How to Evaluate the Skill of Deep Learning Models

by **Jason Brownlee** on May 31, 2017 in **Deep Learning**

I often see practitioners expressing confusion about how to evaluate a deep learning model.

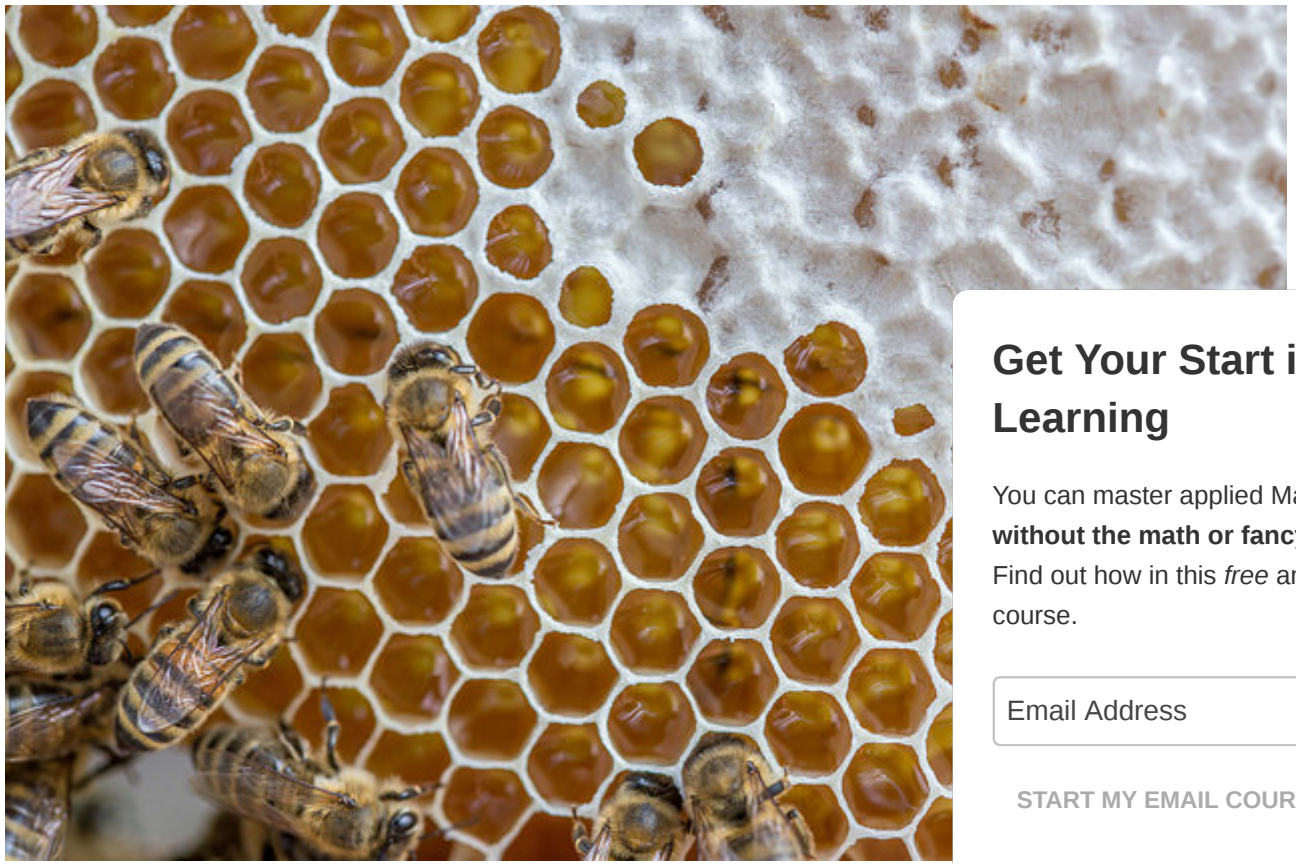
This is often obvious from questions like:

- What random seed should I use?
- Do I need a random seed?
- Why don't I get the same results on subsequent runs?

In this post, you will discover the procedure that you can use to evaluate deep learning models and the rationale for using it.

You will also discover useful related statistics that you can calculate to present the skill of your model, such as standard deviation, standard error, and confidence intervals.

Let's get started.



How to Evaluate the Skill of Deep Learning Models
Photo by Allagash Brewing, some rights reserved.

The Beginner's Mistake

You fit the model to your training data and evaluate it on the test dataset, then report the skill.

Perhaps you use k-fold cross validation to evaluate the model, then report the skill of the model.

Get Your Start in Machine Learning

This is a mistake made by beginners.

It looks like you're doing the right thing, but there is a key issue you have not accounted for:

Deep learning models are stochastic.

Artificial neural networks use randomness while being fit on a dataset, such as random initial weights and random shuffling of data during each training epoch during stochastic gradient descent.

This means that each time the same model is fit on the same data, it may give different predictions and in turn have different overall skill.

Estimating Model Skill (*Controlling for Model Variance*)

We don't have all possible data; if we did, we would not need to make predictions.

We have a limited sample of data, and from it we need to discover the best model we can.

Use a Train-Test Split

We do that by splitting the data into two parts, fitting a model or specific model configuration on the f e predictions on the rest, then evaluating the skill of those predictions. This is called a train-test split a think the model will perform in practice when it makes predictions on new data.

For example, here's some pseudocode for evaluating a model using a train-test split:

```
train, test = split(data)
model = fit(train.X, train.y)
predictions = model.predict(test.X)
skill = compare(test.y, predictions)
```

A train-test split is a good approach to use if you have a lot of data or a very slow model to train, but the resulting skill score for the model will be noisy because of the randomness in the data (variance of the model).

This means that the same model fit on different data will give different model skill scores.

Use k-Fold Cross Validation

We can often tighten this up and get more accurate estimates of model skill using techniques like k-fold cross validation. This is a technique that systematically splits up the available data into k-folds, fits the model on k-1 folds, evaluates it on the held out fold, and repeats this process for each fold.

This results in k different models that have k different sets of predictions, and in turn, k different skill scores.

For example, here's some pseudocode for evaluating a model using a k-fold cross validation:

```
scores = list()
for i in k:
    train, test = split_old(data, i)
    model = fit(train.X, train.y)
    predictions = model.predict(test.X)
    skill = compare(test.y, predictions)
    scores.append(skill)
```

A population of skill scores is more useful as we can take the mean and report the average expected closer to the actual performance of the model in practice. For example:

```
mean_skill = sum(scores) / count(scores)
```

We can also calculate a standard deviation using the mean_skill to get an idea of the average sprea

```
standard_deviation = sqrt(1/count(scores) * sum( (score - mean_skill)^2 ))
```


Estimating a Stochastic Model's Skill (*Controlling for Model Stability*)

Stochastic models, like deep neural networks, add an additional source of randomness.

This additional randomness gives the model more flexibility when learning, but can make the model less stable (e.g. different results when the same model is trained on the same data).

This is different from model variance that gives different results when the same model is trained on different data.

To get a robust estimate of the skill of a stochastic model, we must take this additional source of vari

Get Your Start in Machine Learning

Fix the Random Seed

One way is to use the same randomness every time the model is fit. We can do that by fixing the random number seed used by the system and then evaluating or fitting the model. For example:

```
seed(1)
scores = list()
for i in k:
    train, test = split_old(data, i)
    model = fit(train.X, train.y)
    predictions = model.predict(test.X)
    skill = compare(test.y, predictions)
    scores.append(skill)
```

This is good for tutorials and demonstrations when the same result is needed every time your code is run.

This is fragile and not recommended for evaluating models.

See the post:

- Embrace Randomness in Machine Learning
- How to Get Reproducible Results with Keras

Repeat Evaluation Experiments

A more robust approach is to repeat the experiment of evaluating a non-stochastic model multiple ti

For example:

```
scores = list()
for i in repeats:
    run_scores = list()
    for j in k:
        train, test = split_old(data, j)
        model = fit(train.X, train.y)
        predictions = model.predict(test.X)
        skill = compare(test.y, predictions)
        run_scores.append(skill)
    scores.append(mean(run_scores))
```

Note, we calculate the mean of the estimated mean model skill, the so-called grand mean.

This is my recommended procedure for estimating the skill of a deep learning model.

Because repeats is often >=30, we can easily calculate the standard error of the mean model skill, which is how much the estimated mean of model skill score differs from the unknown actual mean model skill (e.g. how wrong mean_skill might be)

```
standard_error = standard_deviation / sqrt(count(scores))
```

Further, we can use the standard_error to calculate a confidence interval for mean_skill. This assumes that the distribution of the results is Gaussian, which you can check by looking at a Histogram, Q-Q plot, or using statistical tests on the collected scores.

For example, the interval of 95% is (1.96 * standard_error) around the mean skill.

```
interval = standard_error * 1.96
lower_interval = mean_skill - interval
upper_interval = mean_skill + interval
```

There are other perhaps more statistically robust methods for calculating confidence intervals than u s:

- Calculating the Binomial proportion confidence interval.
- Using the bootstrap to estimate an empirical confidence interval.

How Unstable Are Neural Networks?

It depends on your problem, on the network, and on its configuration.

I would recommend performing a sensitivity analysis to find out.

Evaluate the same model on the same data many times (30, 100, or thousands) and only vary the seed for the random number generator.

Then review the mean and standard deviation of the skill scores produced. The standard deviation (average distance of scores from the mean score) will give you an idea of just how unstable your model is.

How Many Repeats?

Get Your Start in Machine Learning

I would recommend at least 30, perhaps 100, even thousands, limited only by your time and computer resources, and diminishing returns (e.g. standard error on the mean_skill).

More rigorously, I would recommend an experiment that looked at the impact on estimated model skill versus the number of repeats and the calculation of the standard error (how much the mean estimated performance differs from the true underlying population mean).

Further Reading

- Embrace Randomness in Machine Learning
- How to Train a Final Machine Learning Model
- Comparing Different Species of Cross-Validation
- Empirical Methods for Artificial Intelligence, Cohen, 1995.
- Standard error on Wikipedia

Summary

In this post, you discovered how to evaluate the skill of deep learning models.

Specifically, you learned:

- The common mistake made by beginners when evaluating deep learning models.
- The rationale for using repeated k-fold cross validation to evaluate deep learning models.
- How to calculate related model skill statistics, such as standard deviation, standard error, and c

Do you have any questions about estimating the skill of deep learning models?
Post your questions in the comments and I will do my best to answer.

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional develo to helping developers get started and get good at applied machine learning. Learn more.
View all posts by Jason Brownlee →

‹ 7 Ways to Handle Large Data Files for Machine Learning

How to Report Classifier Performance with Confidence Intervals ›

Get Your Start in Machine Learning

8 Responses to *How to Evaluate the Skill of Deep Learning Models*

Vikas May 31, 2017 at 2:40 pm #

REPLY

How is running trainings times on same dataset different than running training through same data with multiple epochs ?

Can I say that by doing multiple epochs of data with deep learning serves the same purpose to reduce variation in results due to the stochastic nature of the Deep learning algorithm.

Jason Brownlee June 2, 2017 at 12:44 pm #

Each epoch updates the network weights.

Each run through all epochs results in a different model given different random initial conditions.

Does that help?

SANTOSH GAWDE June 5, 2017 at 2:14 am #

I have a strong background in ERP consulting services industry but I have not done programmi
learning machine I can utilize it to start small startup business.

Jason Brownlee June 5, 2017 at 7:42 am #

REPLY

Yes. Consider starting with Weka to learn applied machine learning without any programming:
http://machinelearningmastery.com/start-here/#weka

Xu Zhang June 7, 2017 at 10:27 am #

REPLY

Thank you so much for your great article. Would you mind me asking three questions?

1. Could you please tell me when we do an evaluation of the Skill of Deep Learning Models? Before hyperparameter tuning or after having selected the best model? Based on my understanding, I chose after. Please correct me if I am wrong.

2. For deep learning, it is such a time-consuming process. If we have to do 30, even 100 or 1000 repeats, is it feasible for a real project?

3. If the final model is an ensemble model which includes several different models, how can I evaluate the skill of the model?

Jason Brownlee June 8, 2017 at 7:36 am #

1. Evaluating the skill in any circumstance.

2. It very well is not feasible for a large project and using checkpointing to save the “best” model see

3. In theory, the same approach could be used.

Ravi Sharma June 22, 2017 at 11:51 pm #

A little off-topic but when do you think the deep learning bubble will pop?

Jason Brownlee June 23, 2017 at 6:42 am #

REPLY

When the techniques stop delivering value to business in general, or practitioners continue to fail to deliver the value promised.

Leave a Reply

Get Your Start in Machine Learning

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

Finally Get Started With Deep Learning

Sick of the fancy math and need for super computers?
Looking for step-by-step tutorials?

Get Your Start in Machine Learning

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

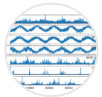
Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

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

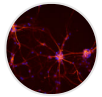
Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

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

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



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



How to Implement the Backpropagation Algorithm From Scratch In Python
NOVEMBER 7, 2016

Get Your Start in Machine Learning