DEPARTMENT OF COMPUTING

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

Intro Lecture and foundations

Author: Anton Zhitomirsky

Contents

1 Pytorch

Pytorch will be used for the coursework, recommended is to look at this link.

2 Books

Recommended NLP Books:

- Speech and Language Processing. Dan Jurafsky and James H. Martin [6]
- A Primer on Neural Network Models for Natural Language Processing. Yoav Goldberg [4]
- Natural Language Processing. Jacob Eisenstein [3]

Recommended ML Books:

- Artificial Intelligence: a Modern Approach. (2009) Stuart Russell & Peter Norvig [8] with the solutions available at [9]
- Machine Learning. (1997) Tom Mitchell [7]
- Neural Networks and Deep Learning. Michael A. Nielsen.
- Introduction to Deep Learning. Eugene Charniak [2]
- Deep Learning by Ian Goodfellow [5]

State of the art NLP:

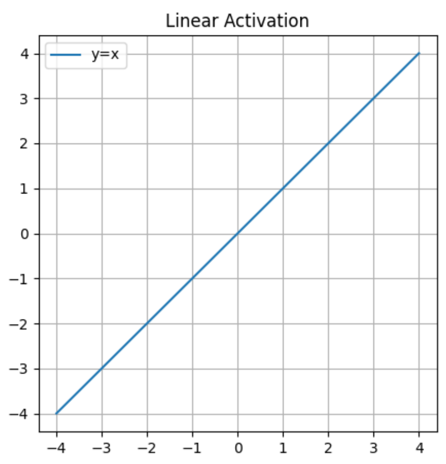
- Papers with code: [1]
- track the progress in Natural Language Processing: [10]

3 ML Refresher

3.1 Linear activation function

Definition 3.1 (Linear Activation).

$$f(x) = c \cdot x$$



- This produces a constant gradient, meaning that during backpropagation, the updates applied to weights are constant and independent of the change in input, denoted by Δx.
- If each layer in a multi-layered network employs a linear activation function, the output of one layer becomes the input to the next, perpetuating linearity throughout the network; no matter how many layers you have, the entire network behaves like a single-layer linear model. This means you could replace all N linear layers with just a single linear layer and achieve the same output. It renders the "depth" of the network irrelevant.
- Useful for regression

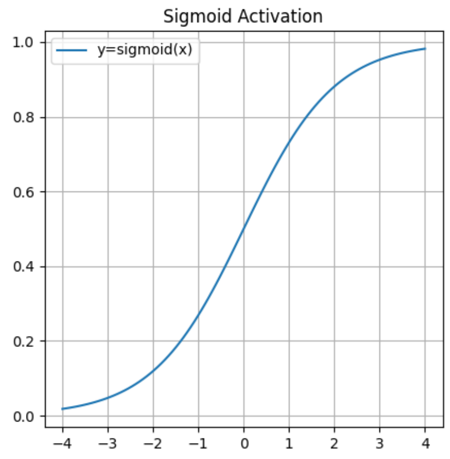
Therefore, while linear activation functions may have some use-cases, they aren't typically chosen for complex machine learning tasks that require the network to capture more complex, non-linear relationships in the data.

3.2 Non-linear activation functions

3.2.1 Sigmoid

Definition 3.2 (Sigmoid Activation).

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

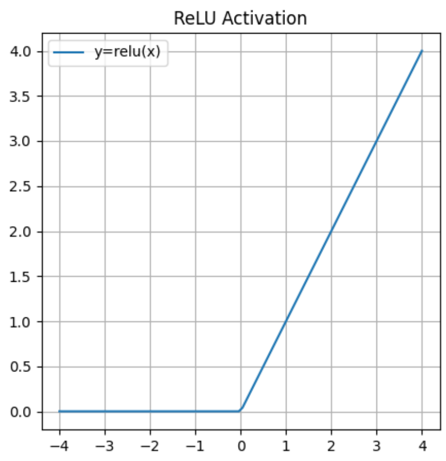


- This function is non-linear, allowing us to stack layers in a neural network, thereby facilitating the learning of more complex representations.
- gives a more analog or continuous output w.r.t binary step function.
- Smooth gradient which is crucial for gradient descent algorithms. One notable characteristic is that between the X values of -2 and 2, the curve is especially steep. This implies that small changes in the input within this region result in significant shifts in output, facilitating rapid learning during the training phase.
- Towards the tails of the function, the curve flattens out, and the output values become less sensitive to changes in input. This results in a vanishing gradient problem, where gradients become too small for the network to learn effectively, leading to slow or stalled training.
- Good for classifiers (binary and multi-label classification)

3.2.2 ReLU

Definition 3.3 (ReLU Activation)**.**

$$f(x) = \max(0, x)$$

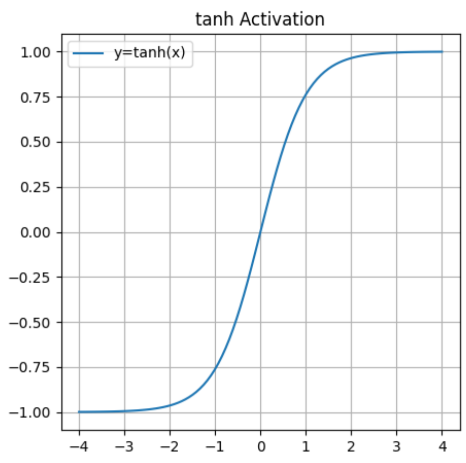


- piecewise linear that outputs the input directly if it is positive, otherwise, it outputs zero
- ReLU is inherently non-linear when considered as a whole, particularly due to the sharp corner at the origin.
- combinations of ReLU functions are also non-linear, enabling us to stack layers in neural networks effectively. Because it is a universal approximator.
- unbounded as $[0, \infty)$
- unboundedness can cause explosions of activations if not managed properly
- tends to produce sparse activations
- In a neural network with many neurons, using activation functions like sigmoid or tanh would cause almost all neurons to activate to some degree, leading to dense activations. ReLU, on the other hand, will often output zero, effectively ignoring some neurons, which can make the network more computationally efficient.
- Dying ReLU problem caused by often outputting zero (If a neuron's output is always zero (perhaps due to poor initialization), the gradient for that neuron will also be zero.) As a result, during backpropagation, the weights of that neuron remain unchanged, effectively "killing" the neuron. This can result in a portion of the neural network becoming inactive, thereby limiting its capacity to model complex functions.

3.2.3 Tanh

Definition 3.4 (Tanh Activation)**.**

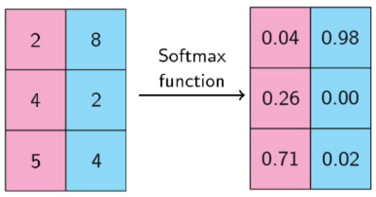
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



- Scaled version of sigmoid but from -1 to 1 instead of 0 to 1
- non-linear, we can stack it
- since $(-1, 1)$ less concern about activations becoming too large and dominating the learning process
- One key benefit of tanh over sigmoid is that its gradient is stronger; that is, the derivatives are steeper. This can make it a better choice for certain problems where faster convergence is desired.
- its outputs are zero-centered, meaning the average output is close to zero. This is beneficial for the learning process of subsequent layers, as it tends to speed up convergence by allowing for a balanced distribution of outputs and gradients.

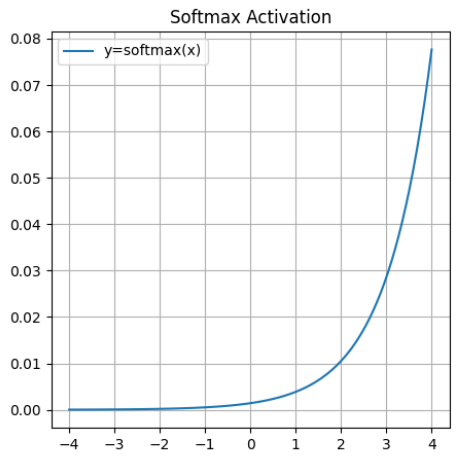
However, like the sigmoid function, tanh also suffers from the vanishing gradient problem when you stack many layers, which can slow down learning.Careful normalization of the inputs is also essential when using tanh to ensure effective learning.

3.2.4 Softmax



Definition 3.5 (Softmax Ativation)**.**

$$softmax(z_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



- applies softmax to an n-dimensional input tensor rescaling them so that the elements of the n-dimensional output tensor lie in the range $[0, 1]$ and sum to 1
- when dealing with image classification, the Softmax activation function is commonly employed to transform the network's logits into probabilities.
- useful for multi-label classification (predicting one class out of many)

3.3 Loss Functions

3.3.1 Mean suqared error

Definition 3.6 (L2 Norm, mean squared error).

$$\ell(x, y) = \boldsymbol{\mathcal{L}} = \{l_1, \ldots, l_N\}^T, \quad l_n = (x_n - y_n)^2$$

With further reduction to a single value can be either $mean(\mathcal{L})$ or $sum(\mathcal{L})$

- Measured discrepancy between the predicted output and the actual ground truth
- in a mini-batch, perform squared difference calculation for each sample within the mini-batch, the individual squared errors can then be combined into a list.
- Useful for regression

3.3.2 Binary cross-entropy

Definition 3.7 (Binary Cross Entropy).

$$\ell(x, y) = \mathcal{L} = \{l_1, \ldots, l_N\}^T,$$
$$l_n = -w_n[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)]$$

- x_n represents the predicted probability for the positive class
- y_n is the ground truth label (0 or 1) for that observation
- w_n is a weights associated with that sample

With further reduction to a single value can be either $mean(\mathcal{L})$ or $sum(\mathcal{L})$

- negative weighted sum of two entropy terms for each observation in the batch
- the input probabilities, x_n, must lie in the [0, 1] range to ensure the loss's validity
- Suitable for binary classification tasks where each observation belongs to one of two classes.
- Useful for binary classification
- Useful for multi-label classification (predicting many classes)

3.3.3 Categorical cross-entropy

useful for multi-label classification (predicting one class out of many)

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_c^{(i)} \log(\hat{y}_c^{(i)}) \tag{3.3.1}$$

3.4 Regularization

any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. (See Chapter 7 of [5])

4 Multi-Layer Perceptron Math

☞ Section not complete. Content of L01.1

References

[1] "Browse State-of-the-Art". In: (). URL: https://paperswithcode.com/sota.

[2] Eugene Charniak. *Introduction to Deep Learning*. 2019.

[3] Jacob Eisenstein. *Natural Language Processing*. 2018. URL: https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf.

[4] Yoav Goldberg. *A Primer on Neural Network Models for Natural Language Processing*. 2015. URL: https://u.cs.biu.ac.il/~yogo/nnlp.pdf.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[6] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 2023. URL: https://web.stanford.edu/~jurafsky/slp3/.

[7] Tom Mitchell. *Machine Learning*. 1997. URL: https://www.cs.cmu.edu/~tom/mlbook.html.

[8] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (Pearson Series in Artifical Intelligence)*. 2020.

[9] Stuart Russell and Peter Norvig. "Artificial Intelligence: A Modern Approach, 4th US ed. ANSWERS". In: (). URL: http://aima.cs.berkeley.edu/.

[10] "Track the progress in Natural Language Processing". In: (). URL: https://github.com/sebastianruder/NLP-progress.