Online Evaluation of Machine Learning Models

Ted Dunning
CTO
What you’ll learn

Monitoring machine learning-based systems is different from monitoring conventional systems. Attendees will come away with an understanding of the difference as well as some practical methods for monitoring real-world systems.

Description

Academic machine learning involves almost exclusively off-line evaluation of machine learning models. In the real-world this is, somewhat surprisingly, often only good enough for a rough cut that eliminates the real dogs. For production work, online evaluation is often the only option to determine which of several final round candidates might be chosen for further use. As Einstein is rumored to have said, theory and practice are the same, in theory. In practice, they are different. So it is with models. Part of the problem is interaction with other models and systems. Part of the problem has to do with variability of the real world. Often, there are adversaries at work. It may even be sunspots. One particular problem arises when models choose their own training data and thus couple back onto themselves.

In addition to these difficulties, production models almost always have service level agreements that have to do with how quickly they must produce results and how often they are allowed to fail. These operational considerations can be as important as the accuracy of the model ... right results returned late are worse than slightly wrong results returned in time.

I will provide a survey of useful ways to evaluate models in real world use. This will include the use of decoy and canary models, non-linear latency histogramming, model-delta diagrams and more. These techniques may sound arcane, but each has a simple heart and should not require any advanced mathematics to understand.
Contact Information

Ted Dunning, PhD
Chief Technology Officer, MapR Technologies
Board member, Apache Software Foundation
O’Reilly author

Email  tdunning@mapr.com    tdunning@apache.org

Twitter @ted_dunning
Agenda

- Why this is much harder than it looks
  - Reference to cows
- Why cross validation and offline evaluation often don’t work
  - Explore versus exploit
- Decoys and canaries
  - Quick rendezvous
- Comparing models
- Keeping an eye on the basics
- Recap and Q&A
Assume a cow is a radially symmetrical sphere
Assume a cow is a radially symmetrical sphere.

Modeling the cow is now simpler.
Assume a cow is a radially symmetrical sphere

Modeling the cow is now simpler but it can’t walk or eat
Assume a cow is a radially symmetrical sphere

Modeling the cow is now simpler fine for modeling cows in orbit
Assume we can do offline evaluation of models
Assume we can do offline evaluation of models

(Academic) life is now much easier
Assume we can do offline evaluation of models

(Academic) life is now much easier but we ignore important realities
Let’s talk about why
Let’s talk about why

Examine this artistic work
Cat Wearing A Shark Costume Cleans The Kitchen On A Roomba. Shark Week. #SharkCat cleaning Kitchen!

12,689,042 views  43K  1.5K  SHARE  SAVE
24 MAGICAL FOOD TRICKS YOU HAVE TO TRY
5-Minute Crafts
Recommended for you

Cat meeting the puppies for the first time.
Stefan Atkinson
12M views

Raven's personality compared to Human Narcissist
Peter Caine Dog Training
466K views

Silver man secret revealed from start to finish, floating and levitating trick
Education
Recommended for you
Training data → Training/test split

Training/test split → Machine learning (train)

Machine learning (train) → Score (model)

Score (test) → New data
model

New data → Score → Viewer reactions
Many models choose their own training data
Many models choose their own training data

What they do today is what they learn from tomorrow
The crux is a choice between exploiting current knowledge and exploring for new knowledge.
Quick thought:
Quick thought:

Worse can be better
Result Dithering

Dithering is used to re-order recommendation results
  • Re-ordering is done randomly

Dithering is guaranteed to make off-line performance worse

Dithering also has a near perfect record of making actual performance much better
Result Dithering

Dithering is used to re-order recommendation results
  • Re-ordering is done randomly

Dithering is *guaranteed* to make off-line performance worse

Dithering also has a near perfect record of making actual performance much better

“*Made more difference than any other change*”
Why Use Dithering?

- Scrolling required at the fold
- Very few people click to second page
- Can't learn about these if nobody sees them
Simple Dithering Algorithm

Synthetic score from log rank plus Gaussian

\[ s = \log r + \mathcal{N}(0, \log \epsilon) \]

Pick noise scale to provide desired level of mixing

\[ \frac{\Delta r}{r} \propto \epsilon \]

Typically

\[ \epsilon \in [1.5, 3] \]

Also… use \([t/T]\) as seed
\[ \epsilon = 1 \]
The good news is that we can make a model better (long term) by adding noise (making it worse in the short term)
The bad news is off-line testing is the ultimate short-term test

It can’t distinguish useful exploration from bad results
This is profoundly depressing

We need some help
But first,
But first,

some bad news
Evaluating models is even harder than it looks

Why not try testing on live data?
Why not use A/B testing to find out which of A or B is better?

Because it doesn’t work like that

Take a good exploiter (A) and a wild explorer (B) and run a test with 95% A and 5% B
Result is likely to be that A will perform better by learning from B’s exploration
Killing B will make A get worse (no exploration)
Killing A will give us a lousy model (no exploitation)
Watchfulness is key

This doesn’t mean we can’t test models
we just need more care than you might think at first

Steps for testing

1. Offline testing is still useful. Look for gross failures, look at differences
2. Online difference testing is still useful. Look for large differences
3. Cautious changes in A/B volumes can work well
   a. Look for changes versus historic performance dependent on bandwidth change
   b. Consider isolating and comparing
      A trains A1, A+B trains A2, A+B trains B
4. Key is to detect changes
So how can we simul-cast multiple models?
Rendezvous architecture
Recording Raw Data (as it really was)
Quality & Reproducibility of Input Data is Important!

Recording raw-ish data is really a big deal
  • Data as seen by a model is worth gold
  • Data reconstructed later often has time-machine leaks
  • Databases were made for updates, streams are safer

Raw data is useful for non-ML cases as well (think flexibility)

Decoy model records training data as seen by models under development & evaluation
Canary for Comparison
What Does the Canary Do?

The canary is a real model, but is very rarely updated
The canary results are almost never used for decisioning

The virtue of the canary is stability

Comparing to the canary results gives insight into new models
Key point: stream-first architecture allows multiple live models
Key point: rendezvous architecture allows exact recording of inputs and outputs
Key point: rendezvous architecture allows live comparison versus the canary model
How to find change

Basic idea is that histograms let us get counts
change in distribution will result in different count proportions

Implementation via t-digest or LogHistogram
we can find reference quantiles at (say) 0.9, 0.99, 0.999, 0.9999
then use those quantiles to probe counts from test data

Compare counts using g-test
2 x n array of counts => score
Score distribution monitoring
More change monitoring

We can repeatably train models
  if we can version control all training code and parameters (yes, we can)
  if we can version control all training data (yes, with platform help)

With those repeatable model builds
  we can build A1 = learn(A), A2 = learn(A + B), B1 = learn(A+B)
  if A1 and A2 produce essentially identical scores, B is not aiding A
    and A2 versus B1 is probably a fair comparison
With no synergy, direct comparison makes sense
More with rendezvous

Score quality is only part of the game

The other part is system reliability

Same monitoring techniques can be used

Monitor latency, monitor rendezvous branch proportions
Score/latency/whatever distribution is the primary unit of monitoring
Let’s talk about how to do that
Cumulative distribution is key
Cumulative distribution is key
Cumulative distribution is key

KS test is standard go-to for comparing distributions, in general …

But we need much more focus
Transpose that graph and look again at just the top end

(because that’s what Gil did)
Latency values for TCP round-trip

- No loss
- 1% loss
- 10% loss
Select quantile cut points

Latency (ms)

Quantile

10% loss
1% loss
No loss
This gives latency cut points at 5, 70, 250 ms
And that gives us counts for red and black data
Quick results of method

For these conditions,

Comparing 100 samples of red/green versus 100,000 green samples
   Average score of roughly 70 for red versus 2.7 for green
   Probably of detecting red with 100 samples is near 100%
   Probability of false positive is near 0

Even with 10 samples, probability of detecting difference is 80%
Scoring 100 samples gives easy decision with low error
Summary

Model evaluation *can* be much harder than it seemed due to training data loop

- Offline evaluation is a fine rough cut, but not much more
- A/B testing is subject to crossfeed
- Rendezvous helps with monitoring
- Proper answers come from careful score monitoring
  - Current versus past
  - Current versus canary
  - With and without challenger data
- You have to monitor to ensure you meet site reliability guarantees as well
Summary

Model evaluation *can* be much harder than it seemed due to training data loop

- Offline evaluation is a fine rough cut, but not much more
- A/B testing is subject to crossfeed

Rendezvous helps with monitoring

Proper answers come from careful score monitoring

- Current versus past
- Current versus canary
- With and without challenger data

You have to monitor to ensure you meet site reliability guarantees as well
Summary

Model evaluation *can* be much harder than it seemed due to training data loop

- Offline evaluation is a fine rough cut, but not much more
- A/B testing is subject to crossfeed

Rendezvous helps with monitoring

Proper answers come from careful score monitoring

- Current versus past
- Current versus canary
- With and without challenger data

You have to monitor to ensure you meet site reliability guarantees as well
Summary of methods

You need quantile sketching

- $t$-digest is a fine quantile sketch and very general
- non-linear histogram (LogHistogram, HdrHistogram) is useful for latencies

And you need test of distribution

- g-test compares counts very well
- KS-test focuses on the wrong thing
Summary of methods

You need quantile sketching

t-digest is a fine quantile sketch and very general

non-linear histogram (LogHistogram, HdrHistogram) is useful for latencies

And you need test of distribution

g-test compares counts very well

KS-test focuses on the wrong thing
Additional Resources: Available Now

O’Reilly book by Ted Dunning & Ellen Friedman © June 2014

Read free courtesy of MapR:

Additional Resources: Available Now

O'Reilly report by Ted Dunning & Ellen Friedman © March 2017
Read free courtesy of MapR:


O'Reilly book by Ted Dunning & Ellen Friedman © March 2016
Read free courtesy of MapR:

https://mapr.com/streaming-architecture-using-apache-kafka-mapr-streams/
Previous book: how to manage machine learning models

Machine learning deserves special techniques

Download free pdf or read free online via @MapR: https://mapr.com/ebook/machine-learning-logistics/
AI & Analytics in Production: How to Make It Work

Download free pdf via MapR: https://mapr.com/ebook/ai-and-analytics-in-production/
Contact Information

Ted Dunning, PhD
Chief Technical Officer, MapR Technologies
Board member (one more day), Apache Software Foundation
O’Reilly author

Email  tdunning@mapr.com        tdunning@apache.org

Twitter  @ted_dunning