Navigation

Machine Learning Mastery
Making Developers Awesome at Machine Learning

Click to Take the FREE LSTMs Crash-Course

Search...

PACE
UNIVERSITY
PREPARE FOR THE FUTURE OF DATA SCIENCE
MASTER OF SCIENCE IN DATA SCIENCE
LEARN MORE

A Gentle Introduction to Exploding Gradients in Neural Networks

by **Jason Brownlee** on December 18, 2017 in **Long Short-Term Memory Networks**

Tweet Tweet Share Share

Last Updated on August 14, 2019

Exploding gradients are a problem where large error gradients accumulate and result in very large updates to neural network model weights during training.

This has the effect of your model being unstable and unable to learn from your training data.

In this post, you will discover the problem of exploding gradients with deep artificial neural networks.

After completing this post, you will know:

- What exploding gradients are and the problems they cause during training.
- How to know whether you may have exploding gradients with your network model.
- How you can fix the exploding gradient problem with your network.

Kick-start your project with my new book Long Short-Term Memory Networks With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let’s get started.

- **Update Oct/2018**: Removed mention of ReLU as a solution.

A Gentle Introduction to Exploding Gradients in Recurrent Neural Networks
Photo by Taro Taylor, some rights reserved.

What Are Exploding Gradients?

An error gradient is the direction and magnitude calculated during the training of a neural network that is used to update the network weights in the right direction and by the right amount.

In deep networks or recurrent neural networks, error gradients can accumulate during an update and result in very large gradients. These in turn result in large updates to the network weights, and in turn, an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values.

The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1.0.

What Is the Problem with Exploding Gradients?

In deep multilayer Perceptron networks, exploding gradients can result in an unstable network that at best cannot learn from the training data and at worst results in NaN weight values that can no longer be updated.

> *… exploding gradients can make learning unstable.*

— Page 282, Deep Learning, 2016.

In recurrent neural networks, exploding gradients can result in an unstable network that is unable to learn from training data and at best a network that cannot learn over long input sequences of data.

> *… the exploding gradients problem refers to the large increase in the norm of the gradient during training. Such events are due to the explosion of the long term components*

— On the difficulty of training recurrent neural networks, 2013.

How do You Know if You Have Exploding Gradients?

There are some subtle signs that you may be suffering from exploding gradients during the training of your network, such as:

- The model is unable to get traction on your training data (e.g. poor loss).
- The model is unstable, resulting in large changes in loss from update to update.
- The model loss goes to NaN during training.

If you have these types of problems, you can dig deeper to see if you have a problem with exploding gradients.

There are some less subtle signs that you can use to confirm that you have exploding gradients.

- The model weights quickly become very large during training.
- The model weights go to NaN values during training.
- The error gradient values are consistently above 1.0 for each node and layer during training.

How to Fix Exploding Gradients?

There are many approaches to addressing exploding gradients; this section lists some best practice approaches that you can use.

1. Re-Design the Network Model

In deep neural networks, exploding gradients may be addressed by redesigning the network to have fewer layers.

There may also be some benefit in using a smaller batch size while training the network.

In recurrent neural networks, updating across fewer prior time steps during training, called truncated Backpropagation through time, may reduce the exploding gradient problem.

2. Use Long Short-Term Memory Networks

In recurrent neural networks, gradient exploding can occur given the inherent instability in the training of this type of network, e.g. via Backpropagation through time that essentially transforms the recurrent network into a deep multilayer Perceptron neural network.

Exploding gradients can be reduced by using the Long Short-Term Memory (LSTM) memory units and perhaps related gated-type neuron structures.

Adopting LSTM memory units is a new best practice for recurrent neural networks for sequence prediction.

3. Use Gradient Clipping

Exploding gradients can still occur in very deep Multilayer Perceptron networks with a large batch size and LSTMs with very long input sequence lengths.

If exploding gradients are still occurring, you can check for and limit the size of gradients during the training of your network.

This is called gradient clipping.

> *Dealing with the exploding gradients has a simple but very effective solution: clipping gradients if their norm exceeds a given threshold.*

— Section 5.2.4, Vanishing and Exploding Gradients, Neural Network Methods in Natural Language Processing, 2017.

Specifically, the values of the error gradient are checked against a threshold value and clipped or set to that threshold value if the error gradient exceeds the threshold.

> *To some extent, the exploding gradient problem can be mitigated by gradient clipping (thresholding the values of the gradients before performing a gradient descent step).*

— Page 294, Deep Learning, 2016.

In the Keras deep learning library, you can use gradient clipping by setting the *clipnorm* or *clipvalue* arguments on your optimizer before training.

Good default values are *clipnorm=1.0* and *clipvalue=0.5*.

- Usage of optimizers in the Keras API

4. Use Weight Regularization

Another approach, if exploding gradients are still occurring, is to check the size of network weights and apply a penalty to the networks loss function for large weight values.

This is called weight regularization and often an L1 (absolute weights) or an L2 (squared weights) penalty can be used.

> *Using an L1 or L2 penalty on the recurrent weights can help with exploding gradients*

— On the difficulty of training recurrent neural networks, 2013.

In the Keras deep learning library, you can use weight regularization by setting the *kernel_regularizer* argument on your layer and using an *L1* or *L2* regularizer.

- Usage of regularizers in the Keras API

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Posts

- How to Fix Vanishing Gradients Using the Rectified Linear Activation Unit (ReLU)

Books

- Deep Learning, 2016.
- Neural Network Methods in Natural Language Processing, 2017.

Papers

- On the difficulty of training recurrent neural networks, 2013.
- Learning long-term dependencies with gradient descent is difficult, 1994.

- Understanding the exploding gradient problem, 2012.

Articles

- Why is it a problem to have exploding gradients in a neural net (especially in an RNN)?
- How does LSTM help prevent the vanishing (and exploding) gradient problem in a recurrent neural network?
- Rectifier (neural networks)

Keras API

- Usage of optimizers in the Keras API
- Usage of regularizers in the Keras API

Summary

In this post, you discovered the problem of exploding gradients when training deep neural network models.

Specifically, you learned:

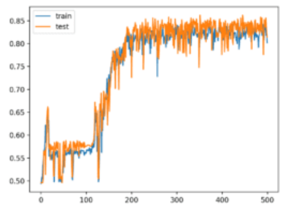
- What exploding gradients are and the problems they cause during training.
- How to know whether you may have exploding gradients with your network model.
- How you can fix the exploding gradient problem with your network.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

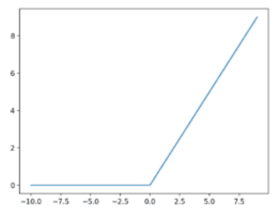
Tweet Tweet Share Share

More On This Topic

How to Avoid Exploding Gradients With Gradient Clipping



How to Fix the Vanishing Gradients Problem Using the ReLU



A Gentle Introduction to the Rectified Linear Unit (ReLU)

Crash Course in Recurrent Neural Networks for Deep Learning

Visualizing the vanishing gradient problem

Ensemble Learning Methods for Deep Learning Neural Networks

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

View all posts by Jason Brownlee →

< A Gentle Introduction to Concept Drift in Machine Learning

A Gentle Introduction to Transfer Learning for Deep Learning >

39 Responses to *A Gentle Introduction to Exploding Gradients in Neural Networks*

Nolan December 18, 2017 at 7:55 am # REPLY

I've used complex gradients in optimisation problems which has helped this issue. Does that practice exist in neural networks?

Jason Brownlee December 18, 2017 at 3:23 pm # REPLY

I've not seen it before, does it work well for you?

Sabah December 18, 2017 at 11:19 am # REPLY

Thanks for the links to the research papers and other literature.
How do you think someone interested in learning cutting edge methods in DL follow latest research? There are so many papers on arxiv, its hard to decide which ones to read or skip.

Jason Brownlee December 18, 2017 at 3:28 pm # REPLY

I recommend focusing on stuff that works up to 1-2 years ago. The bleeding edge is too noisy and most of it is noise that will not be of interest next year.

Amir Hossein December 18, 2017 at 12:24 pm # REPLY

Thank you for the article. It was so useful. These fixes for Exploding Gradients might also be useful to add to the list:

1. Careful Initialization of weights such as "Xavier" initialization or "He" initialization.
2. Perform gradient checking: sometimes the error happens when a certain implementation in the code is wrong; unless one is using pre-packaged libraries such as Tensorflow backprop. You perform GD check if you have implemented your netwirk from scratch

Jason Brownlee December 18, 2017 at 3:28 pm # REPLY

Great tips Amir! Thanks.

Yateen Gupta December 18, 2017 at 6:26 pm #

REPLY

Thanks for the article Jason. But I have a doubt regarding sigmoid or tanh functions being a cause of exploding gradients. They definitely can cause vanishing gradients due to their derivatives tending to zero for large and small activations. So how they can lead to exploding gradients ?

jiaqi,zhang December 19, 2017 at 6:11 pm #

REPLY

in the method to Fix Exploding Gradients section:

1 sigmoid->relu
the gradients of sigmoid is f(1-f), which live in (0,1); while the gradients of relu is {0,1}。 how can this replacement fix exploding gradients?

2 lstm:
lstm fix gradients vanish by replacement multiplication with addition, which transfer long dependency information to last step; also, i don't think this way can fix gradient exploding issue.

Wish to get replay ,thx

Xavier Qiu August 23, 2018 at 6:13 am #

REPLY

actually, the gradient of sigmoid is between 0 and 0.25. And wish to get some explanation for these two problems as well.

Rob vermillion December 19, 2017 at 8:43 am #

REPLY

Where can I find a machine learning program that can be used on a chromebook. Chromebook s won't let you download any use programs.

Jason Brownlee December 19, 2017 at 3:57 pm #

REPLY

I'm not sure, sorry.

Perhaps a javascript library that can do machine learning or deep learning?

For example:
http://cs.stanford.edu/people/karpathy/convnetjs/

JerzySBG May 24, 2018 at 9:15 pm #

REPLY

'Machine learning program' that you can use in webbrowser on any platform:
-free google GPU-cloud:
https://colab.research.google.com/notebooks/welcome.ipynb
Resets data every 12 hours, but for learning about neural networks it's fine.

Jason Brownlee May 25, 2018 at 9:23 am #

REPLY

Great tip.

Many runs in NLP and vision go for days though.

EC2 might be better:
https://machinelearningmastery.com/develop-evaluate-large-deep-learning-models-keras-amazon-web-services/

general December 22, 2017 at 11:33 pm #

REPLY

Amazing read, thank you!

Jason Brownlee December 23, 2017 at 5:18 am #

REPLY

Thanks.

Brad February 16, 2018 at 9:01 am #

REPLY

I recommend this relatively new paper:
https://arxiv.org/abs/1712.05577

Jason Brownlee February 16, 2018 at 9:15 am #

REPLY

Thanks for the ref, did you read it?

Brad February 16, 2018 at 10:51 am #

REPLY

Yes, I'm testing out how the metrics of pre-activation StdDev and pre-activation Sign Diversity play out with Relu, Prelu, & our innovation with a learnable ISRLU (https://arxiv.org/abs/1710.09967)

Jason Brownlee February 16, 2018 at 2:58 pm #

REPLY

Eager to hear how you go!

Anil February 26, 2018 at 10:39 pm #

REPLY

I guess ReLU is used with vanishing gradients not exploding gradients. Am I right?

Jason Brownlee February 27, 2018 at 6:28 am #

REPLY

Both I believe.

Kaan Yilmaz October 15, 2018 at 2:01 am #

REPLY

How using ReLU reduces the exploding gradient problem? ReLUfunction is unbounded for positive local fields. This means ReLUdoesn't limit its output for localField>0. Since the output of a ReLUis going to be an input to another ReLU, outputs will explode due to progressive multiplications. Isn't ReLUis one of the causes of exploding gradient problem because of this?

Jason Brownlee October 15, 2018 at 7:32 am #

REPLY

Indeed, ReLU addresses the vanishing gradient problem.

Fixed.

Radhouane Baba October 9, 2019 at 5:37 am #

REPLY

somehow when i changed from relu to sigmoid, i never got the exploding gradient…

Jason Brownlee October 9, 2019 at 8:16 am #

REPLY

It does not happen all the time, but the deeper the network, or the larger the inputs, the more likely you are to explode (or vanish).

Radhouane Baba October 9, 2019 at 7:28 pm #

do you think i should use the sigmoid (in LSTM) for this reason? or is it outdated??

Jason Brownlee October 10, 2019 at 6:56 am #

I believe lstms use sigmoids and tanh for the diffrent gates.

Using an relu in an lstm can also be effective.

Sina October 9, 2019 at 6:38 am #

REPLY

Hi Jason,

Thanks for your post.
How using Long Short-Term Memory Networks can directly affect the gradient explosion?
I do not see any connection. Furthermore, I believe this may not be a general approach as many applications are not LSTM-friendly, i.e., other architecture might be better candidates.
Thanks

Jason Brownlee October 9, 2019 at 8:18 am #

REPLY

LSTMs reduce the likelihood of an exploding gradient during back prop. Perhaps compare an LSTM vs a SimpleRNN on the same problem?

Lukas February 16, 2020 at 7:31 am #

REPLY

Its infuriating to see articles where “What you will learn” and “Conclusion” is longer than the actual content of the article.

Jason Brownlee February 17, 2020 at 7:39 am # REPLY

Thanks for your feedback Lukas.

Guney April 3, 2020 at 8:37 am # REPLY

Hi Jason,

I did not experience this before, but reading your post made me curious about it. Do you think it is possible to observe exploding gradient problem in a small Feed Forward Neural Network, such as one hidden layer, if it is possible, why can this happen?

Thanks a lot

Jason Brownlee April 3, 2020 at 9:40 am # REPLY

Maybe with very large (unscaled) inputs.

Jessy October 16, 2020 at 6:55 pm # REPLY

hi ,
swish activation should be used within the hidden layer.or can i use in the input layer…
swish activation can be used in the replacement of sigmoid activation in LSTM Architecture. is that possible..

Jason Brownlee October 17, 2020 at 6:00 am # REPLY

What is “swish activation”?

Allen May 5, 2021 at 5:58 am # REPLY

There're some ways not mentioned here.
1 – Use normalization on activation blobs, like batchnorm, layer-norm, etc.
2 – Use normalization on weights, like spectral normalization.
3 – Use modules with skip connections such as residual blocks.

Jason Brownlee May 5, 2021 at 6:15 am #

REPLY

Thanks for sharing.

vitor January 19, 2022 at 10:57 am #

REPLY

I can fraction the input values and the real or use LOG10.

James Carmichael January 20, 2022 at 7:56 am #

REPLY

Thank you for the feedback Vitor!

Leave a Reply

Name (required)

Email (will not be published) (required)

SUBMIT COMMENT

Welcome!
I'm *Jason Brownlee* PhD
and I **help developers** get results with **machine learning**.
Read more

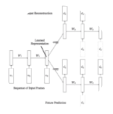
Never miss a tutorial:

Picked for you:

How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras

How to Reshape Input Data for Long Short-Term Memory Networks in Keras

How to Develop an Encoder-Decoder Model with Attention in Keras


A Gentle Introduction to LSTM Autoencoders

How to Use the TimeDistributed Layer in Keras