

Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. It only takes a minute to sign up.



Sign up to join this community

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top



Cross Validated

SPONSORED BY

What are the advantages of ReLU over sigmoid function in deep neural networks?

Asked 6 years, 3 months ago Active 1 year, 1 month ago Viewed 174k times



173



The state of the art of non-linearity is to use rectified linear units (ReLU) instead of sigmoid function in deep neural network. What are the advantages?

I know that training a network when ReLU is used would be faster, and it is more biological inspired, what are the other advantages? (That is, any disadvantages of using sigmoid)?



115

machine-learning

neural-networks

sigmoid-curve



Share Cite Improve this question

Follow

edited Oct 24 '19 at 11:52



Ferdi

4,602

5

39

59

asked Dec 2 '14 at 2:13



RockTheStar

10.3k

29

57

86

- 1 An extra piece of answer to complete on the **Sparse vs Dense performance debate**. Don't think about NN anymore, just think about linear algebra and matrix operations, because forward and backward propagations are a series of matrix operations. Now remember that there exist a lot of optimized operator to apply to sparse matrix and so optimizing those operations in our network could dramatically improve the performance of the algorithm. I hope that could help some of you guys... – **Michael B** Apr 11 '18 at 19:09

7 Answers

Active Oldest Votes



163



Two additional major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient. But first recall the definition of a ReLU is $h = \max(0, a)$ where $a = Wx + b$.

One major benefit is the reduced likelihood of the gradient to vanish. This arises when $a > 0$. In this regime the gradient has a constant value. In contrast, the gradient of sigmoids becomes increasingly small as the absolute value of x increases. The constant gradient of ReLUs results in faster learning.

The other benefit of ReLUs is sparsity. Sparsity arises when $a \leq 0$. The more such units that exist in a layer the more sparse the resulting representation. Sigmoids on the other hand are always likely to generate some non-zero value resulting in dense representations. Sparse representations seem to be more beneficial than dense representations.

Share Cite Improve this answer

Follow

edited May 7 '16 at 21:02



Sycorax ♦

66.1k

19

161

279

answered Dec 3 '14 at 0:41



DaemonMaker

2,387

1

11

11

- 2 When you say the gradient, you mean with respect to weights or the input x ? @DaemonMaker – MAS Jan 30 '15 at 8:10
- 5 With respect to the weights. Gradient-based learning algorithms always taking the gradient with respect to the parameters of the learner, i.e. the weights and biases in a NN. – DaemonMaker Jan 30 '15 at 12:22
- 2 What do you mean by "dense" and "sparse" "representations" ? Query to google "sparse representation neural networks" doesn't seem to come up with anything relevant. – Hi-Angel Feb 10 '17 at 11:52
- 8 "Sparse representations seem to be more beneficial than dense representations." Could you provide a source or explanation? – Rohan Saxena Jan 28 '18 at 20:54
- 3 I don't understand how this answer is at all correct. The "reduced likelihood of the gradient to vanish" leaves something to be desired. The ReLU is ZERO for sufficiently small x . During learning, you gradients WILL vanish for certain neurons when you're in this regime. In fact, it's clearly unavoidable, because otherwise your network will be linear. Batch normalization solves this mostly. This doesn't even mention the most important reason: ReLU's and their gradients. are extremely fast to compute, compared to a sigmoid. – Alex R. Jun 19 '18 at 21:05



Advantage:

81



- Sigmoid: not blowing up activation
- Relu : not vanishing gradient
- Relu : More computationally efficient to compute than Sigmoid like functions since Relu just needs to pick $\max(0, x)$ and not perform expensive exponential operations as in Sigmoids
- Relu : In practice, networks with Relu tend to show better convergence performance than sigmoid. ([Krizhevsky et al.](#))

Disadvantage:

- Sigmoid: tend to vanish gradient (cause there is a mechanism to reduce the gradient as " a " increase, where " a " is the input of a sigmoid function. Gradient of Sigmoid:
 $S'(a) = S(a)(1 - S(a))$. When " a " grows to infinite large ,
 $S'(a) = S(a)(1 - S(a)) = 1 \times (1 - 1) = 0$).
- Relu : tend to blow up activation (there is no mechanism to constrain the output of the neuron, as " a " itself is the output)
- Relu : Dying Relu problem - if too many activations get below zero then most of the units(neurons) in network with Relu will simply output zero, in other words, die and thereby prohibiting learning.(This can be handled, to some extent, by using Leaky-Relu instead.)

Share Cite Improve this answer

Follow

edited Jul 22 '18 at 22:03



Jaideep

13 5

answered May 7 '16 at 20:21



Bill Ancalagon the black

1,574 2 12 16

2 Relu: not vanishing gradient. Huh? $\text{Relu}(ax + b) = 0$ for all $x < -b/a$. – Alex R. Jun 19 '18 at 21:21



Just complementing the other answers:

59

Vanishing Gradients

The other answers are right to point out that the bigger the input (in absolute value) the smaller the gradient of the sigmoid function. But, probably an even more important effect is that the derivative of the sigmoid function is **ALWAYS smaller than one**. In fact it is at most 0.25!



The down side of this is that if you have many layers, you will multiply these gradients, and the product of many smaller than 1 values goes to zero very quickly.

Since the state of the art of for Deep Learning has shown that more layers helps a lot, then this disadvantage of the Sigmoid function is a game killer. **You just can't do Deep Learning with Sigmoid.**

On the other hand the gradient of the ReLu function is either 0 for $a < 0$ or 1 for $a > 0$. That means that you can put as many layers as you like, because multiplying the gradients will neither vanish nor explode.

Share Cite Improve this answer

Follow

edited Feb 19 '18 at 0:48


answered Aug 19 '17 at 14:41



Guilherme de Lazari

754 5 8

13 This is the answer I was looking for. When people are talking about "vanishing gradients" one can't stop wondering "ReLU's gradient is exactly 0 for half of its range. Isn't that 'vanishing'". The way you describe the problem by reminding us that gradients are multiplied over many layers, brings much clarity. – Boris Gorelik Jan 3 '18 at 7:31

- 4 If this were the main reason, then couldn't we just rescale the sigmoid to $1/(1+\exp(-4x))$? Then the derivative is at most 1 (or rescale even more, to give us options above and below 1). I suspect this would perform much worse, because rescaling also reduces the area where the derivative is distinguishable from 0. But I'm not sure this answer tells the full story. – Peter Feb 21 '18 at 13:09 

That is a very good point. I guess the point missing from the other answers is the compounding effect between layers that make the vanishing much more likely. – Guilherme de Lazari Feb 22 '18 at 2:18

- 5 This answer is nonsense. The derivative of a sigmoid with constant parameter 1 is less than 1. But more generally it's $1/(1 + \exp(-ax))$, which can have an arbitrarily large derivative (just take a to be really large, so the sigmoid rapidly goes from 0 to 1). – Alex R. Jun 19 '18 at 21:12
- 11 Also you CAN do deep learning with sigmoids, you just need to normalize the inputs, for example via Batch Normalization. This will centralize your inputs to avoid saturating the sigmoid. In the original paper on Batch Normalization, the sigmoid activation neural network does nearly on par with ReLUs: arxiv.org/pdf/1502.03167.pdf – Alex R. Jun 19 '18 at 21:15



13



An advantage to ReLU other than avoiding vanishing gradients problem is that it has much lower run time. $\max(0,a)$ runs much faster than any sigmoid function (logistic function for example $= 1/(1+e^{(-a)})$ which uses an exponent which is computational slow when done often). This is true for both feed forward and back propagation as the gradient of ReLU (if $a < 0$, $=0$ else $=1$) is also very easy to compute compared to sigmoid (for logistic curve $= e^a / ((1+e^a)^2)$).



Although ReLU does have the disadvantage of dying cells which limits the capacity of the network. To overcome this just use a variant of ReLU such as leaky ReLU, ELU, etc if you notice the problem described above.

Share Cite Improve this answer Follow

answered Jun 19 '18 at 20:42



Toll

131

1

2

- 2 +1. This is one of the only correct answers here. You can also use batch normalization to centralize inputs to counteract dead neurons. – Alex R. Jun 19 '18 at 21:08
- 3 @AlexR. given your comments throughout this post, it would probably be useful if you left an answer of your own. – baxx Jan 2 '20 at 16:38



8



The main reason why ReLU is used is because it is simple, fast, and empirically it seems to work well.

Empirically, early papers observed that training a deep network with ReLU tended to converge much more quickly and reliably than training a deep network with sigmoid activation. In the early days, people were able to train deep networks with ReLU but training deep networks with sigmoid flat-out failed. There are many hypotheses that have attempted to explain why this could be.

- First, with a standard sigmoid activation, the gradient of the sigmoid is typically some fraction between 0 and 1; if you have many layers, these multiply, and might give an overall gradient that is exponentially small, so each step of gradient descent will make only a tiny change to the weights, leading to slow convergence (the vanishing gradient problem). In contrast with

machine learning - What are the advantages of ReLU over sigmoid function in deep neural networks? - Cross Validated

the weights, leading to slow convergence (the vanishing gradient problem). In contrast, with ReLU activation, the gradient of the ReLU is either 0 or 1, so after many layers often the gradient will include the product of a bunch of 1's, and thus the overall gradient is not too small or not too large. But this story might be too simplistic, because it doesn't take into account the way that we multiply by the weights and add up internal activations.

- Second, with sigmoid activation, the gradient goes to zero if the input is very large or very small. When the gradient goes to zero, gradient descent tends to have very slow convergence. In contrast, with ReLU activation, the gradient goes to zero if the input is negative but not if the input is large, so it might have only "half" of the problems of sigmoid. But this seems a bit naive too as it is clear that negative values still give a zero gradient.

Since then, we've accumulated more experience and more tricks that can be used to train neural networks. For instance, batch normalization is very helpful. When you add in those tricks, the comparison becomes less clear. It is possible to successfully train a deep network with either sigmoid or ReLU, if you apply the right set of tricks.

I suspect that ultimately there are several reasons for widespread use of ReLU today:

1. Historical accident: we discovered ReLU in the early days before we knew about those tricks, so in the early days ReLU was the only choice that worked, and everyone had to use it. And now that everyone uses it, it is a safe choice and people keep using it.
2. Efficiency: ReLU is faster to compute than the sigmoid function, and its derivative is faster to compute. This makes a significant difference to training and inference time for neural networks: only a constant factor, but constants can matter.
3. Simplicity: ReLU is simple.
4. Fragility: empirically, ReLU seems to be a bit more forgiving (in terms of the tricks needed to make the network train successfully), whereas sigmoid is more fiddly (to train a deep network, you need more tricks, and it's more fragile).
5. Good enough: empirically, in many domains, other activation functions are no better than ReLU, or if they are better, are better by only a tiny amount. So, if ReLU is simple, fast, and about as good as anything else in most settings, it makes a reasonable default.

Share Cite Improve this answer Follow

answered Jan 6 '20 at 20:09



D.W.

5,422

2

38

53

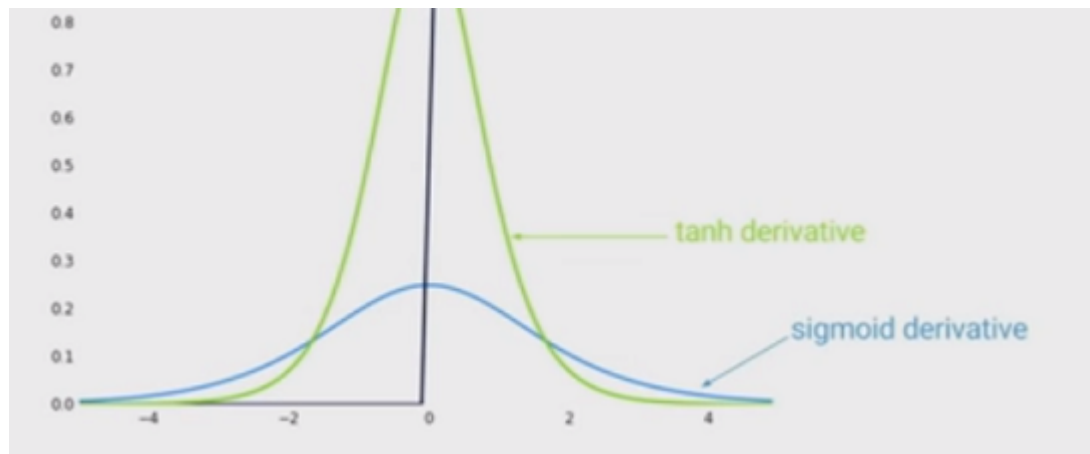


5



Main benefit is that the derivative of ReLU is either 0 or 1, so multiplying by it won't cause weights that are further away from the end result of the loss function to suffer from the vanishing gradient problem:





Share Cite Improve this answer Follow

answered Aug 17 '19 at 10:25



Maverick Meerkat

1,525 8 18

3

1. ReLu does not have the vanishing gradient problem. Vanishing gradients lead to very small changes in the weights *proportional to the partial derivative* of the error function. The gradient is multiplied n times in back propagation to get the gradients of lower layers. The effect of

multiplying the gradient n times makes the gradient to be even smaller for lower layers, leading to a very small change or even no change in the weights of lower layers. Therefore, the deeper the network, the more the effect of vanishing gradients. This makes learning per iteration slower when activation functions that suffer from vanishing gradients is used e.g Sigmoid and tanh functions. Kindly refer [here](#)

2. ReLU function is not computationally heavy to compute compared to sigmoid function. This is well covered above.

Share Cite Improve this answer Follow

answered Jan 12 '20 at 6:31



Mirikwa

61 1