ReLU and Dropout

Andrew Pozzuoli

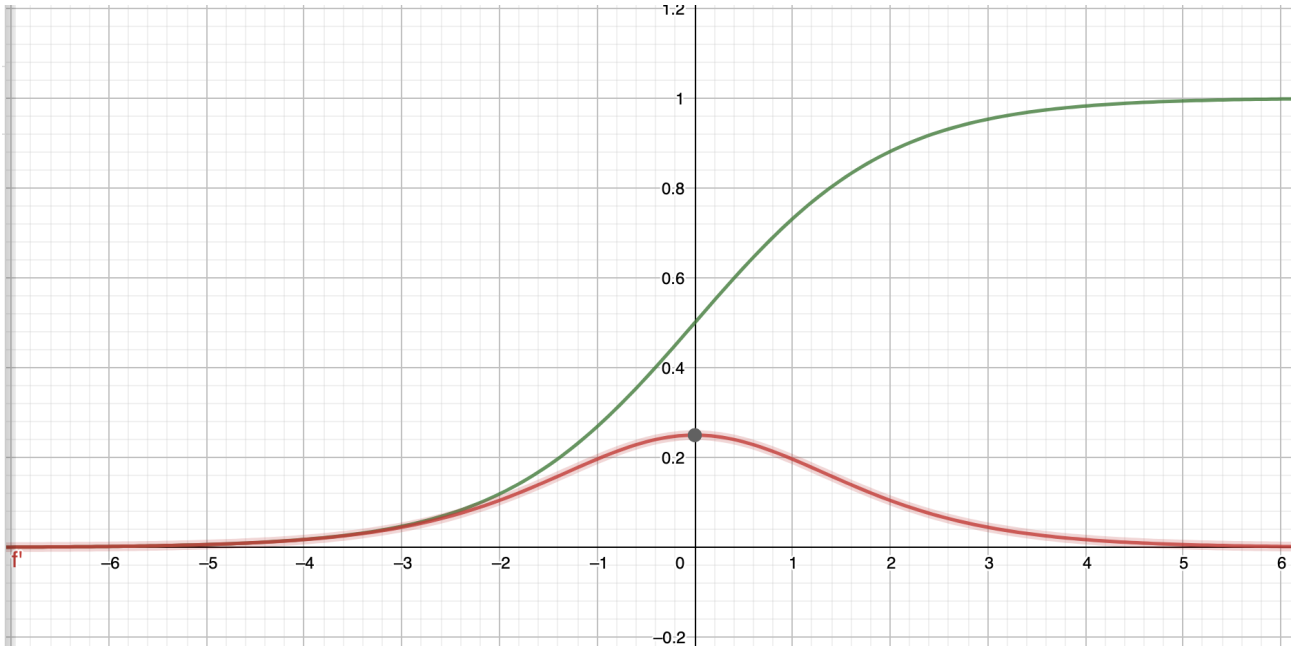
COSC 5P77

February 26, 2021

Overview

Sigmoid Activation Function

Figure: Sigmoid Activation Function and its derivative



$$sig(t) = \frac{1}{1 + e^{-t}} \tag{1}$$

Sigmoid Activation Function

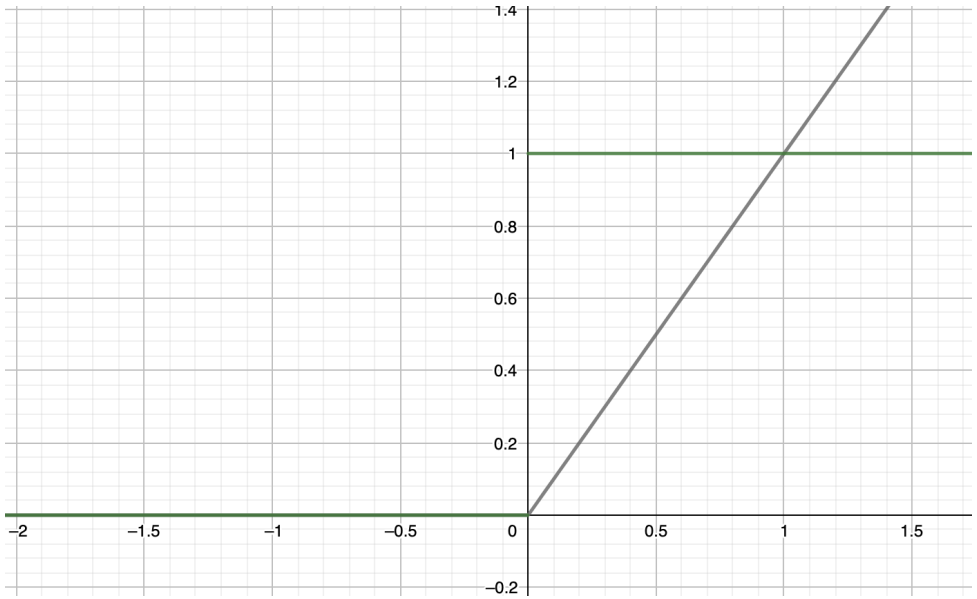
The Vanishing Gradient Problem [1]

- Small gradients for large positive and large negative values.
- Chain product of tiny values causes a "vanishing" product in earlier layers.
- Worst case scenario: updates halt completely due to the vanishing gradient.

The problem is also present with the tanh activation function for the same reasons.

Rectified Linear Units (ReLU)

Figure: Rectified Linear Units



$$r(x) = max(0, x) \tag{2}$$

[2]

Solution to Vanishing Gradient

Rectified Linear Units are advantageous over sigmoid (and tanh) since they combat the problem of the vanishing gradient [1].

- Derivative is 1 for positive inputs so gradient persists for neurons that fire.
- No leveling out for large positive values.

Other Improvements Over Sigmoid

Other benefits of rectified linear units have been observed.

- Sparsity: only have to look at positive values. In the paper *On Rectified Linear Units for Speech Processing*, it was found that on average, about 80% of units were zero after training (Zeiler et al., 2012) [3].
- Function is faster to compute due to no exponents or division [3].

Dying Neuron Effect

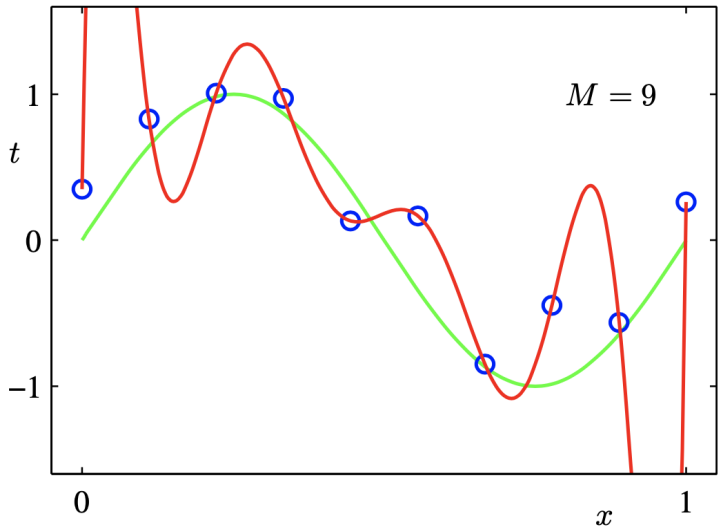
Due to the presence of exact zeroes for negative values, the chain product can become zero if even one of the values in the chain is zero. In this case, there is no gradient flow backwards and the neuron dies [2].

Possible solutions:

- Replace the exact 0 with a small negative value
- Leaky ReLU: $f(x) = \max(0.1x, x)$

Overfitting

Figure: Example of Overfitting from *Pattern Recognition and Machine Learning*, (Bishop, 2006) [4]



- The neural net trains well but tests poorly
- Result of memorizing the training data
- Poor generalization
- Usually a byproduct of too few data points or long training time

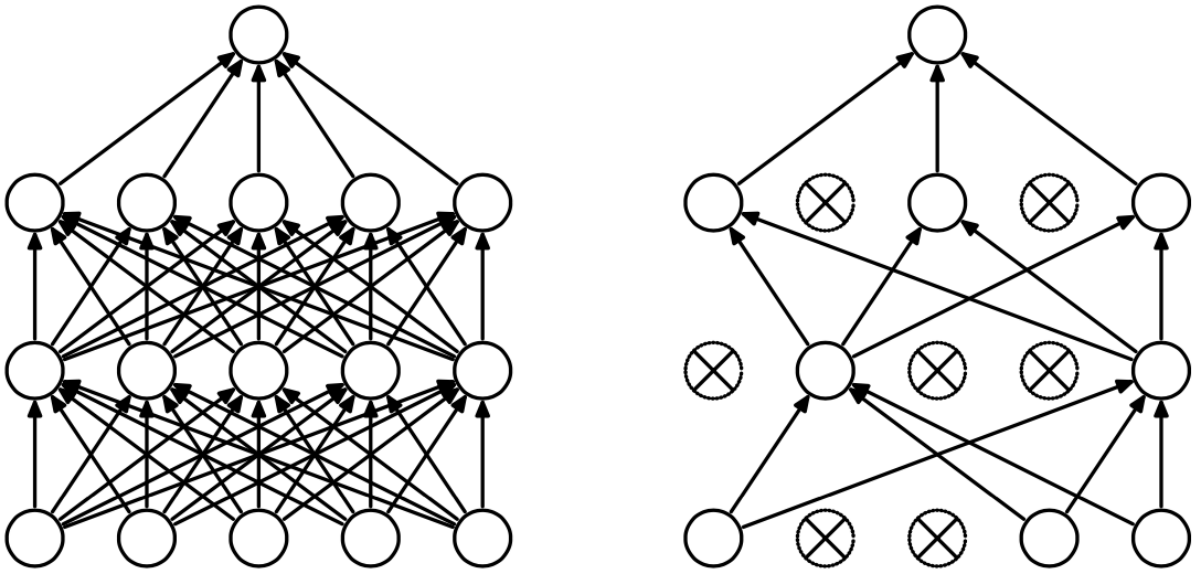
Regularization

Combat overfitting by reducing the complexity of the neural net [5]

- Reduce weights
- Average the predictions of possible parameter settings (difficult to scale this technique)
- Sparse representation (eg. ReLU)
- Combining multiple architectures (highly expensive to train multiple neural nets and recombine them)

Dropout

Figure: Standard neural net compared to a dropout neural net from *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (Srivastave et al., 2014) [5]

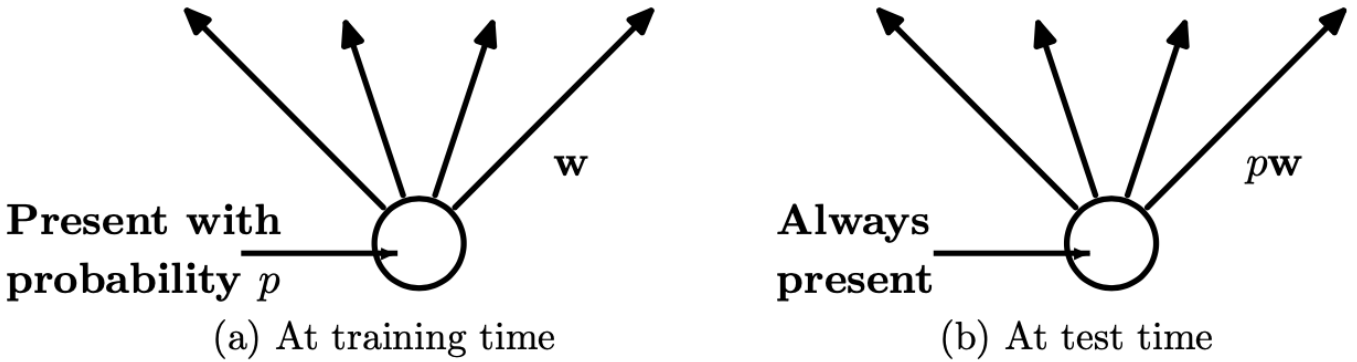


(a) Standard Neural Net (b) After applying dropout.

Main idea is to randomly drop units during training including all connections. Each pass trains on a different architecture.

Dropout

Figure: Dropout unit at training and at testing from *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (Srivastave et al., 2014) [5]



(a) At training time (b) At test time

- Hyperparameter p is assigned to each unit as the probability of that unit being dropped during a pass.
- After training, all units are reconnected and all weights are now multiplied by p.

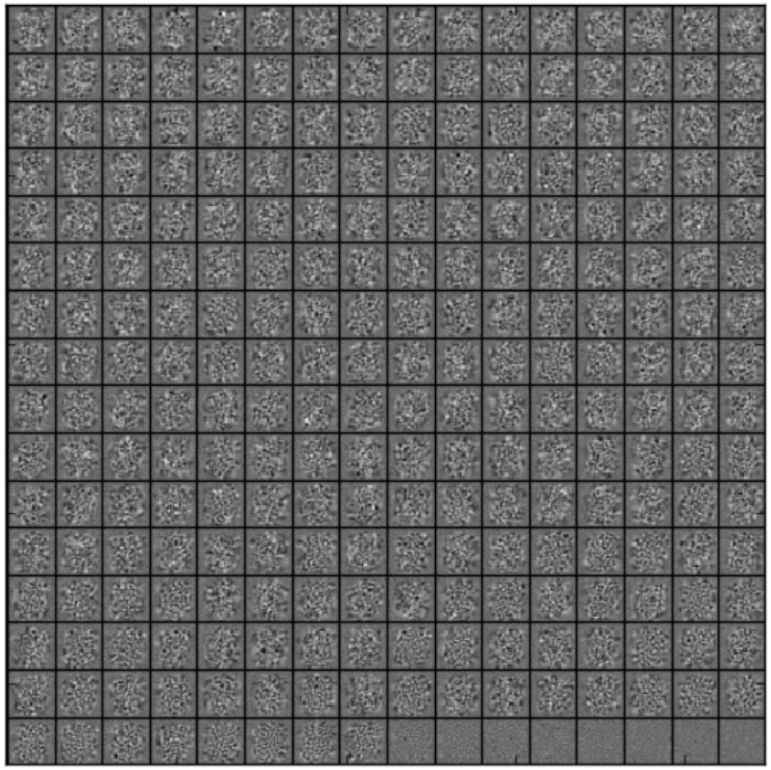
Benefits of Dropout

The paper, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (Srivastave et al., 2014) found a number of benefits of dropout empirically and offered conjecture of why this works well as a general technique [5].

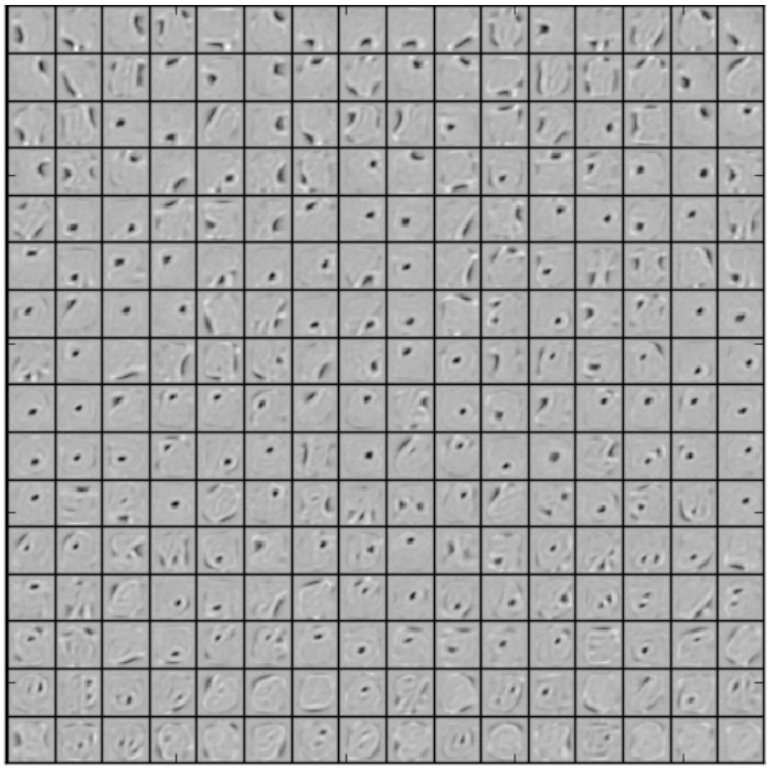
- Combines the regularization techniques of reducing the weights and training on multiple architectures.
- State-of-the-art results for a number of benchmark data sets (SVHN, ImageNet, CIFAR-100, MNIST).
- Co-adaptation cannot occur easily due to the threat of any unit being dropped at any time during training. That is to say, no unit can rely on any other to correct its mistakes and must work harder on its own.
- Weights cannot become too large since any given unit can no longer be relied upon.
- Due to the regularization from constant dropout, the neural net can be trained for longer without overfitting.

Benefits of Dropout

Figure: Learned Features on MNIST from *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (Srivastave et al., 2014) [5]



(a) Without dropout



(b) Dropout with $p = 0.5$.

Drawbacks of Dropout

Training time was found to be two to three times longer with dropout since each pass trains on a different random architecture [5].

- Noisy parameter updates
- Computed gradients are not the gradients of the final network

ReLU and Dropout

Dropout has been found to work best with larger steps in updating weights (higher momentum) [5], [6].

- Larger steps needed since dropping units introduces a lot of noise.
- Better to use a piece-wise linear function rather than a continuous function with small gradients.

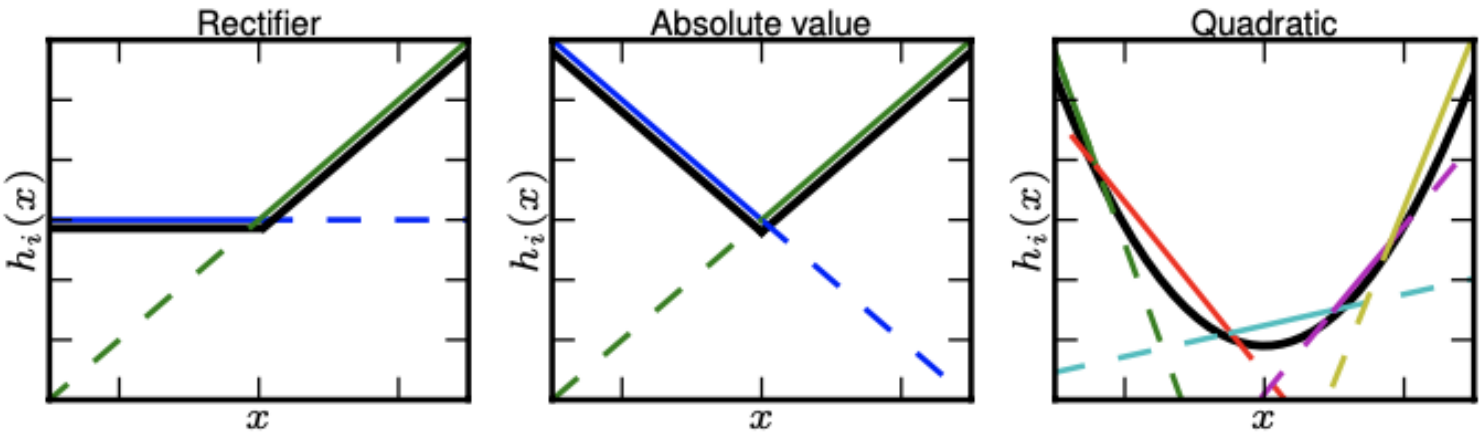
Maxout

Introduced in the 2013 paper, *Maxout Netorks* as a way of designing activation units that directly work with dropout to maximize its effectiveness (Goodfellow et al., 2013) [6].

- Each unit's activation function is a maximum of inputs.
- Able to approximate any continuous function.
- Preserves large steps in gradients.
- ReLU is an example of a maxout unit (maximum of input and 0).
- When ReLU is combined with dropout, the neural net tends to perform better since it has both the regularizing power of dropout along with the large steps in updating afforded by ReLU [5], [6].

Maxout

Figure: Graphical representations of various maxout functions and how they can approximate continuous functions from *Maxout Networks* (Goodfellow et al., 2013) [6]



Future work

Possible future work could include

- Investigating ReLU variants that address the dying neuron problem such as Leaky ReLU
- Speeding up dropout training time
- Investigating a deterministic counterpart to dropout (analytic method for selecting units to drop rather than dropping units randomly)

Bibliography I

[1] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," en, p. 9,

[2] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," en, *arXiv:1803.08375 [cs, stat]*, Feb. 2019, arXiv: 1803.08375. [Online]. Available: http://arxiv.org/abs/1803.08375 (visited on 02/15/2021).

[3] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On rectified linear units for speech processing," en, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada: IEEE, May 2013, pp. 3517–3521, ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6638312. [Online]. Available: http://ieeexplore.ieee.org/document/6638312/ (visited on 02/15/2021).

Bibliography II

[4] C. M. Bishop, *Pattern recognition and machine learning*, en, ser. Information science and statistics. New York: Springer, 2006, ISBN: 978-0-387-31073-2.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," en, p. 30,

[6] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," *arXiv:1302.4389 [cs, stat]*, Sep. 2013, arXiv: 1302.4389. [Online]. Available: http://arxiv.org/abs/1302.4389 (visited on 02/15/2021).