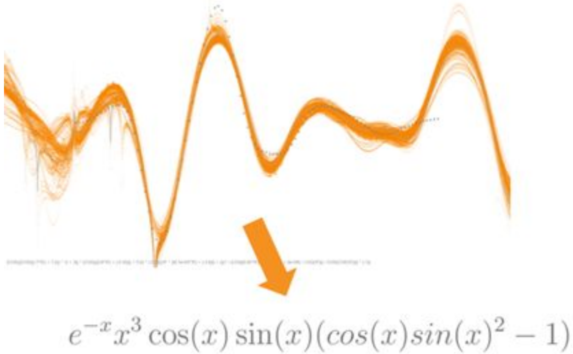
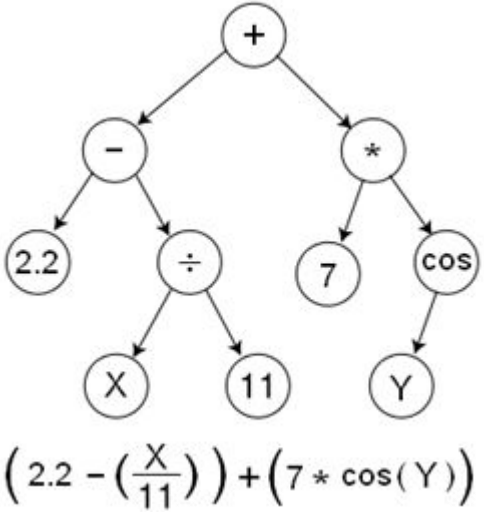
Meta-learning for Multi-task Symbolic Regression

6.883 Final Project
12/8/20
Sam Kim and Ileana Rugina

Symbolic Regression





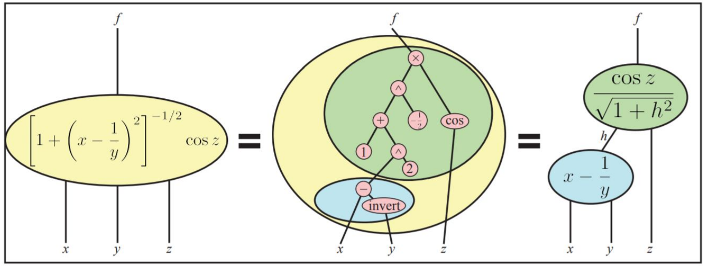


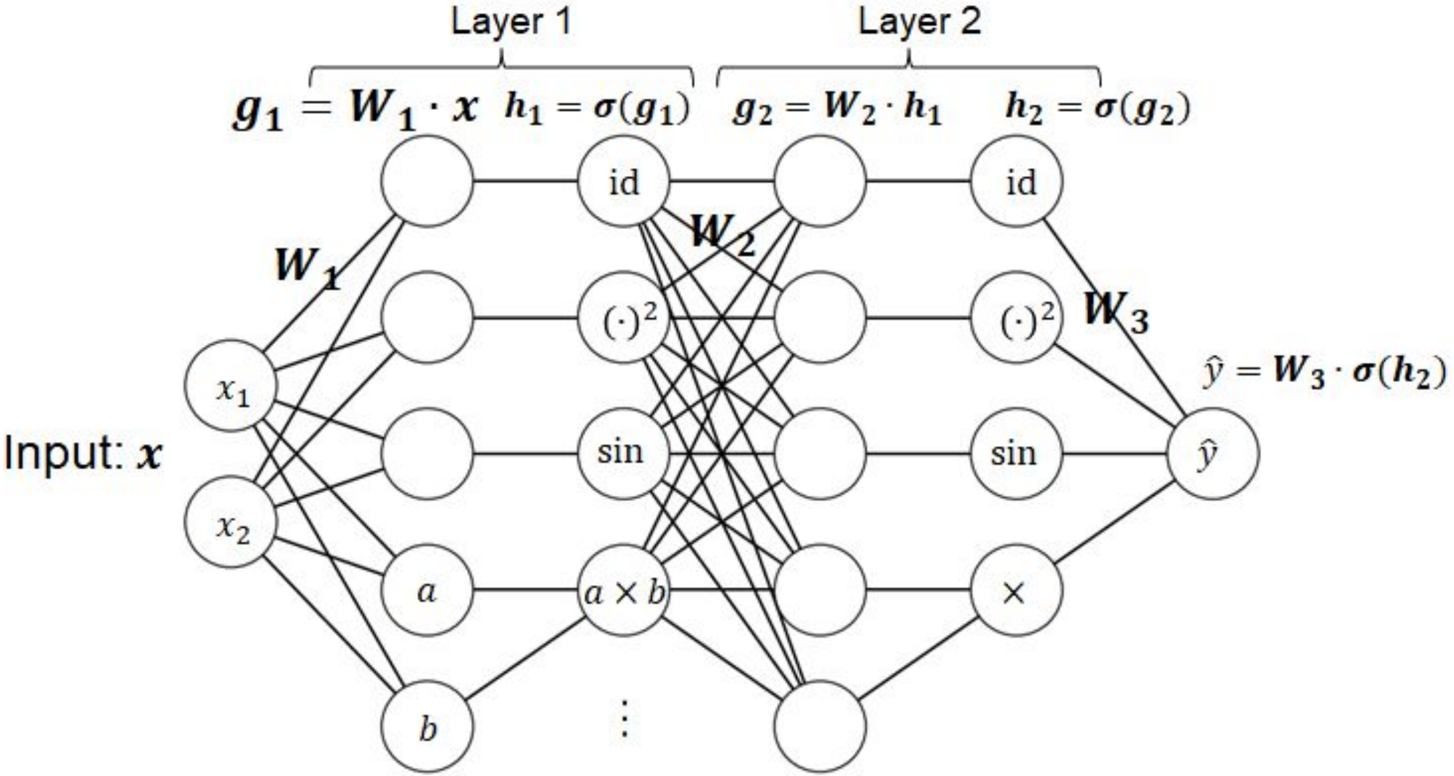
Figure 2: All functions can be represented as tree graphs whose nodes represent a set of basic functions (middle panel). Using a neural network trained to fit a mystery function (left panel), our algorithm seeks a decomposition of this function into others with fewer input variables (right panel), in this case of the form $f(x,y,z)=g[h(x,y),z]$.

Offers interpretable results that can extrapolate

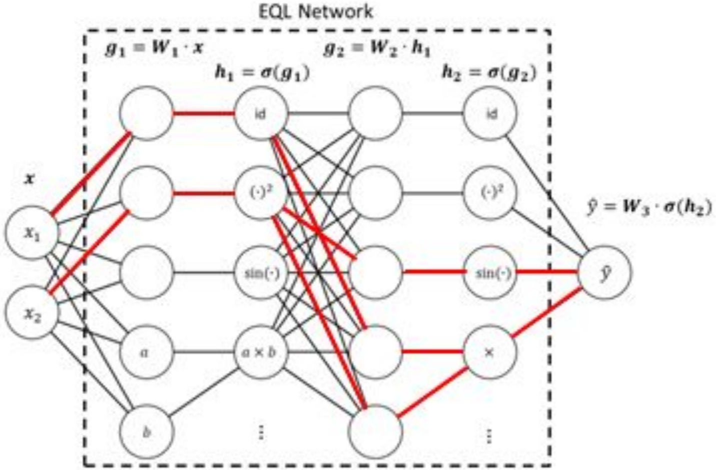
Typically implemented using genetic programming

AI Feynman 2.0
Uses neural networks to discover symmetries and simplify the problem

Equation Learner (EQL) Network



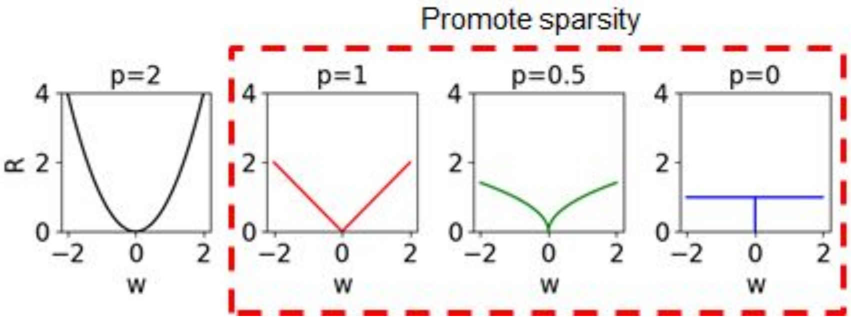
Key Ingredient: Sparsity



Sparsity through regularization

$$\text{Loss function} = \text{MSE} + R$$

$$R = \sum |w|^p$$

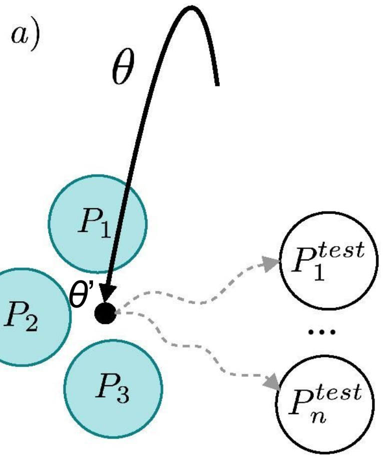


Meta-Learning Methods:

- ## Joint Training

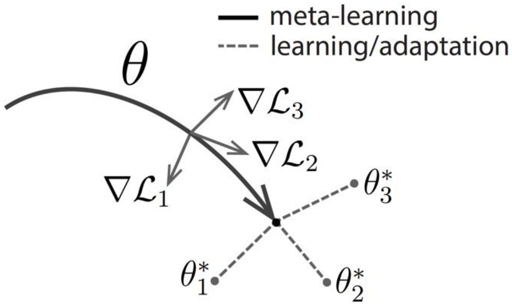
Train on all functions “simultaneously” - pass in data from each function sequentially and perform parameter updates.

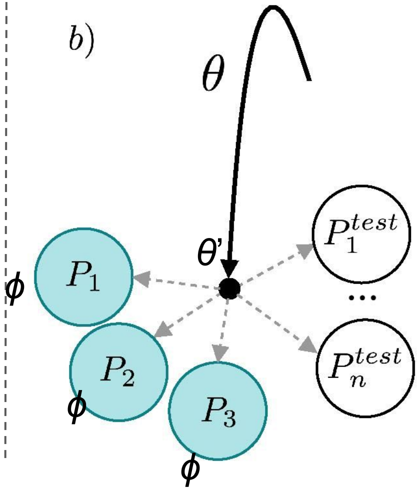
<u>No</u> second order terms or inductive bias towards good initialization



- ## Model-Agnostic Meta-Learning (MAML)

Find a good initialization by “fine-tuning” on each function during training to get θ’. Propagates second-order derivatives from θ’ to the initial θ.





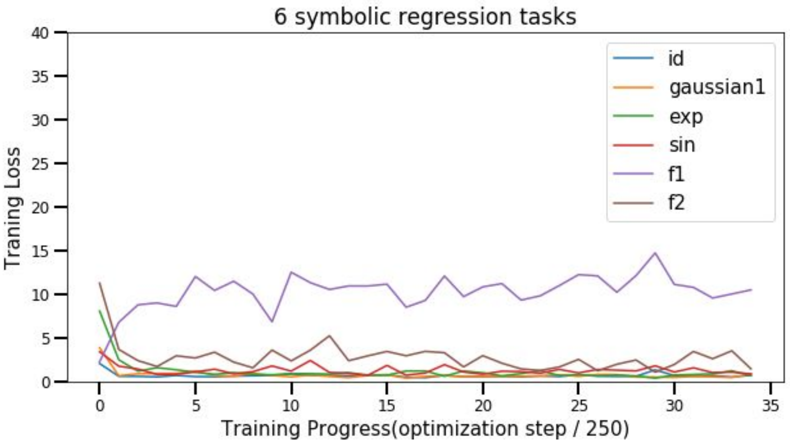
Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

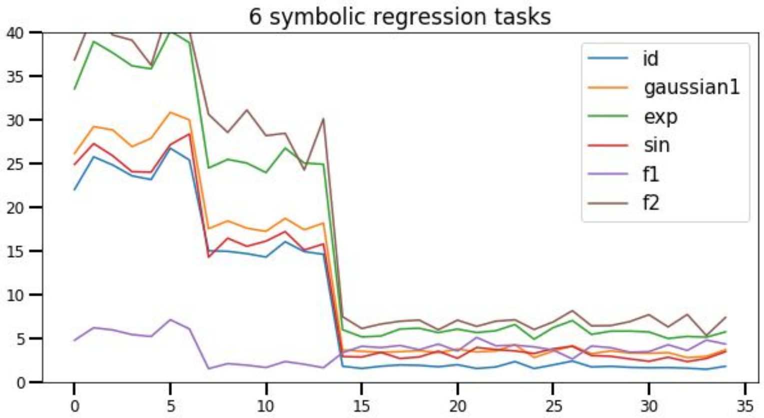
```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

Initial Results: Regularization Method

L_0 sparsity regularization

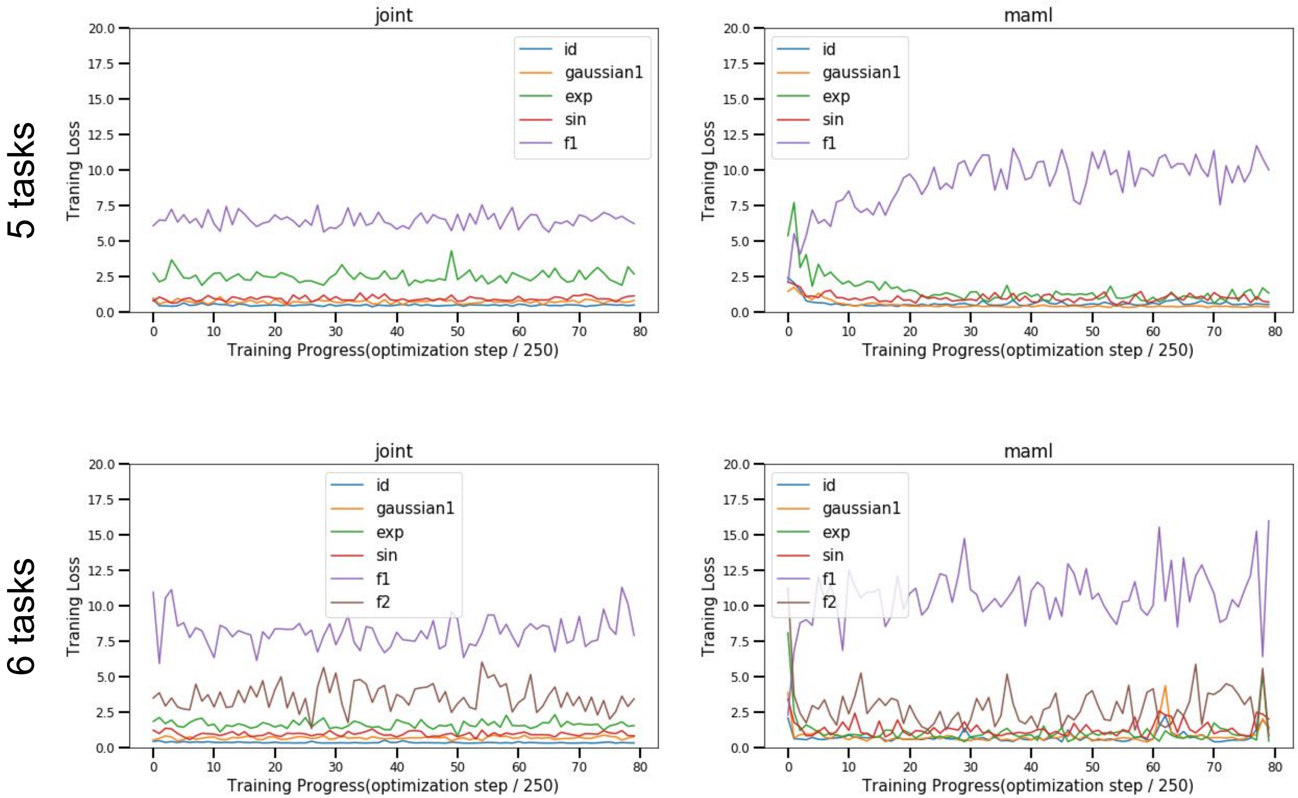


$L_{0.5}$ sparsity regularization



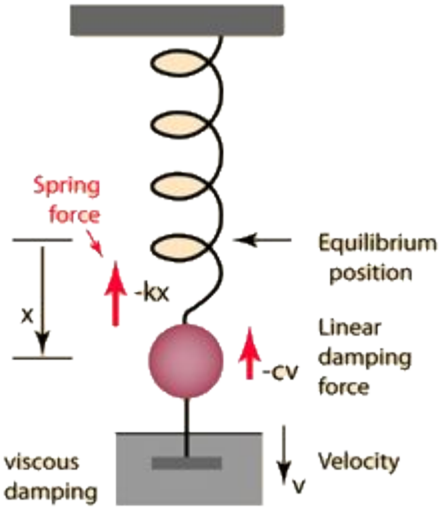
Initial Results: Method Comparison

ID	formula
"id"	$f(x) = x$
"gaussian1"	$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2 * \pi}}$
"exp"	$f(x) = \exp(x)$
"sin"	$f(x) = \sin x$
"f1"	$f(x) = x \cdot \sin x - 3$
"f2"	$f(x) = x^2 + 3 \cdot x + 1$



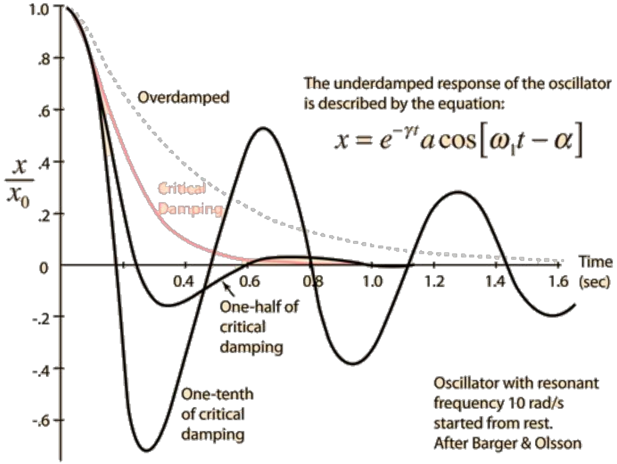
L_0 regularization

Distribution of Functions: Damped Harmonic Oscillator



$$F = -kx - c\frac{\mathrm{d}x}{\mathrm{d}t} = m\frac{\mathrm{d}^2x}{\mathrm{d}t^2},$$

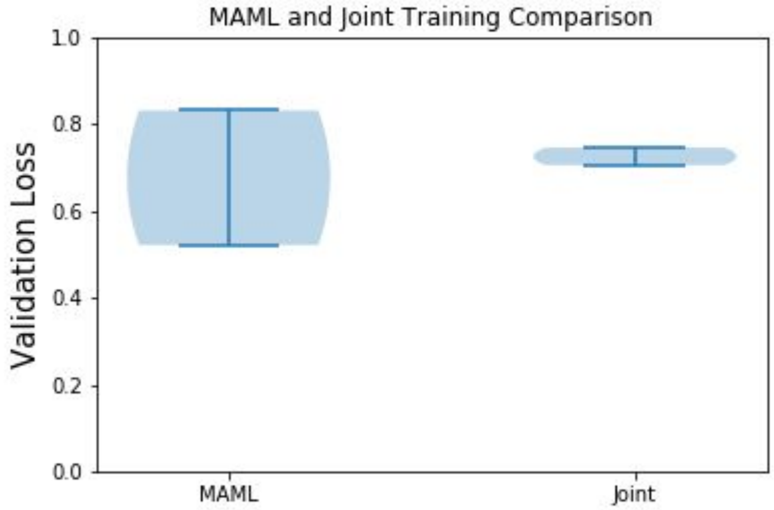
$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + 2\zeta\omega_0\frac{\mathrm{d}x}{\mathrm{d}t} + \omega_0^2 x = 0,$$



Results: Drawing from the Function Distribution

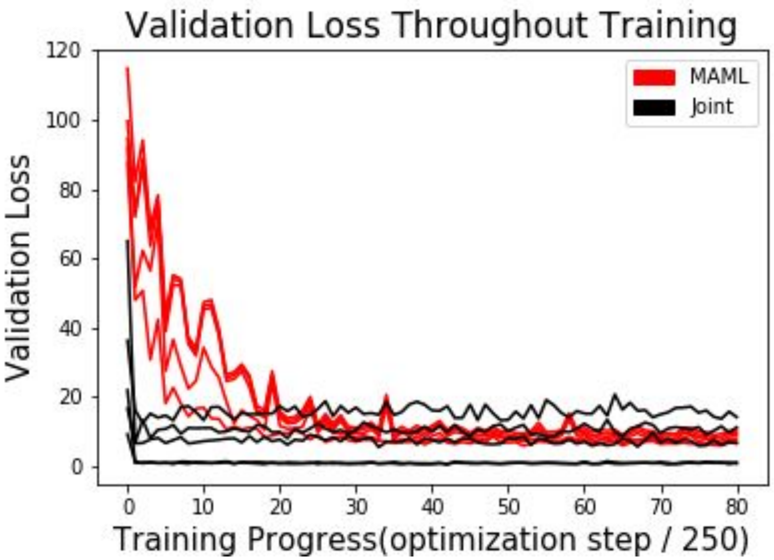
Function generation process:
Sample *C* ~ Bernoulli(0.5)

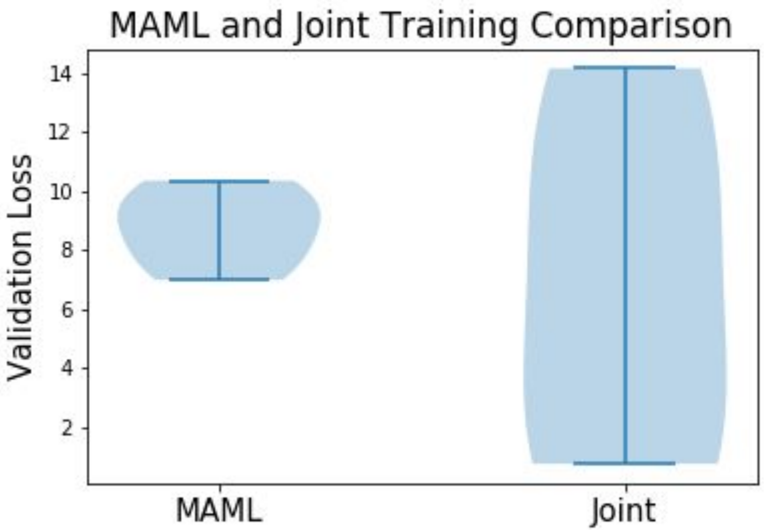
- If C = 0
  - Sample ω ~ Unif(0.5, 2)*2π
  - Sample ϕ ~ Unif(0, 2π)
  - Function:
    - f(x) = sin(ωx+ϕ)
- Else
  - Sample a,b ~ Unif(-1, 1)
  - Function:
    - f(x) = a□e^x + b□e^x



MAML	Joint Training
0.68	0.72

Out of Domain Experiments





MAML	Joint Training
8.76	6.75

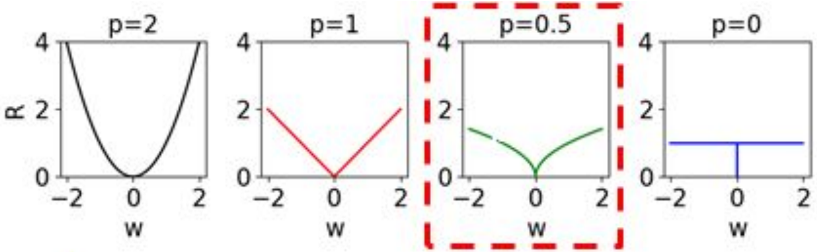
Conclusion

Demonstrate meta-learning on the EQL network for symbolic regression

When functions are drawn from the same distribution (which is often the case in certain fields of science and engineering), meta-learning improves performance and generalizability of the EQL network

MAML outperforms joint training on average for test functions drawn from same distribution as training

Smoothed L0.5 Sparsity



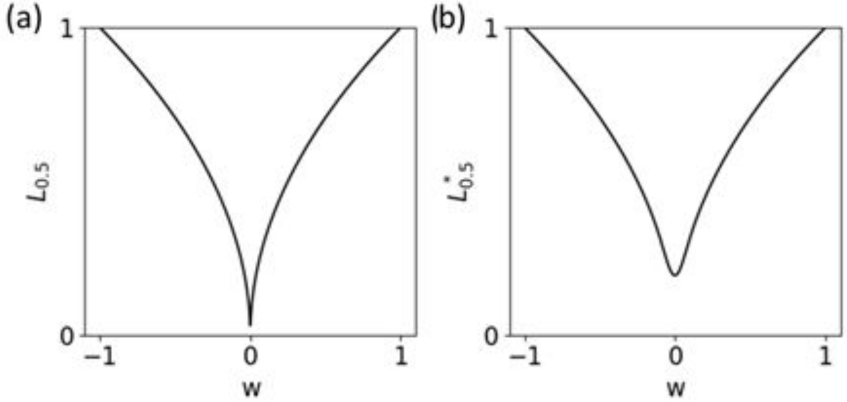
$L_{0.5}$

- Promotes sparsity stronger than L_1
- Downsides:
    - Non-convex
    - Infinite gradient

Smoothed $L_{0.5}$ regularization

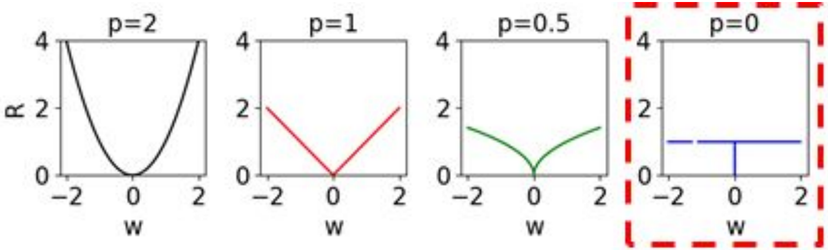
- Avoids infinite gradient

$$L^*_{0.5}(w) = \begin{cases} |w|^{1/2} & w \geq a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{1/2} & w < a \end{cases}$$



Fan, Qinweiet, et al.. "Convergence of online gradient method for feedforward neural networks with smoothing L1/2 regularization penalty." *Neurocomputing* 131 (2014): 208-216.

Relaxed L0 Sparsity



L_0

- Promotes sparsity without penalizing magnitude
- Downside: non-differentiable

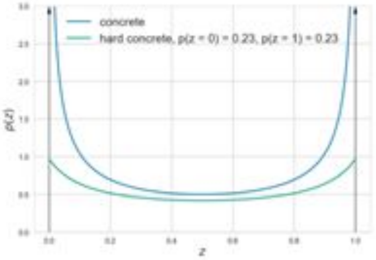
Relaxed L_0 regularization

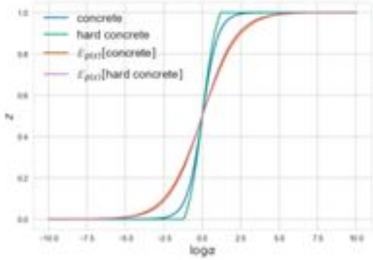
- Can calculate gradients for backprop

Reparameterize weights as

$$\boldsymbol{\Theta} = \widetilde{\boldsymbol{\Theta}} \odot \boldsymbol{z}$$

where $\boldsymbol{z}$ is a stochastic variable drawn from the Hard Concrete distribution





Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).