Double Descent Demystified: Identifying, Interpreting & Ablating the Sources of a Deep Learning Puzzle

Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W. Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo

A review by Jack Hanke

October 27, 2024

It's 2011...

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton want to win the ImageNet LSVRC-2010 contest, an image classification competition with over 1000 different classes.

It's 2011...

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton want to win the ImageNet LSVRC-2010 contest, an image classification competition with over 1000 different classes.

They use a subset of the ImageNet dataset consisting of 1.2 Million 256×256 images to train a 60 million parameter convolutional neural network called *AlexNet*.

It's 2011...

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton want to win the ImageNet LSVRC-2010 contest, an image classification competition with over 1000 different classes.

They use a subset of the ImageNet dataset consisting of 1.2 Million 256×256 images to train a 60 million parameter convolutional neural network called *AlexNet*.

Alexnet achieves state-of-the-art performance and propels the study of deep learning into the mainstream.

It's 2011...

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton want to win the ImageNet LSVRC-2010 contest, an image classification competition with over 1000 different classes.

They use a subset of the ImageNet dataset consisting of 1.2 Million 256×256 images to train a 60 million parameter convolutional neural network called *AlexNet*.

Alexnet achieves state-of-the-art performance and propels the study of deep learning into the mainstream.

Likely indirectly due to their work, we have all been in a similar situation. You solved a problem with a neural network and now have a large collection of inscrutable weights θ.

Congrats! You just trained a model!

Congrats! You just trained a model!

Question: How does your model work?

Congrats! You just trained a model!

Question: How does your model work?

- What does θ_{343} do in service of the final output? This is the *blackbox problem*.

Congrats! You just trained a model!

Question: How does your model work?

- What does θ_{343} do in service of the final output? This is the *blackbox problem*.
- The answer to this is the world of interpretability research, and is dependent on the specific problem your model is trying to solve.

Congrats! You just trained a model!

Question: How does your model work?

- What does θ_{343} do in service of the final output? This is the *blackbox problem*.
- The answer to this is the world of interpretability research, and is dependent on the specific problem your model is trying to solve.

Question: Why does your model work?

Congrats! You just trained a model!

Question: How does your model work?

- What does θ_{343} do in service of the final output? This is the *blackbox problem*.
- The answer to this is the world of interpretability research, and is dependent on the specific problem your model is trying to solve.

Question: Why does your model work?

- Why does a model with so many parameters not just memorize the data? This is the *double descent problem*.

Congrats! You just trained a model!

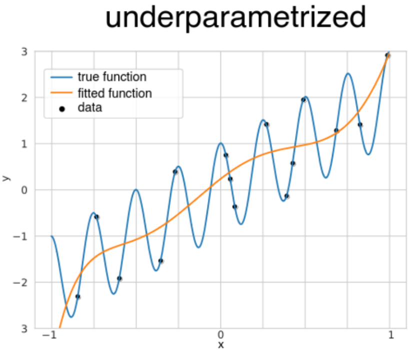
Question: How does your model work?

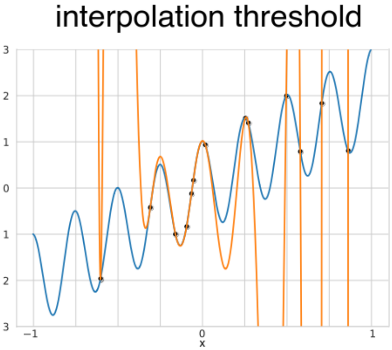
- What does θ_{343} do in service of the final output? This is the *blackbox problem*.
- The answer to this is the world of interpretability research, and is dependent on the specific problem your model is trying to solve.

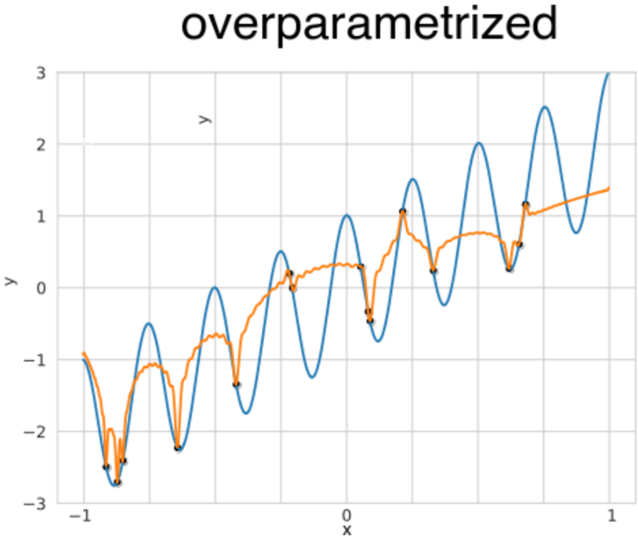
Question: Why does your model work?

- Why does a model with so many parameters not just memorize the data? This is the *double descent problem*.
- The answer to this is (in part) this paper.

The Traditional View






overparametrized
y
3
2
1
0
−1
−2
−3
−1
0
1
x
y

What is double descent?

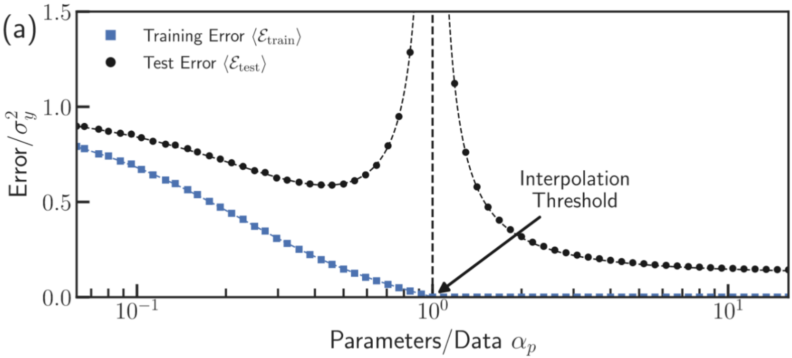
This paper defines double descent as:

A phenomenon in machine learning that many classes of models can, under relatively broad conditions, exhibit where as the number of parameters increases, the test loss falls, rises, then falls again.

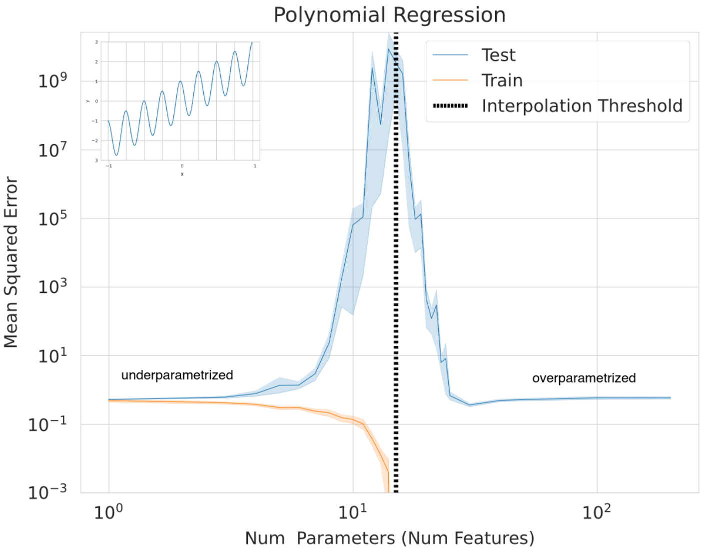
What is double descent?

This paper defines double descent as:

A phenomenon in machine learning that many classes of models can, under relatively broad conditions, exhibit where as the number of parameters increases, the test loss falls, rises, then falls again.



Double descent in polynomial regression - Empirical



Terminology

- Let P be the number of models parameters
- Let N be the number of training data
- Let D be the dimensionality of the data

Terminology

- Let P be the number of models parameters
- Let N be the number of training data
- Let D be the dimensionality of the data
- A model is *underparameterized* if $\frac{N}{P} > 1$
- A model is *overparameterized* if $\frac{N}{P} < 1$
- A model is at the *interpolation threshold* if $\frac{N}{P} = 1$

Terminology

- Let P be the number of models parameters
- Let N be the number of training data
- Let D be the dimensionality of the data
- A model is *underparameterized* if $\frac{N}{P} > 1$
- A model is *overparameterized* if $\frac{N}{P} < 1$
- A model is at the *interpolation threshold* if $\frac{N}{P} = 1$

We will next study linear models, which have a fixed value of $P = D + 1$. Therefore, double descent occurs in the direction of increasing N.

Double descent in linear regression - Mathematical

The underparametrized regime is the classic least-squares minimization problem:

$$\hat{\vec{\beta}}_{under} = \text{argmin}_{\vec{\beta}} ||X\vec{\beta} - Y||_2^2,$$

which is solved by

$$\vec{\beta}_{under} = (X^T X)^{-1} X^T Y.$$

Double descent in linear regression - Mathematical

The underparametrized regime is the classic least-squares minimization problem:

$$\hat{\vec{\beta}}_{under} = \text{argmin}_{\vec{\beta}} ||X\vec{\beta} - Y||_2^2,$$

which is solved by

$$\vec{\beta}_{under} = (X^T X)^{-1} X^T Y.$$

For the overparameterized regime, the above optimization problem has infinite solutions. Therefore, we need to choose a different optimization problem:

$$\hat{\vec{\beta}}_{over} = \text{argmin}_{\vec{\beta}} ||\vec{\beta}||_2^2 \text{ s.t. } \forall n \in (1, \dots, N) \ \vec{x}_n \vec{\beta} = y_n$$

which is solved by

$$\vec{\beta}_{over} = X^T (X X^T)^{-1} Y.$$

Why this choice?

$$\hat{\vec{\beta}}_{over} = \text{argmin}_{\vec{\beta}} ||\vec{\beta}||_2^2 \text{ s.t. } \forall n \in (1, \ldots, N) \; \vec{x}_n \vec{\beta} = y_n$$

We choose this optimization problem because *it is the optimization problem that gradient decent implicity minimizes*!

Double descent in nonlinear models - Intuition

TODO

Summary

TODO

Thank you for listening!