L10

Metrics

L1 and L2 Regularization

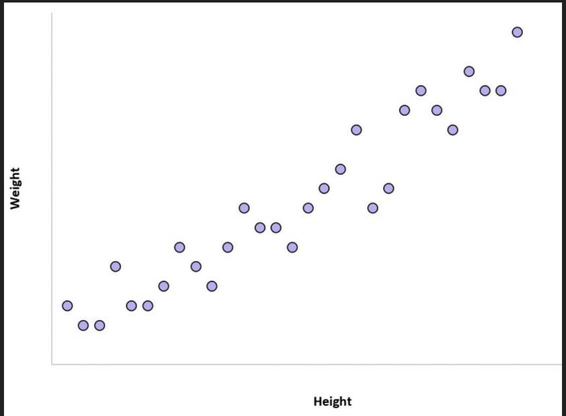
Vanishing / exploding gradients

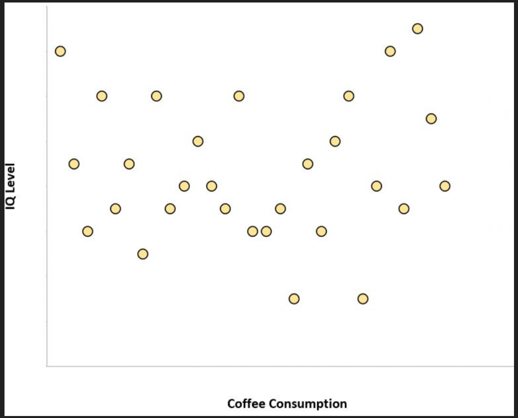
Activation functions

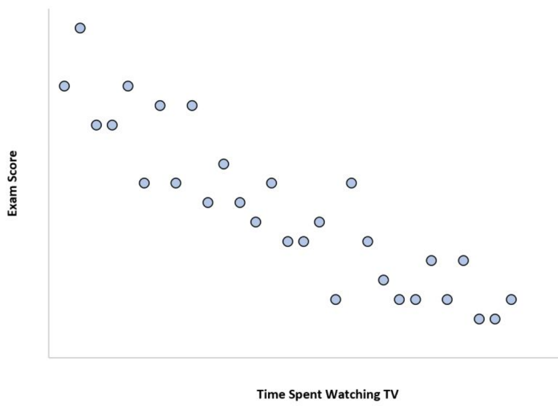
Weight initialization

Metrics https://scikit-learn.org/stable/modules/model_evaluation.html

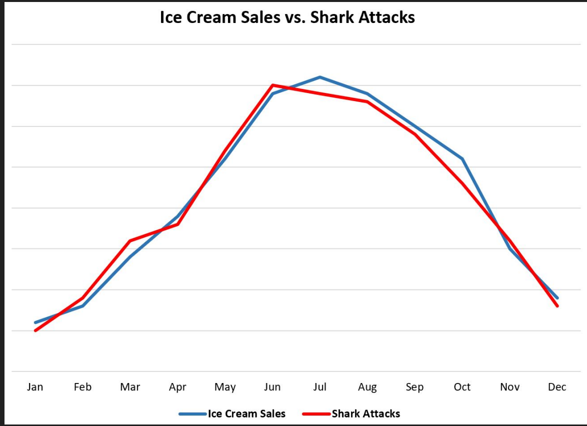
Correlation and causation

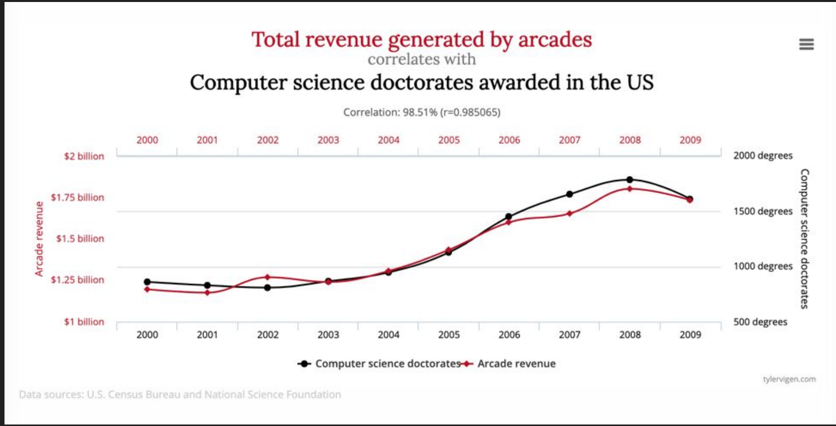


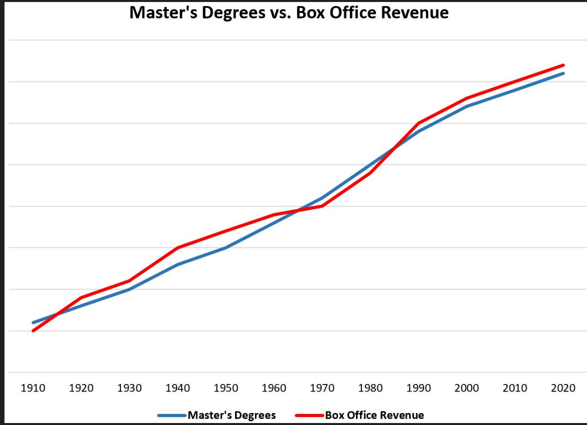


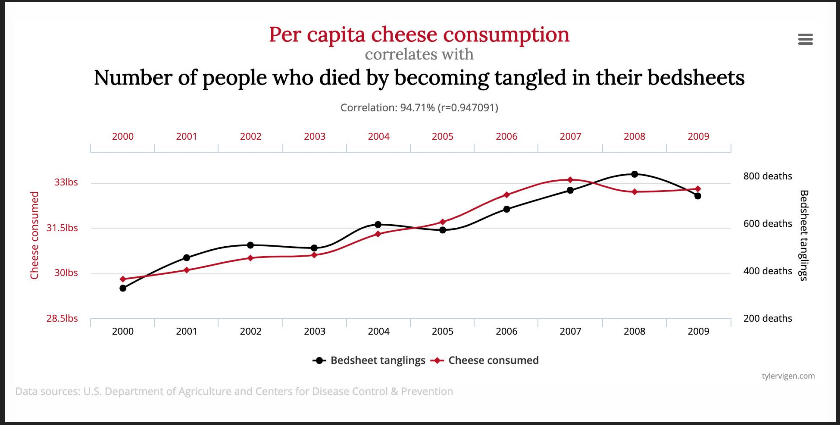


Correlation does not imply causation



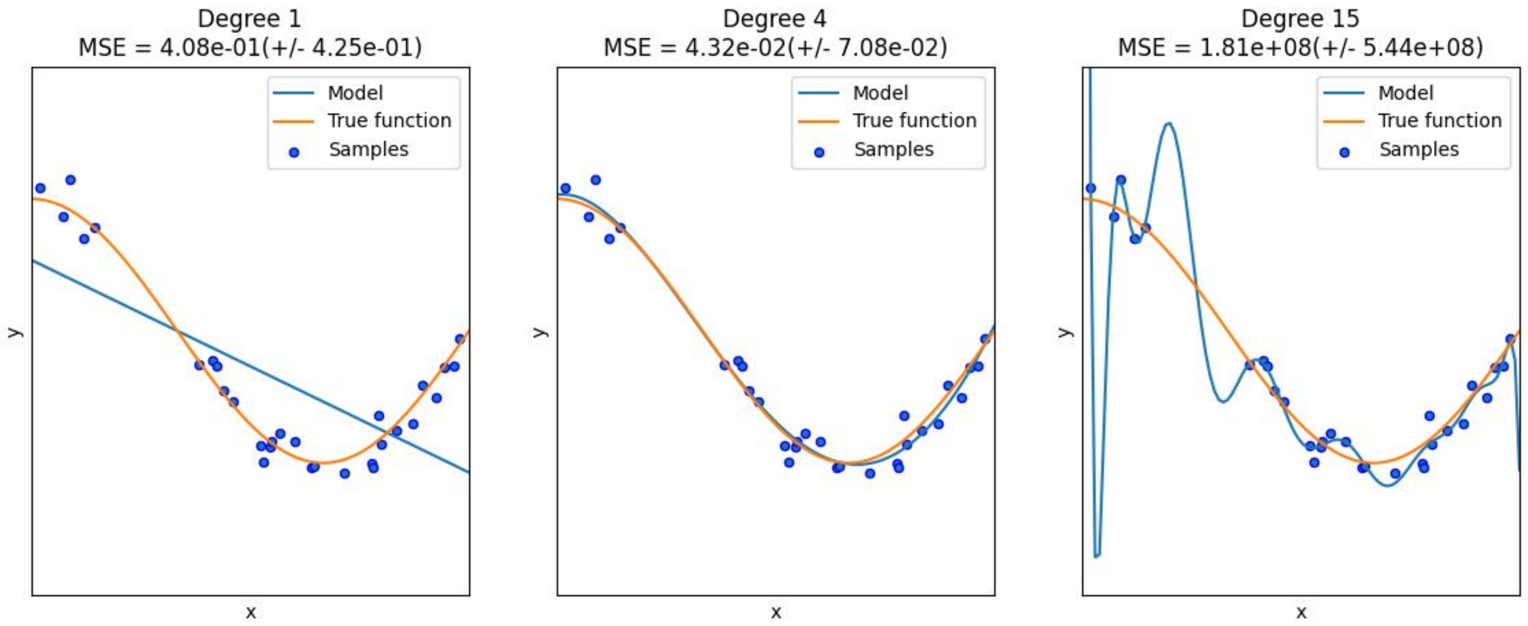




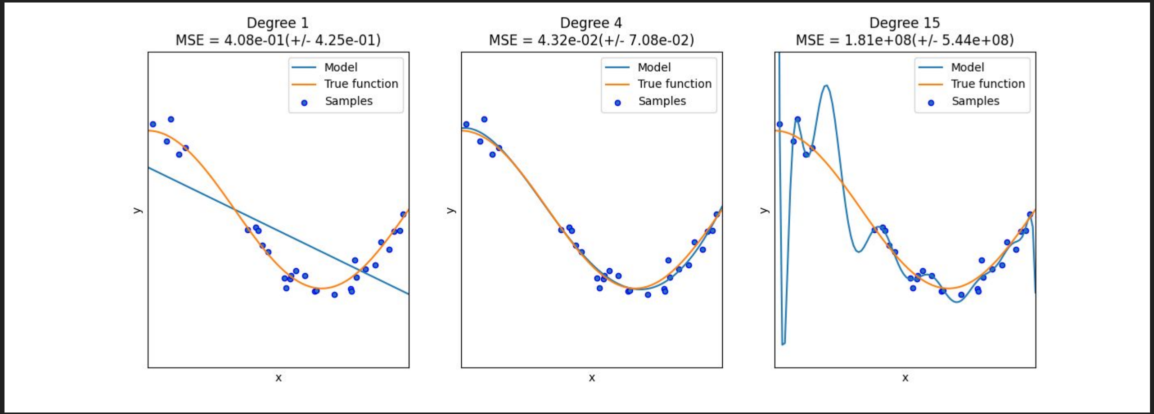


L1 and L2 Regularization

- Lower the weights



Why Lower Weights?



$$\widehat{y} = w_0 + w_1 x$$

$$\widehat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

$$\widehat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6 + w_7 x^7 + w_8 x^8 + w_9 x^9 + w_{10} x^{10} + w_{11} x^{11} + w_{12} x^{12} + w_{13} x^{13} + w_{14} x^{14} + w_{15} x^{15}$$

How to Lower Weights?

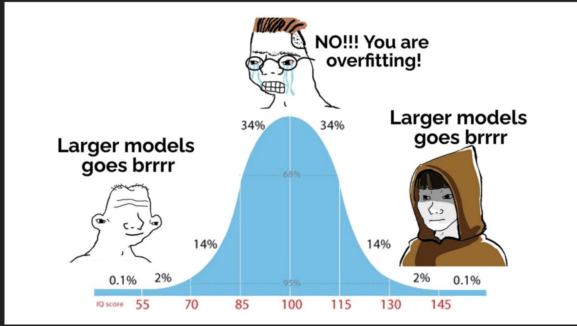
$$MSE = \sum_{i}^{N} (y_i - Wx_i)^2$$

$$MSE_{l1} = \sum_{i}^{N} (y_i - Wx_i)^2 + \lambda \sum_{j}^{M} |W_j|$$

$$MSE_{l2} = \sum_{i}^{N} (y_i - Wx_i)^2 + \lambda \sum_{j}^{M} W_j^2$$

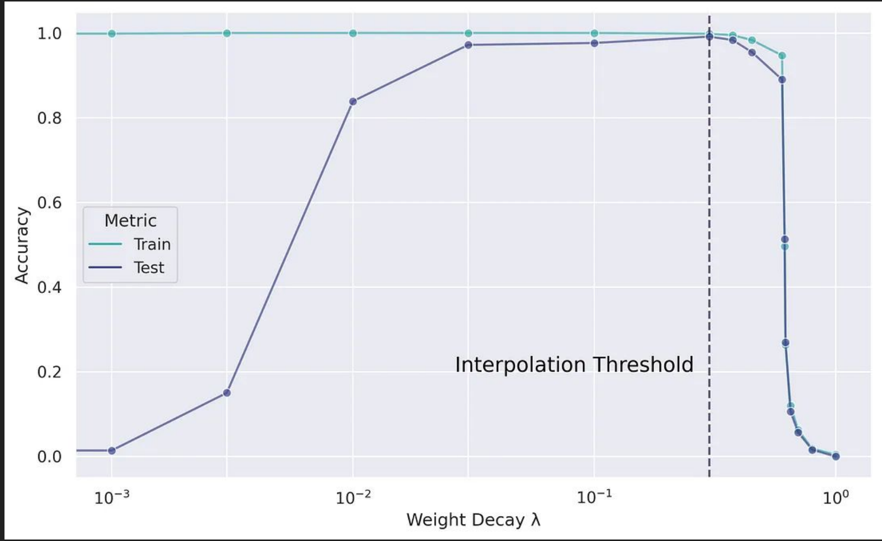
What if lambda=0, what if lambda=1, what if lambda=inf?

L1 and L2 Regularization



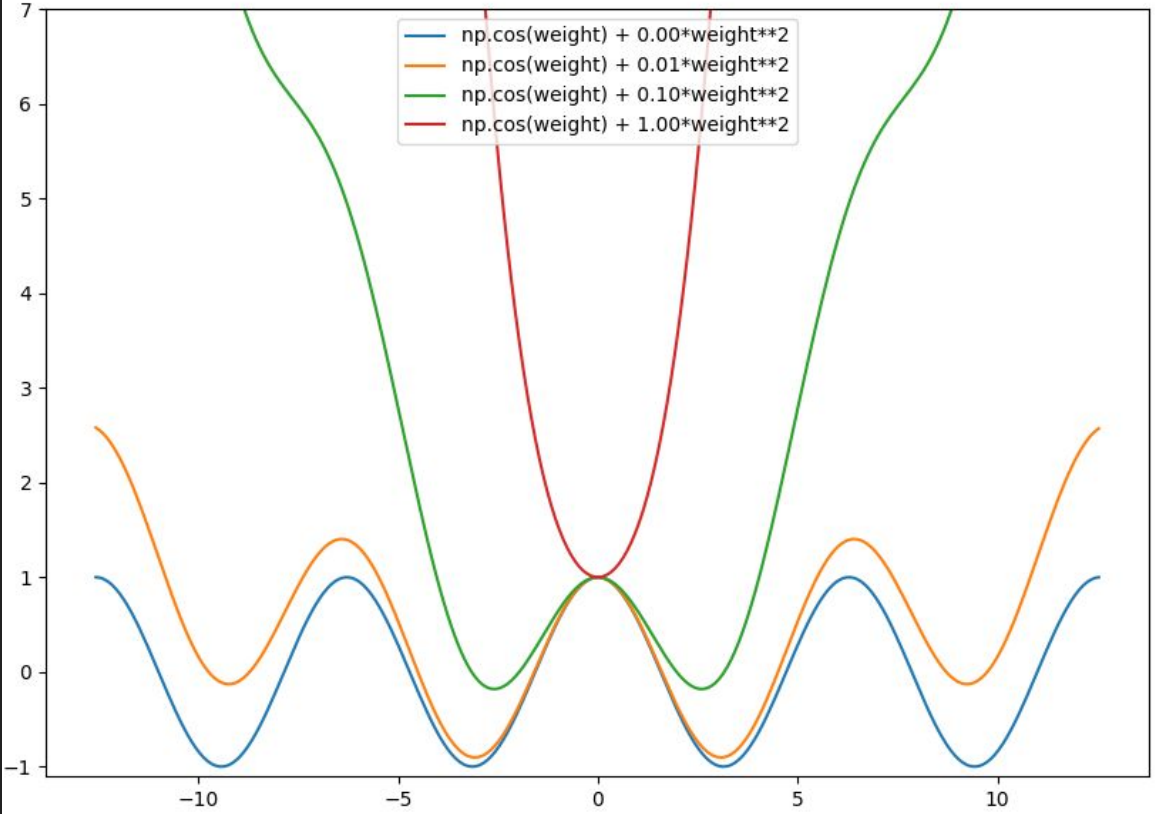
L1 Regularization == L1 Norm == Lasso Regression

L2 Regularization == L2 Norm == Ridge Regression == Weight Decay



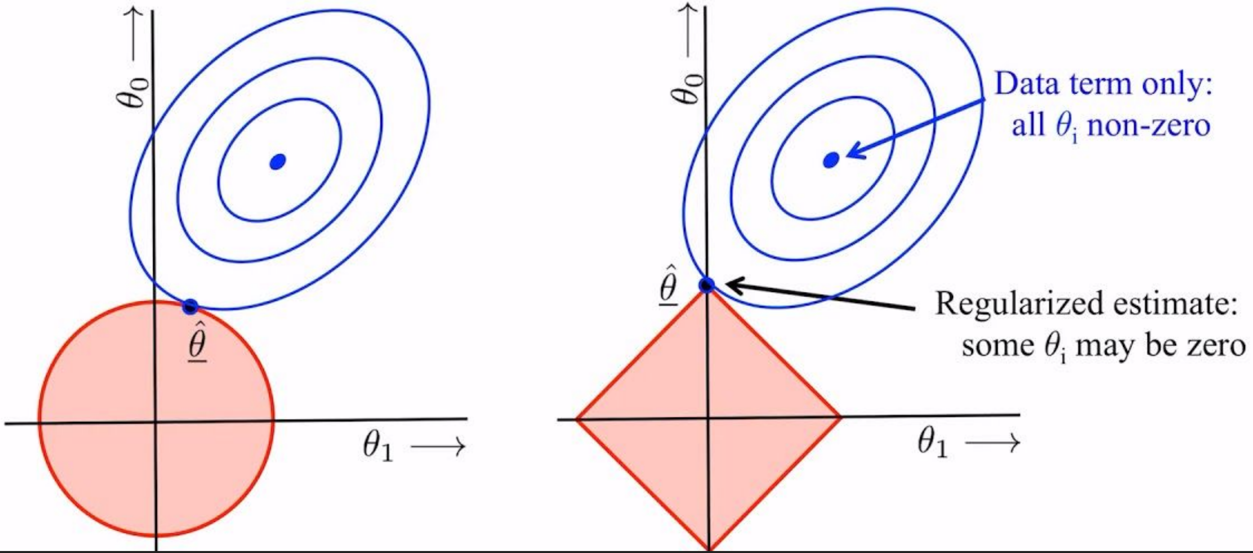
Ref: https://towardsdatascience.com/weight-decay-and-its-peculiar-effects-66e0aee3e7b8

L1 and L2 changes Loss function topology



L1 vs L2

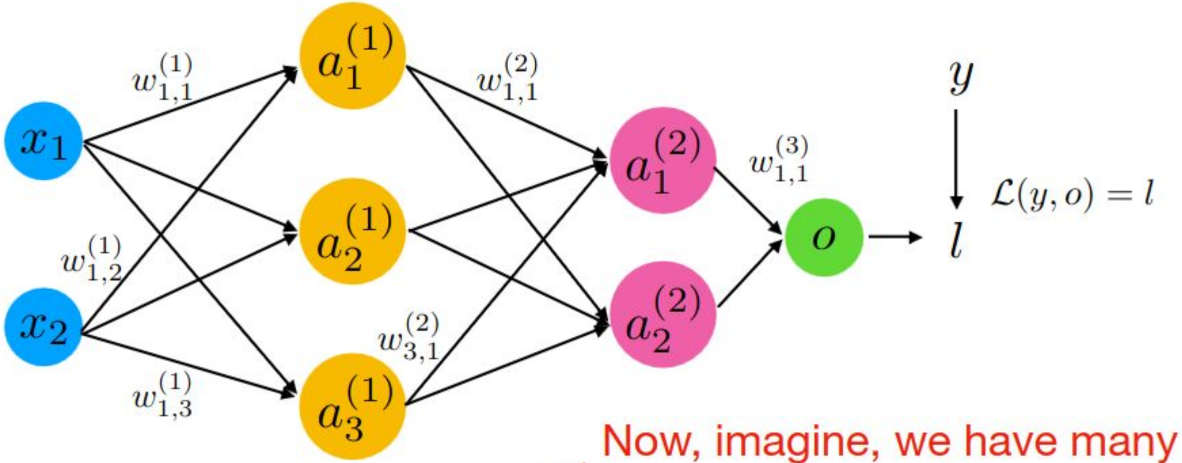
- L1 tends to generate sparser solutions than a quadratic regularizer



Go Find L2 in Pytorch!

https://pytorch.org/docs/stable/generated/torch.optim.SGD.html

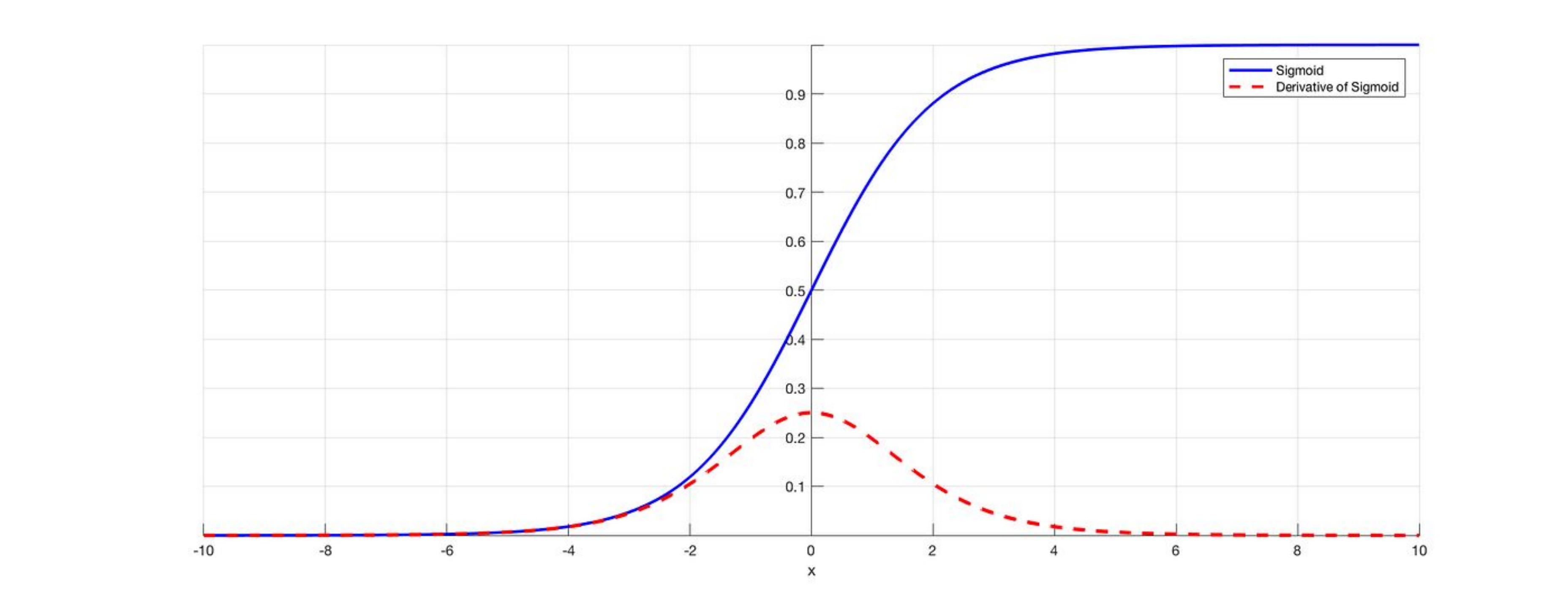
Vanishing / exploding gradients



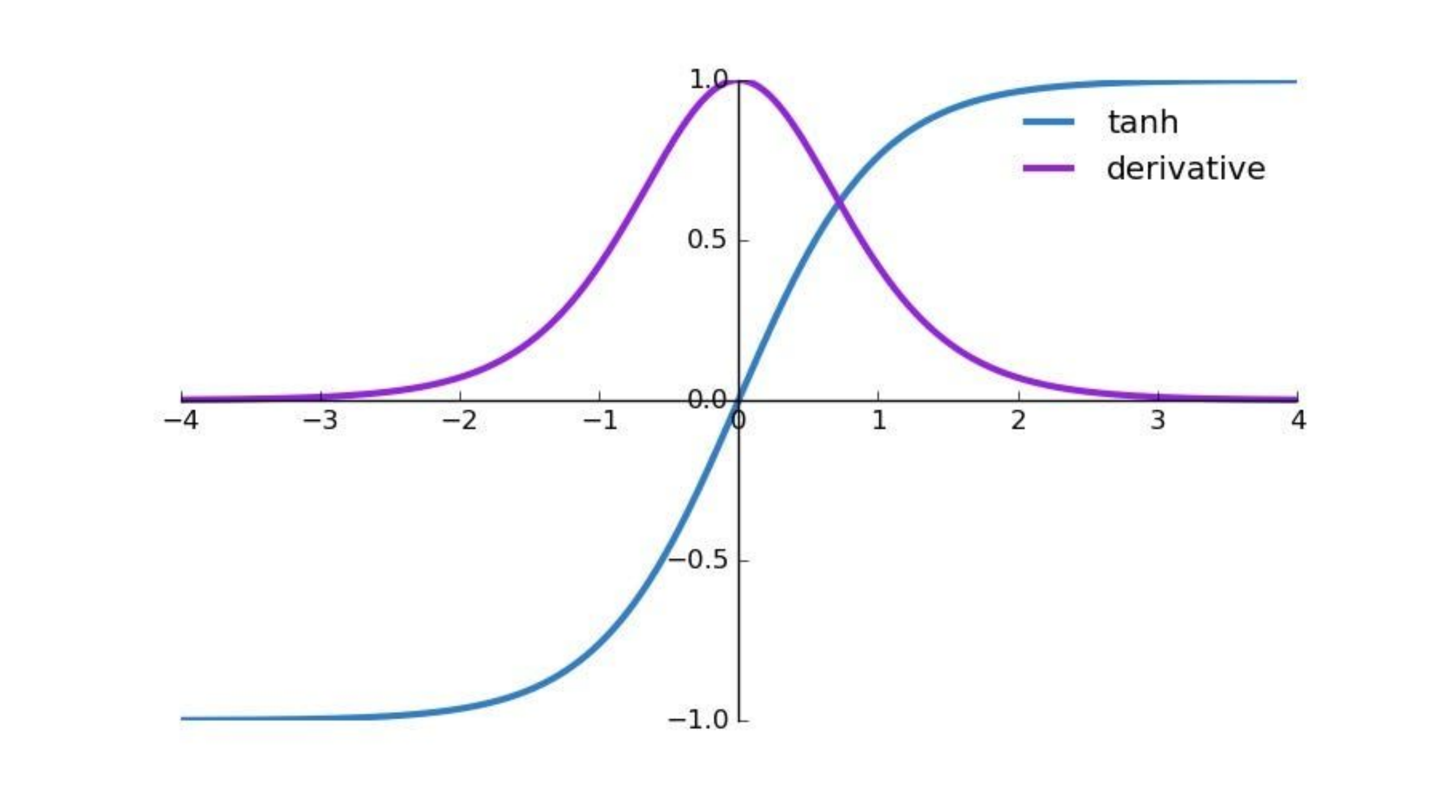
Now, imagine, we have many layers and sigmoid activations ...

$$\frac{\partial l}{\partial w_{1,1}^{(1)}} = \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_1^{(2)}} \cdot \boxed{\frac{\partial a_1^{(2)}}{\partial a_1^{(1)}}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}} + \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_2^{(2)}} \cdot \frac{\partial a_2^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}}$$

Vanishing / exploding gradients



tanh



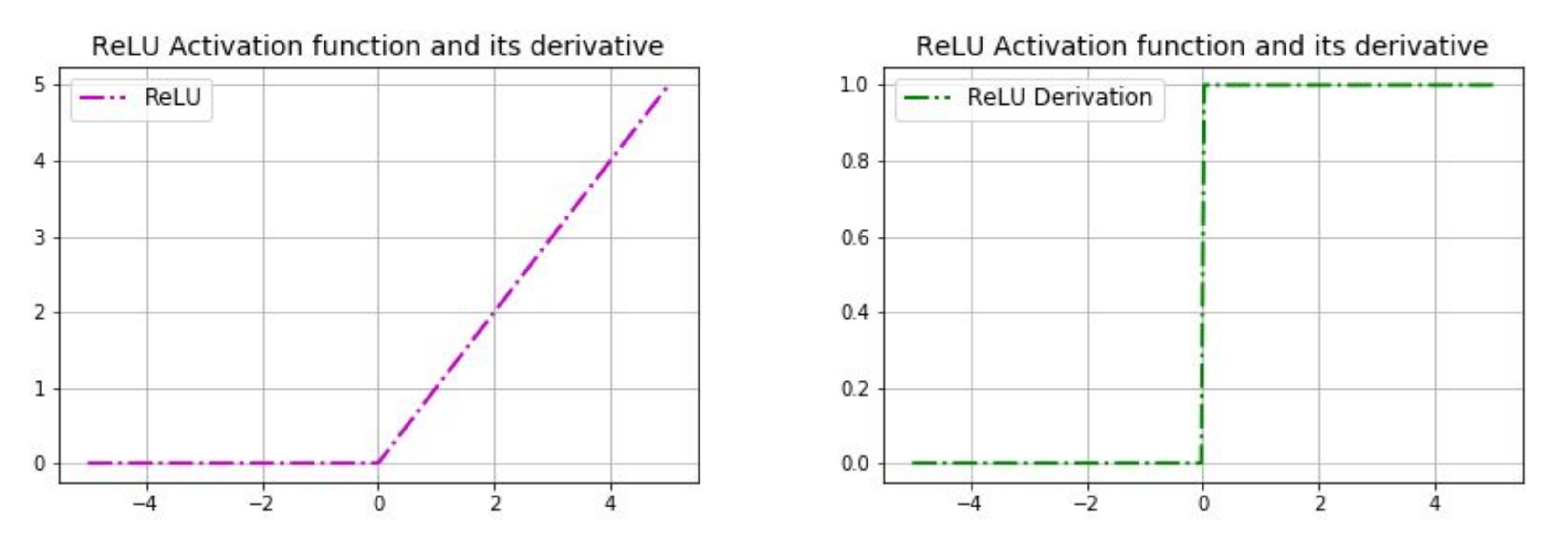
tanh

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

$$\frac{d}{dx}\tanh(x) = \frac{(e^{x} + e^{-x})(e^{x} + e^{-x}) - (e^{x} - e^{-x})(e^{x} - e^{-x})}{(e^{x} + e^{-x})^{2}}$$

$$= 1 - \frac{(e^{x} - e^{-x})^{2}}{(e^{x} + e^{-x})^{2}} = 1 - \tanh^{2}(x)$$

ReLU

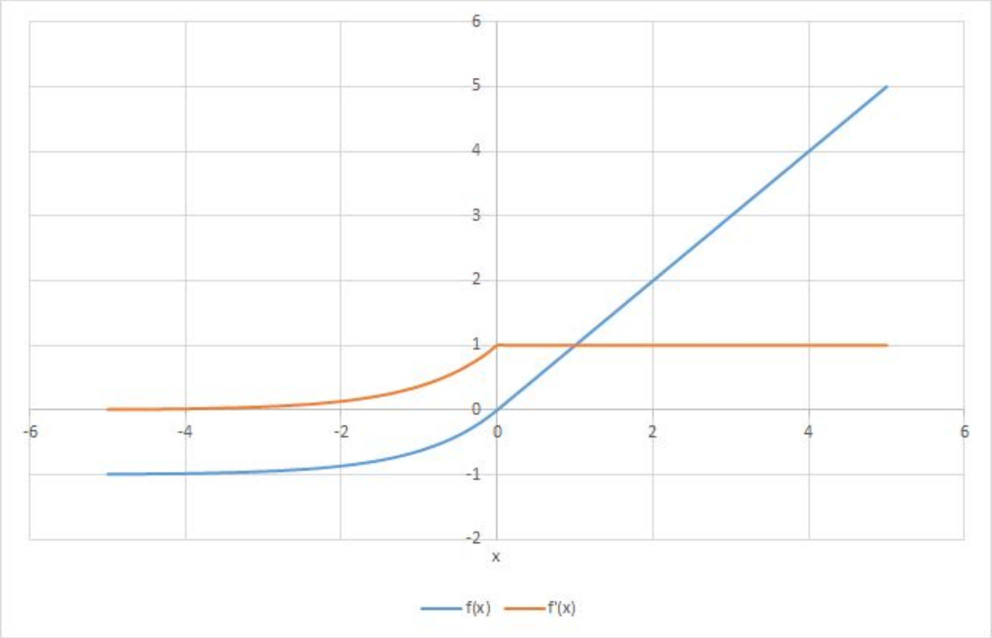


ReLU Activation function and its derivative
ReLU
ReLU Activation function and its derivative
ReLU Derivation

ELU

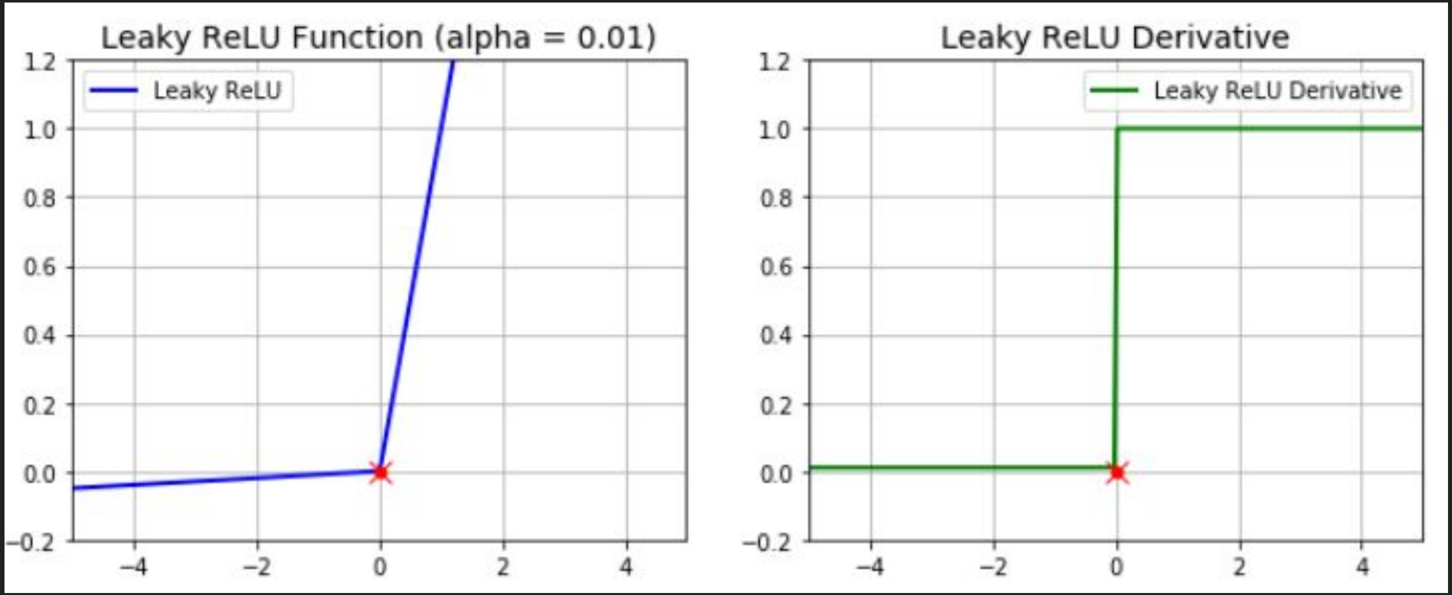
https://paperswithcode.com/method/elu

$$f(x) = x \text{ if } x > 0$$
$$\alpha(\exp(x) - 1) \text{ if } x \leq 0$$

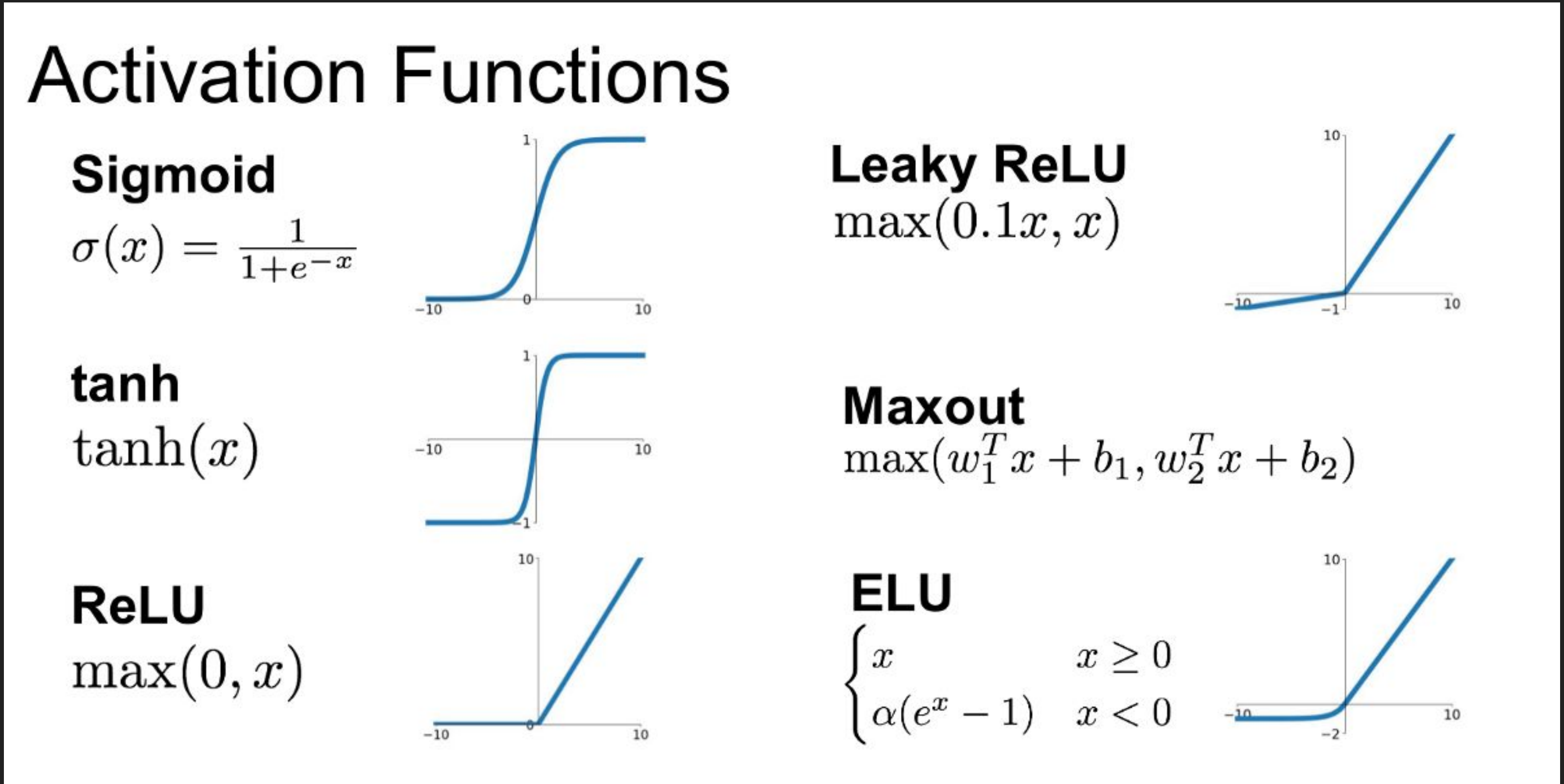


Leaky ReLU

https://paperswithcode.com/method/leaky-relu



List



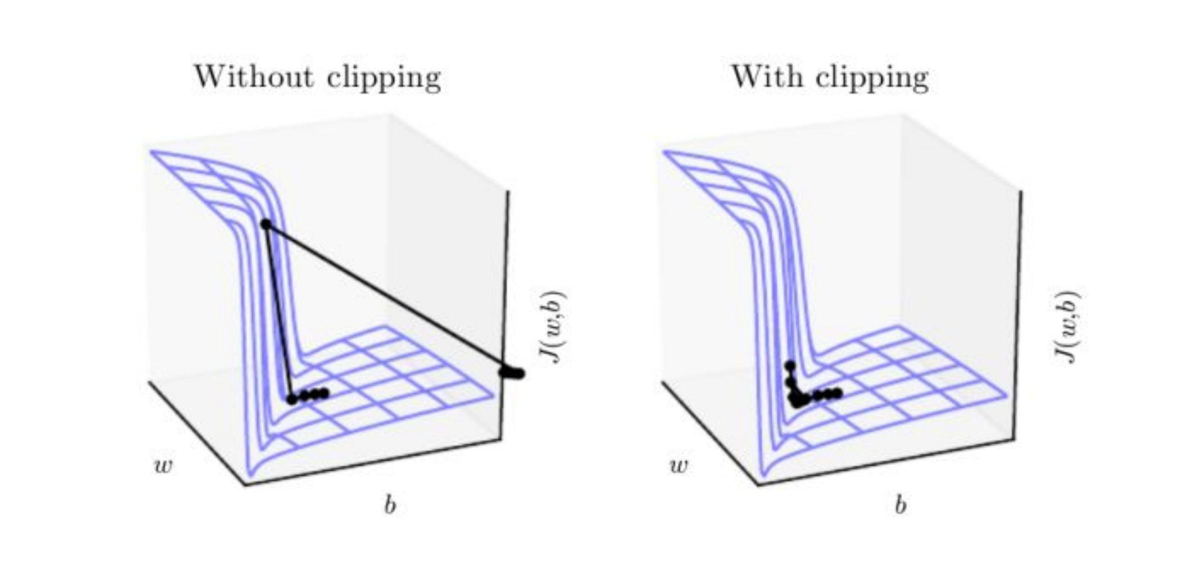
Activation Functions
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$
tanh
$\tanh(x)$
ReLU
$\max(0, x)$
Leaky ReLU
$\max(0.1x, x)$
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$
ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

Exploding Gradient. Gradient Clipping

https://paperswithcode.com/method/gradient-clipping

https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html

https://github.com/pytorch/pytorch/issues/309#issuecomment-327304962

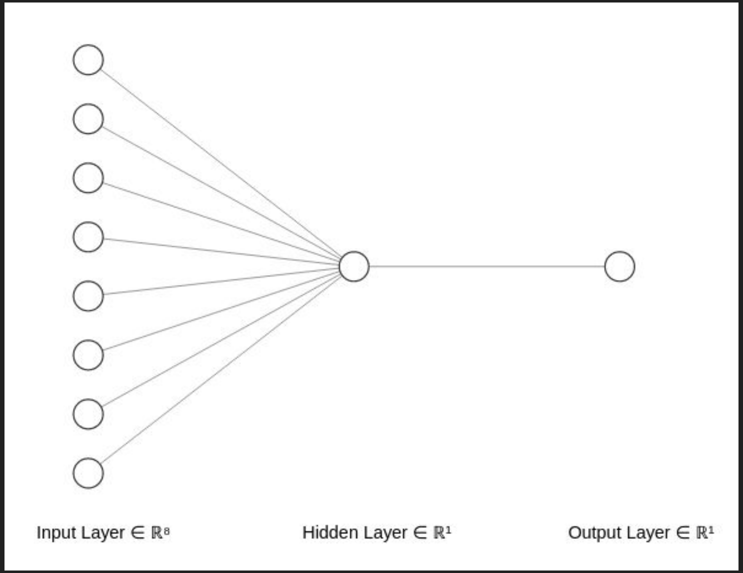


Proper Weight Initialization to Solve Vanishing Gradient

- Traditionally, we can initialize weights by sampling from a random uniform distribution in range [0, 1], or better, [-0.5, 0.5]
- Or, we could sample from a Gaussian distribution with mean 0 and small variance (e.g., 0.1 or 0.01)

$$W^{(l)} = \mathrm{np.random.normal}(loc = 0.0, scale = 1.0) \cdot 0.01$$

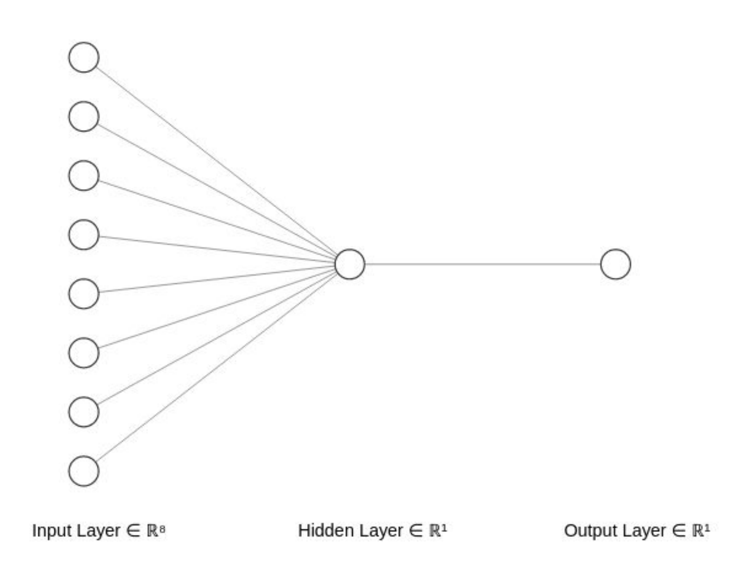
Problem with Traditional Weight Initialization



Ref: https://github.com/ashishpatel26/Tools-to-Design-or-Visualize-Architecture-of-Neural-Network

Problem with Traditional Weight Initialization

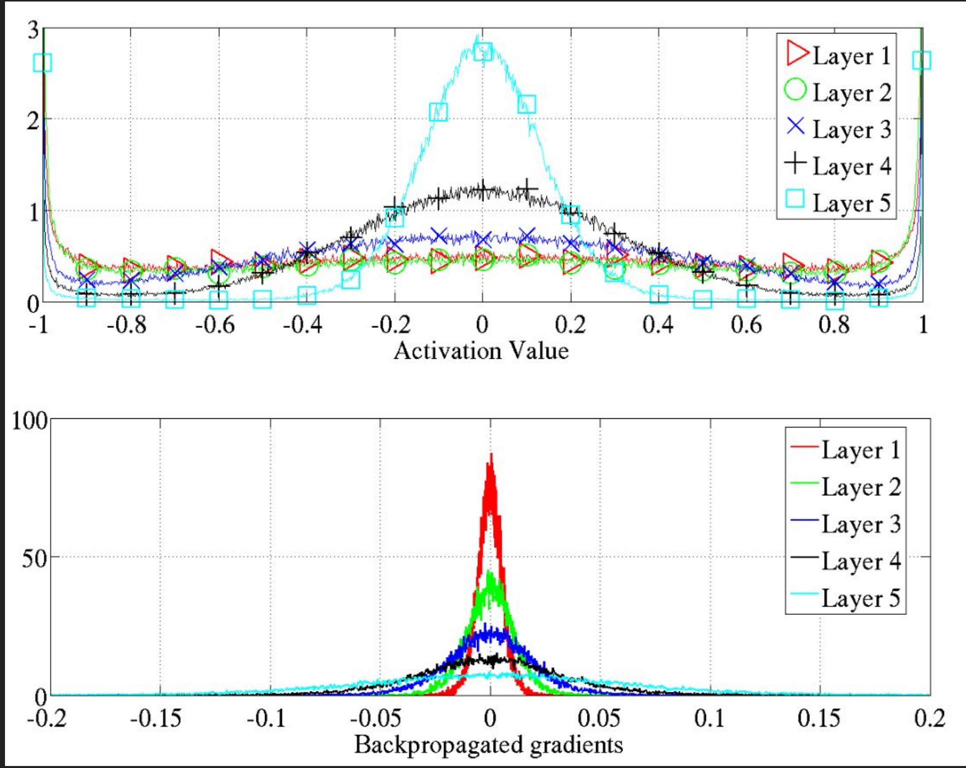
Gradient Saturation



$$\widehat{y} = \sigma(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8)$$

Ref: https://github.com/ashishpatel26/Tools-to-Design-or-Visualize-Architecture-of-Neural-Network

Vanishing Gradient. Layer-by-Layer



Weight Initialization with Normalization

Xavier Initialization, He Initialization

$$W^{(l)} := W^{(l)} * \sqrt{\frac{gain}{fan_{in}}}$$

$$W^{(l)} := W^{(l)} * \sqrt{\frac{gain}{fan_{out}}}$$

$$W^{(l)} := W^{(l)} * \sqrt{\frac{gain}{fan_{in} + fan_{out}}}$$

$$W^{(l)} := W^{(l)} * \sqrt{\frac{gain}{(fan_{in} + fan_{out})/2}}$$

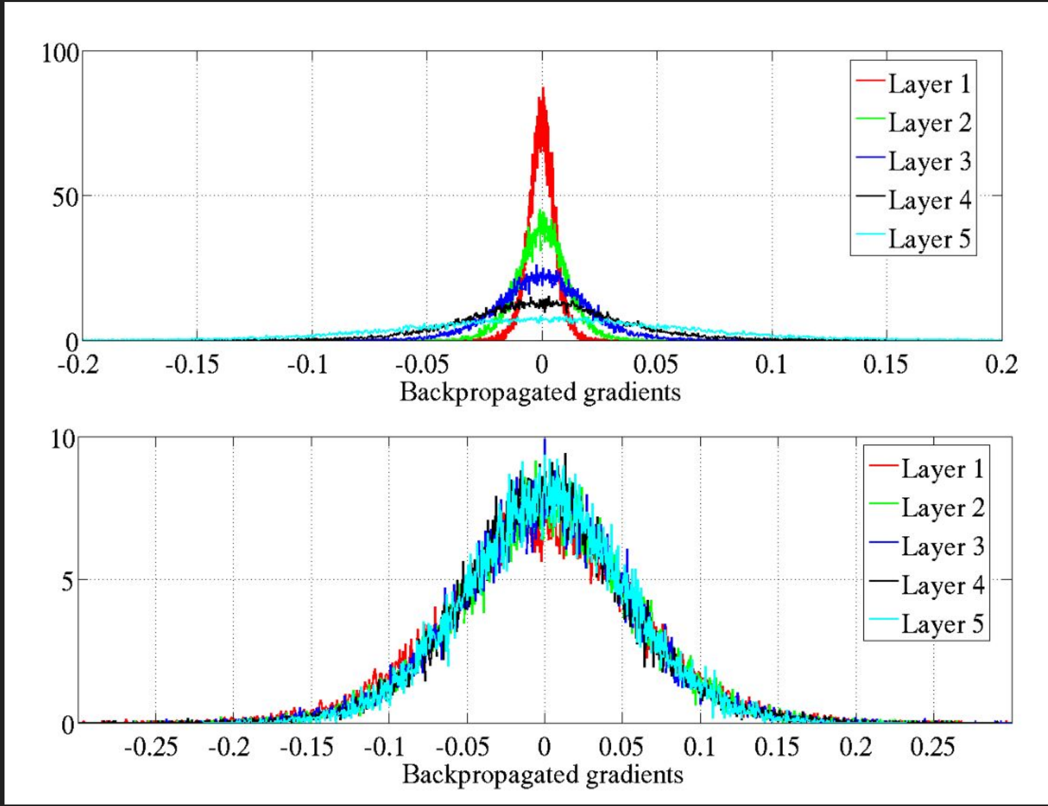
Weight Initialization

nonlinearity	gain
Linear / Identity	1
Conv{1,2,3}D	1
Sigmoid	1
Tanh	$\frac{5}{3}$
ReLU	$\sqrt{2}$
Leaky Relu	$\sqrt{\frac{2}{1+\text{negative\_slope}^2}}$
SELU	$\frac{3}{4}$

Initialization	Activation function	Variance (σ^2)
Glorot	• Linear • Tanh • Logistic • Softmax	$\sigma^2 = \frac{1}{fan_{avg}}$
He	• ReLU • Variants of ReLU	$\sigma^2 = \frac{2}{fan_{in}}$
LeCun	• SELU	$\sigma^2 = \frac{1}{fan_{in}}$

https://pytorch.org/docs/master/nn.init.html

Gradients after Proper Initialization



Note

If BatchNorm is used, initial feature weight choice is less important

Final Notes

Vanishing Gradient:

- Proper weight initialization
- Choose proper activation function
- Architecture tweaks (residual connections)

Gradient Explosion:

- Gradient clipping
- Learning rate normalization (ADAM, LAMB optimizers)

References

- https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1
- https://neptune.ai/blog/vanishing-and-exploding-gradients-debugging-monitoring-fixing
- https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html
- https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820
- https://paperswithcode.com/method/swish

HW

- Apply classification metrics to MNIST classification problem
    - Accuracy (per class and general)
    - Precision (per class and general)
    - Recall (per class and general)
    - F1-score (per class and general)
    - Confusion matrix
    - Classification report