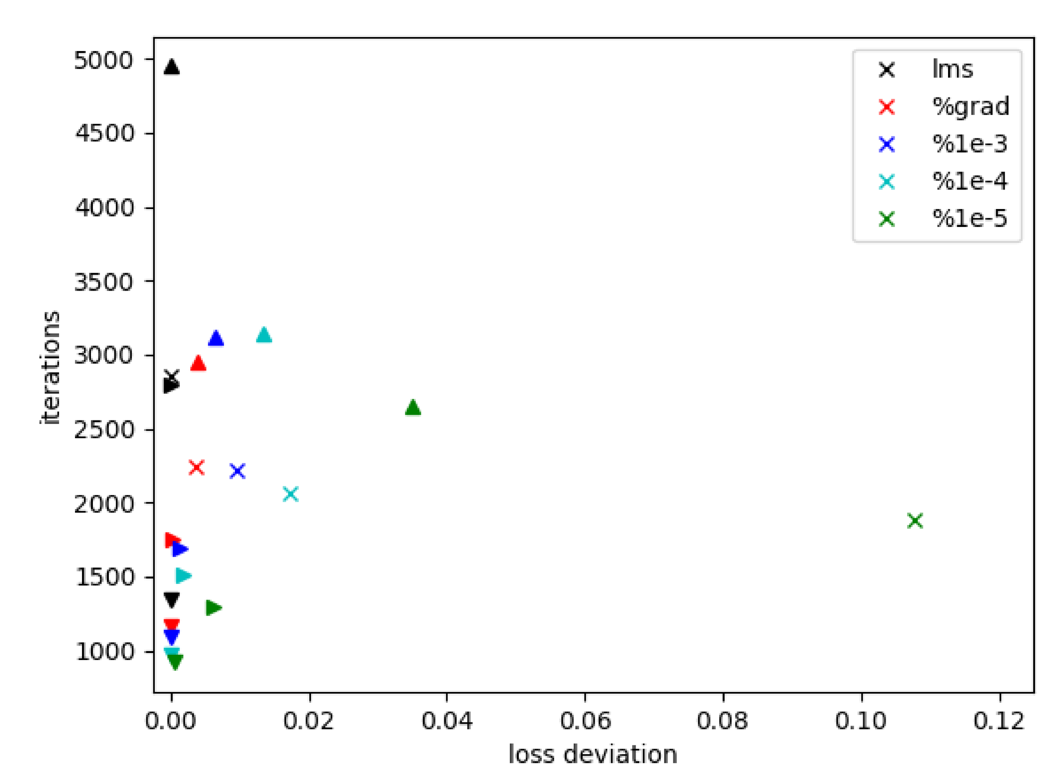


Figure 1: 100 trial full-batch Poisson regression comparing deviation in loss (from LMS) against convergence rate of standard LMS to % LMS with values of $\varepsilon^2 \in \{\nabla J, 10^{-3}, 10^{-4}, 10^{-5}\}$. $\times, \triangleright, \triangle, \triangledown$ denote mean, median, and upper/lower quartiles.

Applications of General % LMS

Logistic Regression Classification

	LMS	%LMS	Newton
Initialization	0	Random	0
Iterations	13442	6032	941
Accuracy	83%	83%	83%

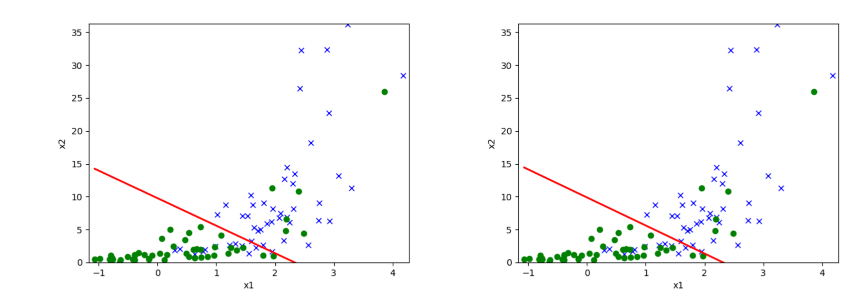


Figure 2: LMS Classification vs %LMS Classification

Neural Network MNIST Classification

`Architecture:` single hidden layer with 150 sigmoid units, softmax output layer
`Training:` epochs=20, batch size=20, learning rate=0.1, cross-entropy loss, 50K MNIST examples
`Testing:` 10K MNIST examples.

	LMS	%LMS
Training loss	0.10528	0.14121
Test loss	0.3178	0.3279
Accuracy	92.21%	91.02%

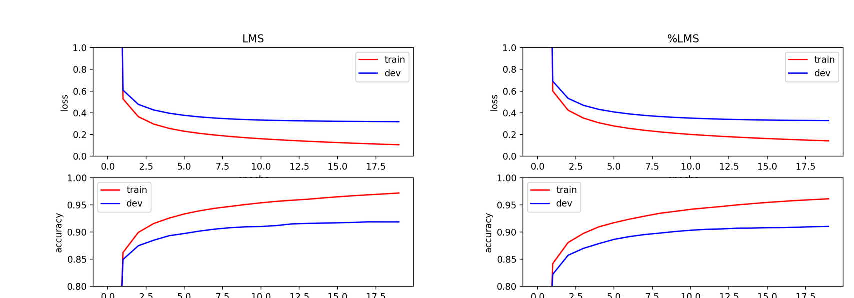


Figure 3: MNIST neural network training with LMS vs %LMS

Acknowledgements

We would like to thank Dr. Bernard Widrow for sharing his ideas and his guidance.

Widrow, B., Kim, Y., & Park, D. (2015). The Hebbian-LMS learning algorithm. IEEE Computational intelligence magazine

Next Steps

1. Derive convergence properties and learning curve analytically
2. Compare the performance of % LMS when classifying different distributions (so far only Guassian and Poisson distributions analyzed)