Embrace Randomness in Machine Learning

by **Jason Brownlee** on September 28, 2016 in **Machine Learning Algorithms**

Why Do You Get Different Results On Different Runs Of An Algorithm With The Same Data?

Applied machine learning is a tapestry of breakthroughs and mindset shifts.

Understanding the role of randomness in machine learning algorithms is one of those breakthroughs.

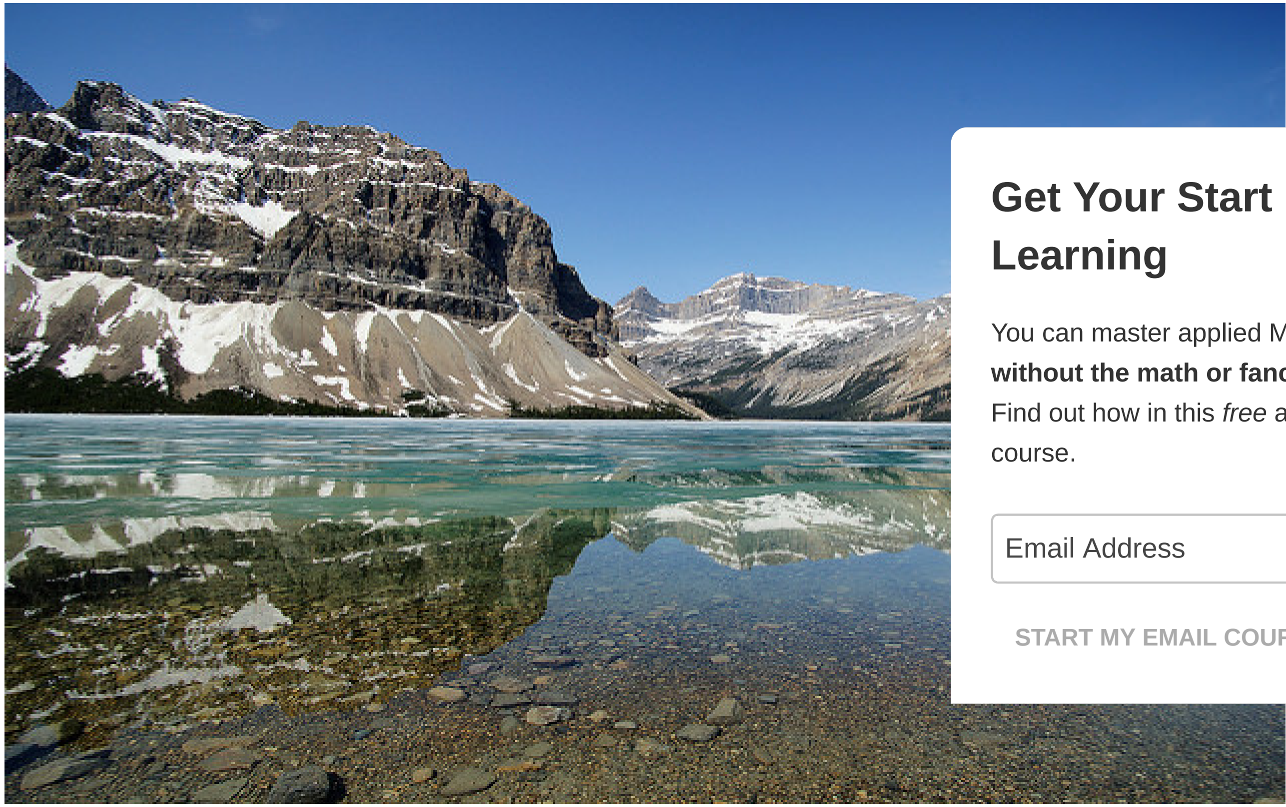
Once you get it, you will see things differently. In a whole new light. Things like choosing between one algorithm and another, hyperparameter tuning and reporting results.

You will also start to see the abuses everywhere. The criminally unsupported performance claims.

In this post, I want to gently open your eyes to the role of random numbers in machine learning. I want to give you the tools to embrace this uncertainty. To give you a breakthrough.

Let's dive in.

(*special thanks to Xu Zhang and Nil Fero who promoted this post*)



Embrace Randomness in Applied Machine Learning
Photo by Peter Pham, some rights reserved.

Why Are Results Different With The Same Data?

A lot of people ask this question or variants of this question.

You are not alone!

I get an email along these lines once per week.

Here are some similar questions posted to Q&A sites:

- Why do I get different results each time I run my algorithm?
- Cross-Validation gives different result on the same data
- Randomness in Artificial Intelligence & Machine Learning
- Why are the weights different in each running after convergence?
- Does the same neural network with the same learning data and same test data in two computer

Machine Learning Algorithms Use Random Numbers

Machine learning algorithms make use of randomness.

1. Randomness in Data Collection

Trained with different data, machine learning algorithms will construct different models. It depends o rent data is called the model variance (as in the bias-variance trade off).

So, the data itself is a source of randomness. Randomness in the collection of the data.

2. Randomness in Observation Order

The order that the observations are exposed to the model affects internal decisions.

Some algorithms are especially susceptible to this, like neural networks.

It is good practice to randomly shuffle the training data before each training iteration. Even if your algorithm is not susceptible. It's a best practice.

3. Randomness in the Algorithm

Algorithms harness randomness.

An algorithm may be initialized to a random state. Such as the initial weights in an artificial neural network.

Votes that end in a draw (and other internal decisions) during training in a deterministic method may rely on randomness to resolve.

4. Randomness in Sampling

We may have too much data to reasonably work with.

In which case, we may work with a random subsample to train the model.

5. Randomness in Resampling

We sample when we evaluate an algorithm.

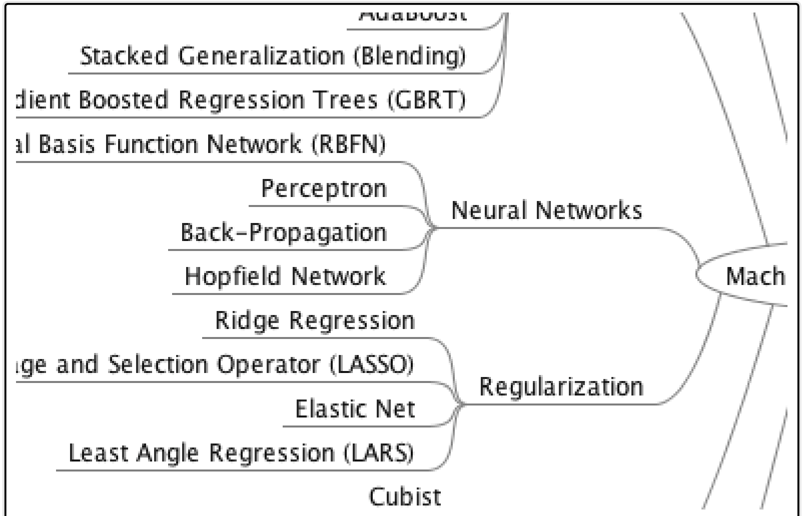
We use techniques like splitting the data into a random training and test set or use k-fold cross valida

The result is an estimate of the performance of the model (and process used to create it) on unseen

No Doubt

There's no doubt, randomness plays a big part in applied machine learning.

The randomness that we can control, should be co



Sample of the handy machine learning algorithms mind map.

Random Seeds and Reproducible Results

Run an algorithm on a dataset and get a model.

Can you get the same model again given the same data?

You should be able to. It should be a r ng project.

We achieve reproducibility in applied machine learning by using the exact same **code**, **data** and **sec**

Random numbers are generated in software using a pretend random number generator. It's a simple numbers that are random enough for most applications.

This math function is deterministic. If it uses the same starting point called a seed number, it will give

Problem solved.
Mostly.

We can get reproducible results by fixing the random number generator's seed before each model we construct.

In fact, this is a best practice.

We should be doing this if not already.

In fact, we should be giving the same sequence of random numbers to each algorithm we compare and each technique we try.

It should be a default part of each experiment we run.

Machine Learning Algorithms are Stochastic

If a machine learning algorithm gives a different model with a different sequence of random numbers, then which model do we pick?

Ouch. There's the rub.

I get asked this question from time to time and I love it.

It's a sign that someone really gets to the meat of all this applied machine learning stuff – or is about to.

- Different runs of an algorithm with…
- Different random numbers give…
- Different models with…
- Different performance characteristics…

But the differences are within a range.

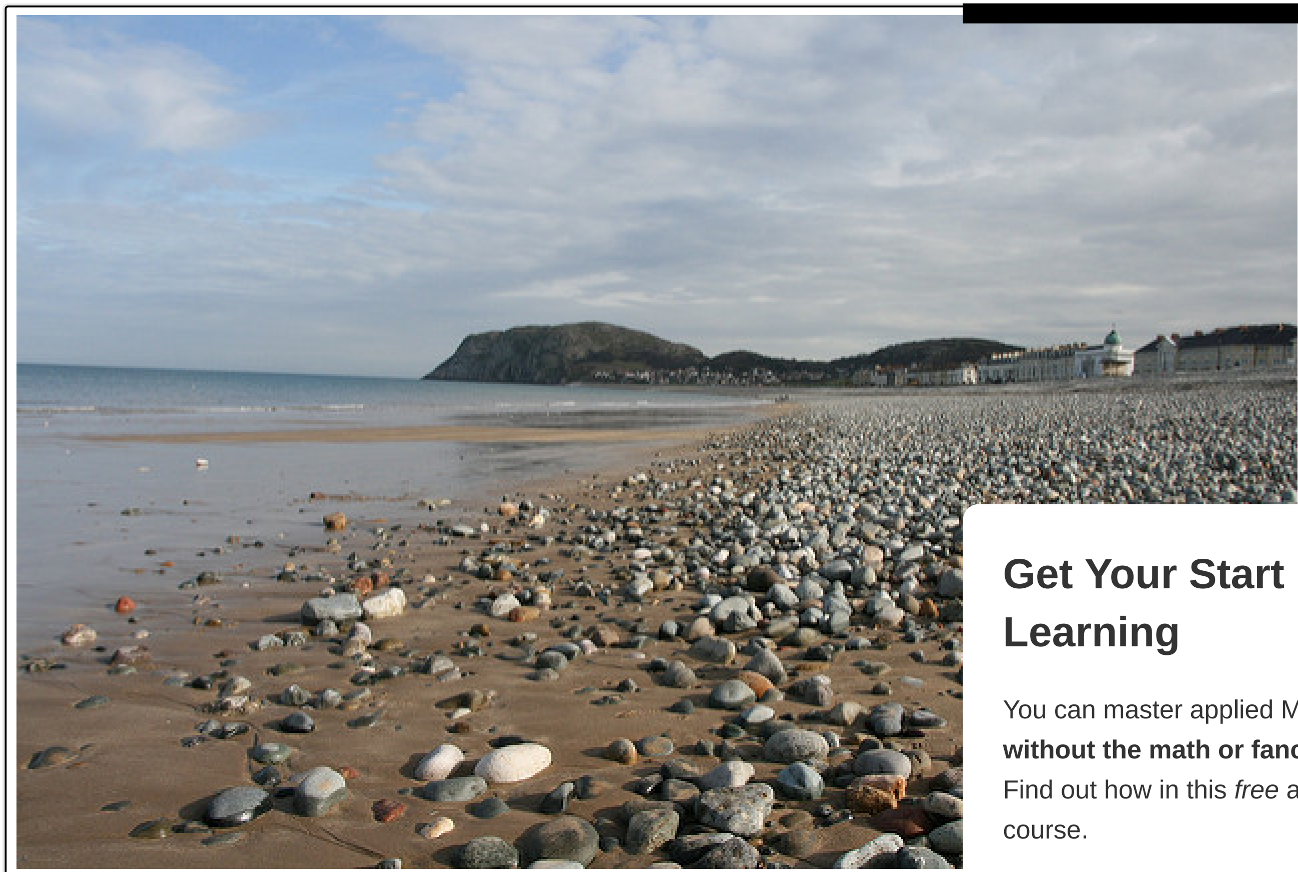
A fancy name for this difference or random behavior within a range is stochastic.

Machine learning algorithms are stochastic in practice.

- Expect them to be stochastic.
- Expect there to be a range of models to choose from and not a single model.
- Expect the performance to be a range and not a single value.

These are very real expectations that you MUST address in practice.

What tactics can you think of to address these expectations?



Machine Learning Algorithms Use Random Numbers
Photo by Pete, some rights reserved.

Tactics To Address The Uncertainty of Stochastic Algorith

Thankfully, academics have been struggling with this challenge for a long time.

There are 2 simple strategies that you can use:

1. Reduce the Uncertainty.
2. Report the Uncertainty.

Tactics to Reduce the Uncertainty

If we get different models essentially every time we run an algorithm, what can we do?

How about we try running the algorithm many times and gather a population of performance measures.

We already do this if we use *k*-fold cross validation. We build *k* different models.

We can increase *k* and build even more models, as long as the data within each fold remains representative of the problem.

We can also repeat our evaluation process *n* times to get even more numbers in our population of performance measures.

This tactic is called random repeats or random restarts.

It is more prevalent with stochastic optimization and neural networks, but is just as relevant generally. Try it.

Tactics to Report the Uncertainty

Never report the performance of your machine learning algorithm with a single number.

If you do, you've most likely made an error.

You have gathered a population of performance measures. Use statistics on this population.

This tactic is called report summary statistic

The distribution of results is most likely a Gaussian, so a great start would be to report the mean and
highest and lowest performance observed.

In fact, this is a best practice.

You can then compare populations of result measures when you're performing model selection. Such as:

- Choosing between algorithms.
- Choosing between configurations for one algorithm.

You can see that this has important implications on the processes you follow. Such as: to select which [redacted] and choosing algorithm hyperparameters.

Lean on statistical significance tests. Statistical tests can determine if the difference between one population of result measures is significantly different from a second population of results.

Report the significance as well.

This too is a best practice, that sadly does not have enough adoption.

Wait, What About Final Model Selection

The final model is the one prepared on the entire training dataset, once we have chosen an algorithr

It's the model we intend to use to make predictions or deploy into operations.

We also get a different final model with different sequences of random numbers.

I've had some students ask:

> *Should I create many final models and select the one with the best accuracy on a hold out va*

"*No*" I replied.

This would be a fragile process, highly dependent on the quality of the held out validation dataset. Y a small sample of data.

Sounds like a recipe for overfitting.

In general, I would rely on the confidence gained from the above tactics on reducing and reporting uncertainty. Often I just take the first model, it's just as good as any other.

Sometimes your application domain makes you care more.

In this situation, I would tell you to build an ensemble of models, each trained with a different rando[redacted]

Use a simple voting ensemble. Each model makes a prediction and the mean of all predictions is reported as the final prediction.

Make the ensemble as big as you need to. I think 10, 30 or 100 are nice round numbers.

Maybe keep adding new models until the predictions become stable. For example, continue until the variance of the predictions tightens up on some holdout set.

Summary

In this post, you discovered why random numbers are integral to applied machine learning. You can't really escape them.

You learned about tactics that you can use to ensure that your results are reproducible.

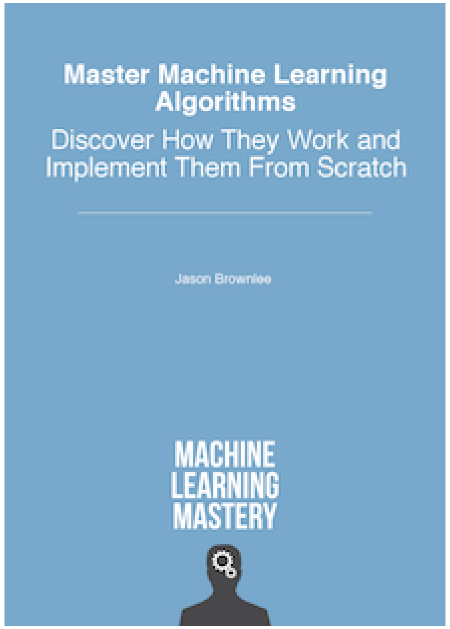
You learned about techniques that you can use to embrace the stochastic nature of machine learnin results.

For more information on the importance of reproducible results in machine learning and techniques t

- Reproducible Machine Learning Results By Default

Do you have any questions about random numbers in machine learning or about this post?

Ask your question in the comments and I will do my best to answer.



Discover how in my new Ebook: Master [redacted]

It covers **explanations** and **examples** of **10 top algorithms**, like:
Linear Regression, *k-Nearest Neighbors*, *Support Vector Machines* and much more…

Finally, Pull Back the Curtain on
Machine Learning Algorithms

Skip the Academics. Just Results.

Click to learn more.

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professiona
dedicated to helping developers get started and get good at applied machine learning.
Learn more.

View all posts by Jason Brownlee →

‹ Python Machine Learning Mini-Course

Deploy Your Predictive Model To Production ›

17 Responses to *Embrace Randomness in Machine Learning*

Cameron DP September 28, 2016 at 6:09 am # REPLY

A question that I've faced is whether to use a fixed seed in a production algorithm. If the results are customer facing, is there any harm in using a seed so the results are consistent? (Maybe not consistent, but the variance is atleast reduced)

Jason Brownlee September 28, 2016 at 7:43 am # REPLY

Indeed Cameron, a tough one.

I have used fixed seeds in production before. I want variability to come from the data.

Today, I might make a different call. I might deploy an ensemble of the same model and drive stabilit

Peter P September 7, 2017 at 1:41 am #

Hello Dr. Brownlee,

I kind of get the idea of creating an ensemble of the same model, but how do you implement the

Jason Brownlee September 7, 2017 at 12:57 pm # REPLY

Count the predictions for each class and use the max function, or use an existing implementation like that in Weka or sklearn.

Alan Beckles MD MS October 4, 2016 at 4:41 pm #

This is an excellent thought provoking post. Best practice may be your recommendation to build [redacted] seeds, then use voting.

Jason Brownlee October 5, 2016 at 8:24 am # REPLY

Thanks Alan, I'm glad you found it useful.

Rajesh October 7, 2016 at 4:48 am # REPLY

Really amazing blog it is a lot we can learn here, still we have to agree that handling data is alw

Wes Turner January 14, 2017 at 6:22 pm #

In addition to specific libraries' function parameters like 'seed', Python, for example, reads 0 or environment variable.

From "Ten Simple Rules for Reproducible Computational Research" http://journals.plos.org/ploscompbio

> Rule 6: For Analyses That Include Randomness, Note Underlying Random Seeds

Are random seed(s) are just another parameter to optimize? (I tend to agree with your ensembling recor

Jason Brownlee January 15, 2017 at 5:28 am # REPLY

Thanks Wes.

Sophia August 2, 2017 at 12:53 pm #

REPLY

Sometimes, the training results are all 0 and the error can't convergence to a small value, showing a line vs the iterations. But if I train it again without any change, it has a reasonable good result. What's the problem? And how can I avoide the failed training? Thank you. Really learn a lot from your post!

Jason Brownlee August 3, 2017 at 6:43 am #

REPLY

Consider repeating the experiment multiple times and take the average result as a summary of model performance.

For example, see this post:
http://machinelearningmastery.com/evaluate-skill-deep-learning-models/

Sharps September 14, 2017 at 9:06 am #

Hello Jason!
Thanks for you're all posts/analysis/documentations/answers!!
I roam in NNs coming from sql dev and i see a new dimension now!

Jason Brownlee September 15, 2017 at 12:09 pm #

I'm glad they helped.

Nil December 11, 2017 at 11:00 pm #

REPLY

Hi, Dr. Jason,

Thank you, this post helped me so much, to handle with my uncertainty.

Your posts a really good and helpfully.

Best Regards.

Jason Brownlee December 12, 2017 at 5:33 am #

REPLY

I'm glad to hear that.

Dinto January 2, 2018 at 10:40 am #

REPLY

Do you suggest an ensemble of models also for prediction?

Jason Brownlee January 2, 2018 at 3:59 pm #

For sure.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

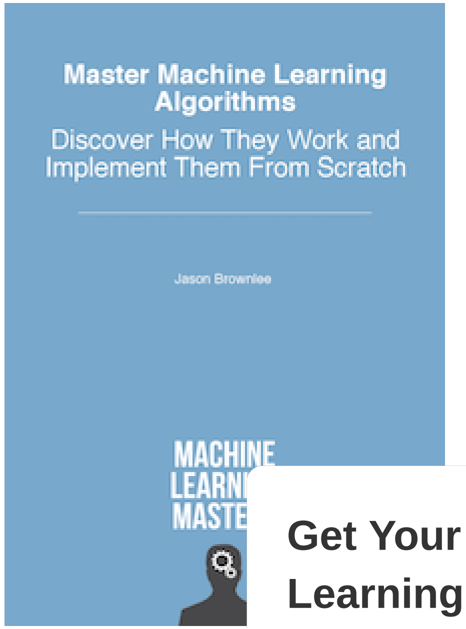
Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More

Understand Machine Learning Algorithms!

Sick of all the advanced math?
Need step-by-step explainations for top algorithms?
Want worked examples in spreadsheets?

Finally Understand Machine Learning Algorithms!

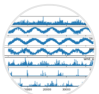


POPULAR

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
JULY 21, 2016

Your First Machine Learning Project in Python Step-By-Step
JUNE 10, 2016

Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016



Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017

How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016

Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016

How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras
AUGUST 9, 2016