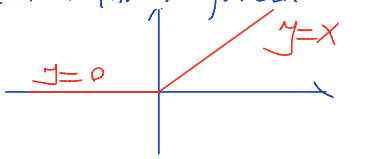
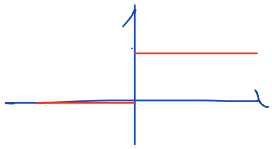
- Linear Model
  - limited: no cross term $x_1 \cdot x_2$ → introduce nonlinearity
  - Efficient: GPU
  - Stable

- Rectified Linear Unit (ReLU)

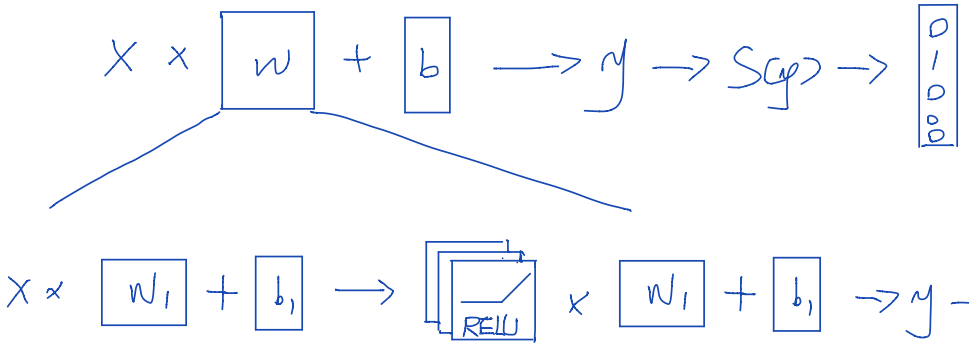
Simplest non-linear function



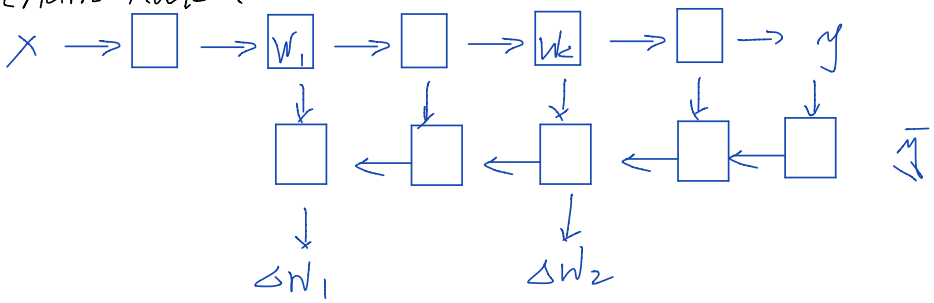
Derivative



- Network of ReLUs



- The chain Rule:



- Training a Deep Neural Network

  - Going Deeper rather than wider:

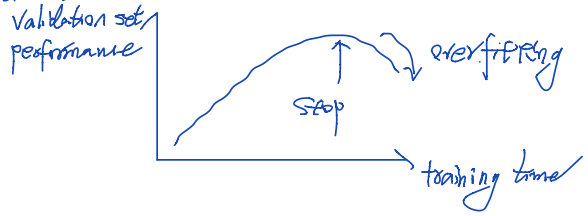
    wider (more neurons in single layer): not efficient, hard to train.

    deeper (more layers): parameter efficiency. Capture hierarchical structure in deep model.

- Regularization

  - Avoid overfit:

    - Early Termination:



    - Regularization

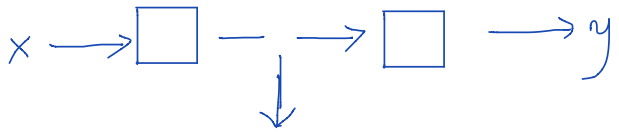
      Artificially add constraints on the parameters, implicitly reduce free parameters

  - L2 Regularization

$$\mathcal{L}' = \mathcal{L} + \beta \underbrace{\frac{1}{2}\|W\|_2^2}_{\frac{1}{2}(W_1^2 + W_2^2 + \cdots + W_n^2)}$$

$\mathcal{L}$: loss β: hyperparameter

- Dropout (regularization)



Take activations from one layer next layer, randomly drop half of them (set to 0), then repeat the process.

Intuition: Your network should never rely on any given activation to be present, because they may get squashed at any given moment so it's a way to force to learn the redundant representation, prevent overfitting.

Introduced by Geoffrey Hinton

Training and Evaluation:

- Training

  randomly drop half of activations, at the same time double the value of remaining activations get output y_t,

  repeat this for many times, get the average of prediction $y_e = \overline{y_t}$ then on average this is equivelent to not dropping the activations. Where y_e is the evaluation of dropout