Machine Learning and Tabletop Science

Peter W. Bryant
IBM Research, Rio de Janeiro

pbryant@br.ibm.com

2018 CBPF Python Summer Camp

27 Feb. 2018

Outline

1. Machine Learning in science
2. Experiment to benchmark
3. Classical analysis
4. Machine Learning analysis
5. Discussion

Outline

1 Machine Learning in science

2 Experiment to benchmark

3 Classical analysis

4 Machine Learning analysis

5 Discussion

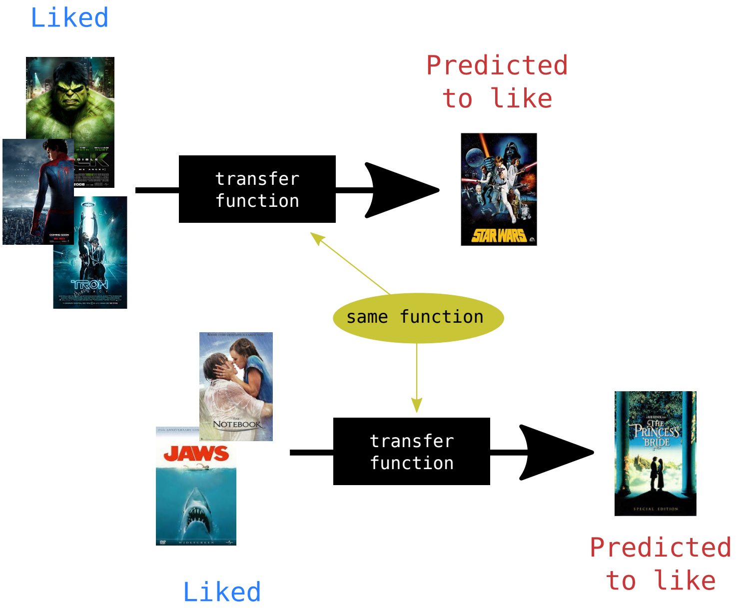
Machine Learning Uses

- In Mature (and Maturing) Technologies
  - medical diagnosis
  - language translation & processing
  - recommendation systems

- In Science
  machine learning can and will increasingly be exploited at **"every stage of the scientific process"**[1]

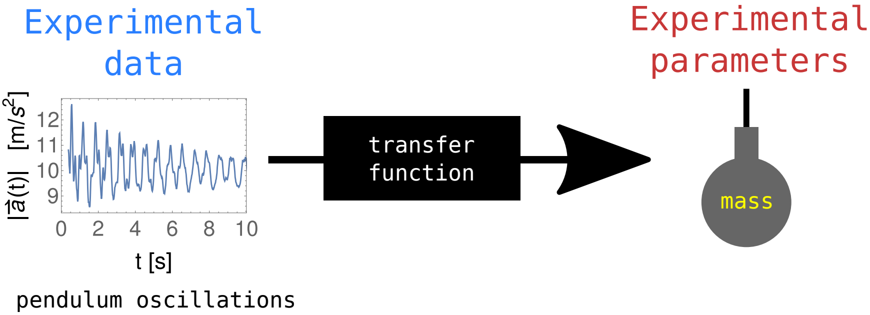
[1] Mjolsness and DeCoste, *Science*, 293(5537):20512055, 2001.

What Does it Do? An Example

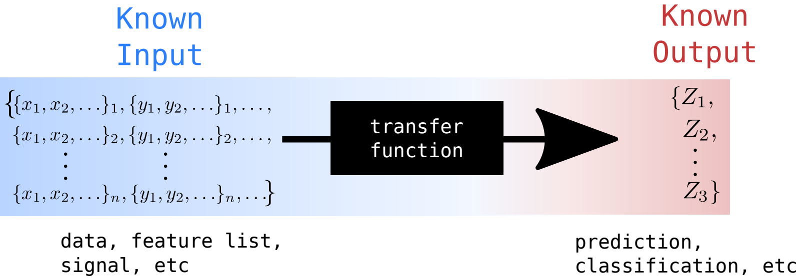


In Science

- Not limited to movies
- Can be experimental data



The (Supervised) Learning Part



- Learning from training:

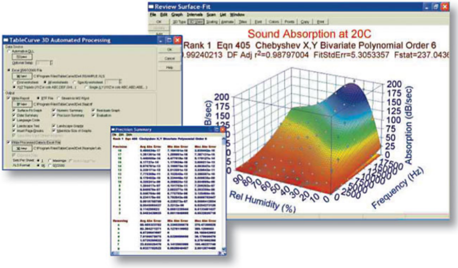
  Using as many as possible, known Input→Output pairs,

  automatically find **transfer function** that maps **any** input

  to the *best possible approximation* of output

“Automatically”

TableCurve 3D – Model Complex
Data Sets Fast and Easy



Eliminate Tedious Data Analysis Chores with TableCurve 3D

TableCurve 3D uses a selective subset procedure to fit 36,000 of its 453,697,387 built-in equations from all disciplines to find the one that provides the ideal fit – instantly!

What once could take days of tedious work now takes minutes, with a much more powerful result.

In Tabletop Science

Interesting implications for observational or tabletop science

- Pros
  - not explicitly programmed
  - can be effective
    even when observed signal is not understood

- Cons
  - lack of understanding
    why it does or does not work
  - uncertainty, accuracy & precision not well defined

To be useful, it must be benchmarked!

Outline

1 Machine Learning in science

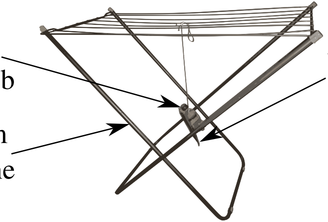
2 Experiment to benchmark

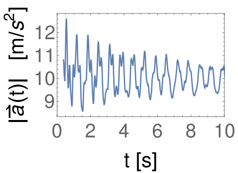
3 Classical analysis

4 Machine Learning analysis

5 Discussion

To benchmark, we need an experiment we **understand**.

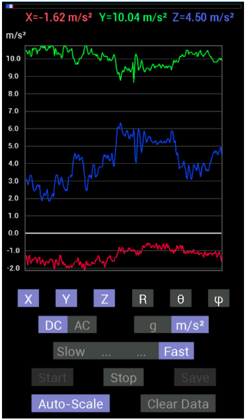


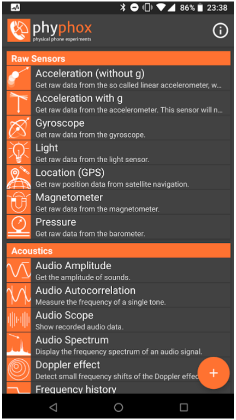


Pendulum on Flexible Structure

- vary the mass of the pendulum by adding water to the bottle
- via phone's accelerometer observe $|\vec{a}(t)|$

Data Acquisition





- Accelerometer Meter App
- buggy

- phyphox App
- http://phyphox.org/

Experiment and Challenge

Hypothesis

Based on $|\vec{a}(t)|$, one can "predict" the Δm added to the pendulum.

. . . physical understanding *vs.* black box. . .

. . . Pete *vs.* ML. . .

. . . human *vs.* machine . . .

Outline

1 Machine Learning in science

2 Experiment to benchmark

3 Classical analysis

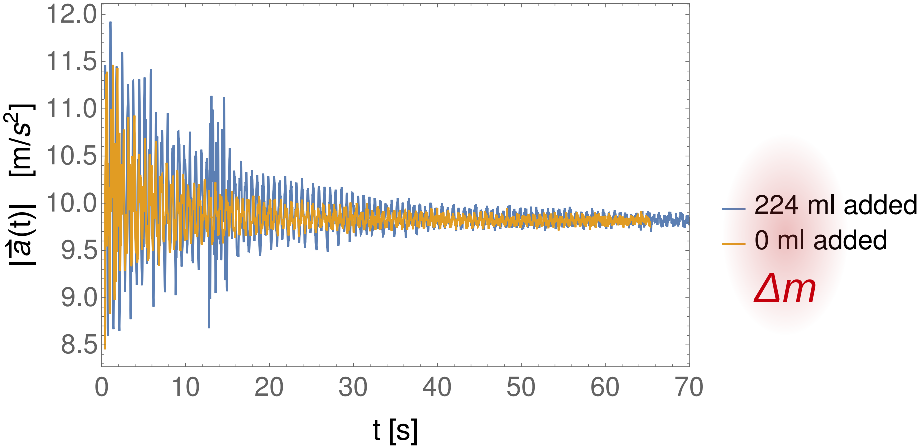
4 Machine Learning analysis

5 Discussion

Classical Analysis

By Human (me)

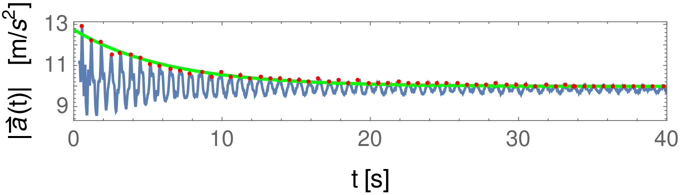
Acceleration Damping



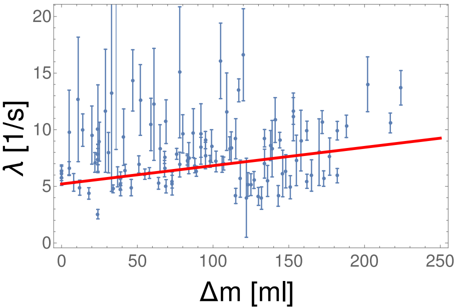
Idea: determine Δm based on damping rate

Acceleration Damping

- fit $h + ae^{-t/\lambda}$ to the peaks for each of $n = 107$ different Δm

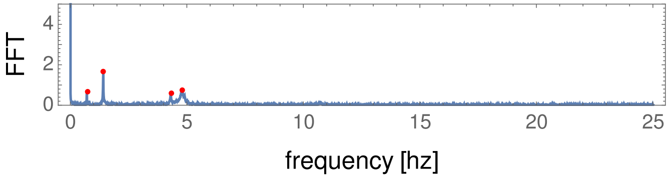


- is there a good enough $\lambda(\Delta m)$?

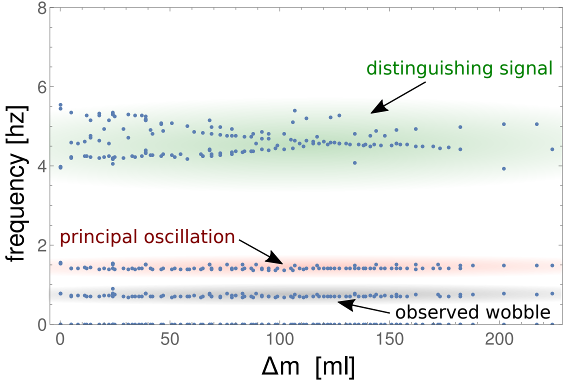


- error bars reflect 0.95 confidence in fit to peaks
- linear fit weighted by (error bar)$^{-2}$

Frequency Analysis

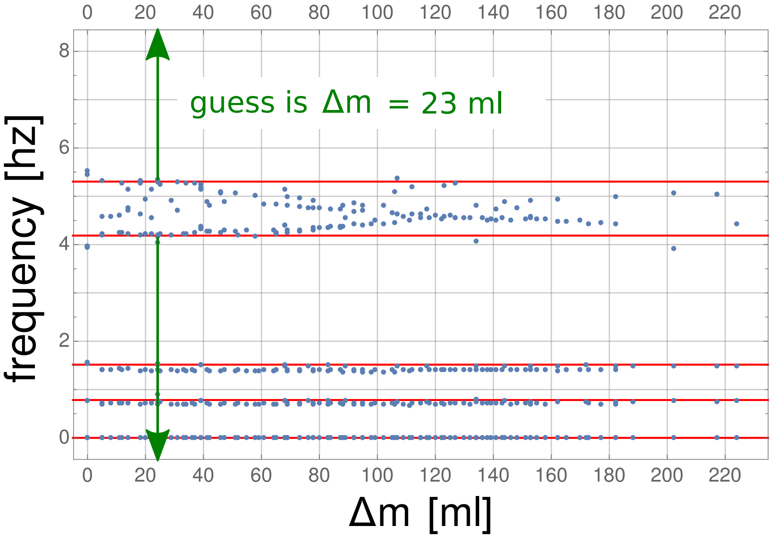


- investigate peak position as a function of Δm



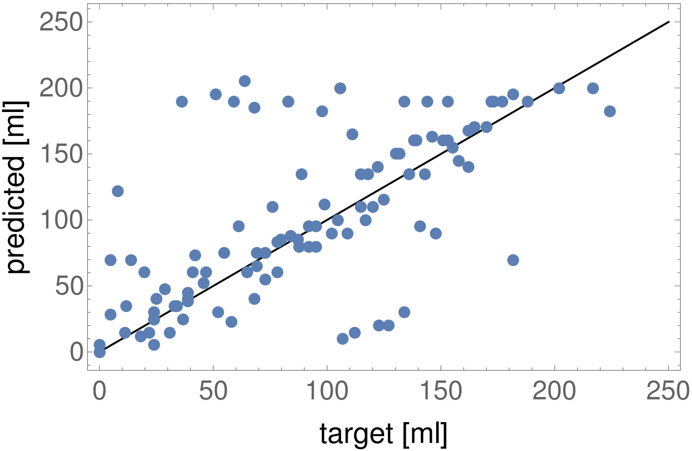
Frequency Analysis

- Procedure: eliminate one sample at random from the plot and try to identify its Δm



Frequency Analysis

- Result of frequency analysis (human)



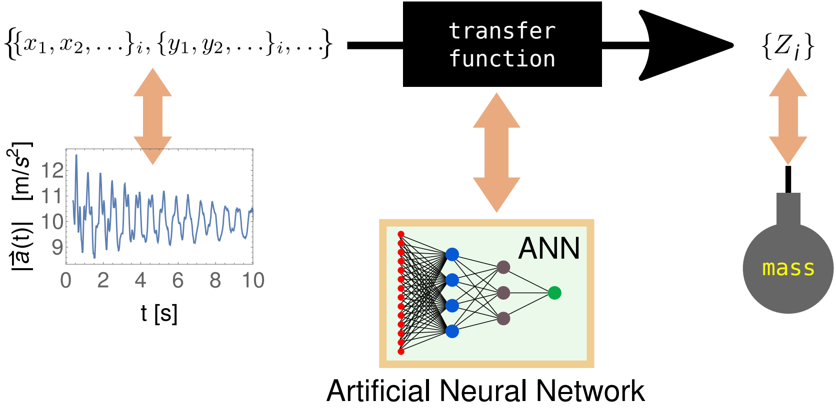
- Will compare this with result from Machine Learning

Outline

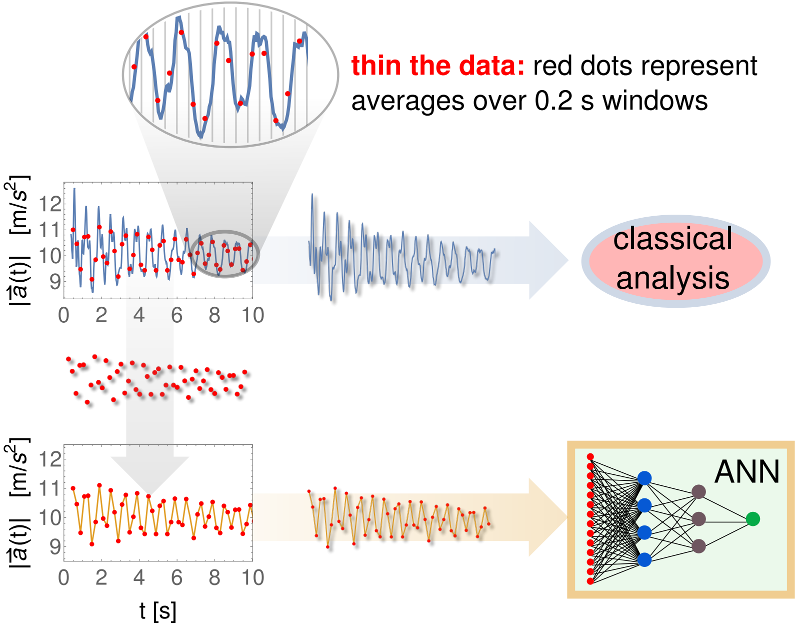
1. Machine Learning in science
2. Experiment to benchmark
3. Classical analysis
4. Machine Learning analysis
5. Discussion

By Machine (🤖)

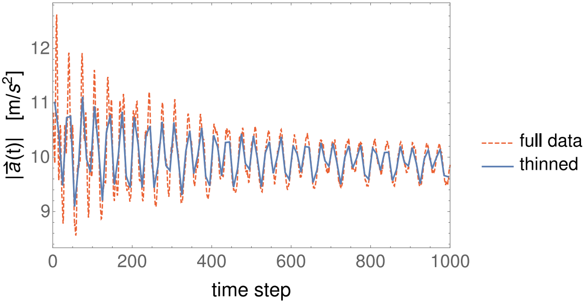
Neural Net as the Black Box



Input Data

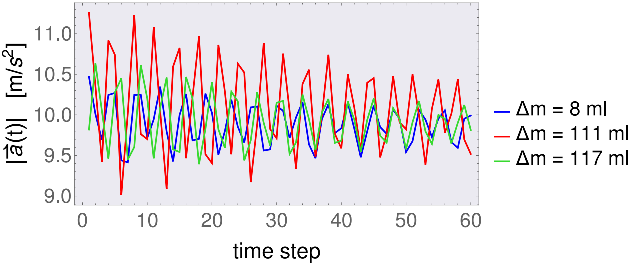

thin the data: red dots represent averages over 0.2 s windows
$|\vec{a}(t)|$ [m/s^2]
12
11
10
9
0 2 4 6 8 10
classical analysis
$|\vec{a}(t)|$ [m/s^2]
12
11
10
9
0 2 4 6 8 10
t [s]
ANN

Input Data



- Example of thinning

- Is there a Pattern?



Implementation: Structure

- Mathematica version 11

```
ann = NetChain[{8, Cos, 4, SummationLayer[]}, "Input" → 60] (* define net *)
```

NetChain[

	Input	vector (size: 60)
1	LinearLayer	vector (size: 8)
2	Cos	vector (size: 8)
3	LinearLayer	vector (size: 4)
4	SummationLayer	real
	Output	real

(uninitialized)

]

- label networks with lists describing structure: {**8**,**Cos**,**4**}
  - for the experts, these lists alternate
    dimension of a linear layer, and
    function applied to each element of a layer

Implementation: Training

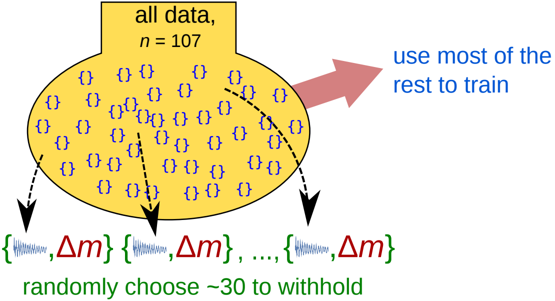
```
NetTrain[ann, trainingData] (* to train *)

trainingData[[12 ;; 14]] (* input data for three of the 107 tests *)

{{1.14562, -0.207284, -0.466552, 1.01765, 0.18692, -0.917693, 0.496379, 0.840023,
  -0.846064, -0.330171, 0.95813, -0.148744, -0.466055, 0.826876, 0.184943, -0.862647,
  0.243687, 0.658004, -0.512469, -0.074549, 0.788917, -0.376339, -0.616019,
  0.486098, 0.286579, -0.555809, 0.22177, 0.5065, -0.655119, -0.417777, 0.597801,
  -0.00837573, -0.410878, 0.452254, 0.120329, -0.792143, -0.0763571, 0.551581,
  -0.291638, -0.174894, 0.482475, -0.244652, -0.592974, 0.427831, 0.269625, -0.621387,
  -0.153248, 0.39723, -0.261063, -0.236525, 0.352168, -0.188027, -0.523538,
  0.236691, 0.167431, -0.607959, -0.188227, 0.335691, -0.205893, -0.208135} → 224,
 {1.10862, 0.509886, -0.708956, 0.664948, 0.770939, -0.931129, -0.176914, 1.14216,
  -0.22917, -0.528597, 0.942038, 0.136561, -0.965027, 0.437942, 0.782123, -0.638582,
  -0.0454665, 0.834143, -0.452232, -0.569118, 0.816586, 0.162004, -0.776123,
  0.304396, 0.474985, -0.604444, -0.0142264, 0.724935, -0.415539, -0.572572,
  0.501186, 0.0681227, -0.549794, 0.317755, 0.341629, -0.631694, -0.191605, 0.468102,
  -0.312153, -0.35029, 0.456545, -0.0491232, -0.591027, 0.125816, 0.205136, -0.495011,
  -0.0751806, 0.397468, -0.34284, -0.389028, 0.272049, -0.121962, -0.486495,
  0.172175, 0.172165, -0.47167, -0.161412, 0.195697, -0.358447, -0.300288} → 76,
 {0.652449, 0.953987, -0.651745, 0.156728, 1.05708, -0.525574, -0.565353, 1.11061,
  0.163903, -0.921972, 0.43355, 0.593627, -0.637385, 0.120172, 0.876946, -0.517878,
  -0.644989, 0.73621, 0.157167, -0.549981, 0.485022, 0.3859, -0.818013, -0.115633,
  0.71834, -0.30111, -0.373566, 0.615491, -0.0440685, -0.630154, 0.327604, 0.321961,
  -0.604172, -0.0416889, 0.557419, -0.332296, -0.339326, 0.454639, -0.136797,
  -0.550573, 0.307234, 0.251552, -0.53006, -0.0991151, 0.304945, -0.379797,
  -0.256908, 0.421445, -0.111945, -0.530935, 0.0768261, 0.0964023, -0.375272,
  0.0271001, 0.257772, -0.421114, -0.3975, 0.188928, -0.0910437, -0.315047} → 83}
```

Training and Evaluation Routine

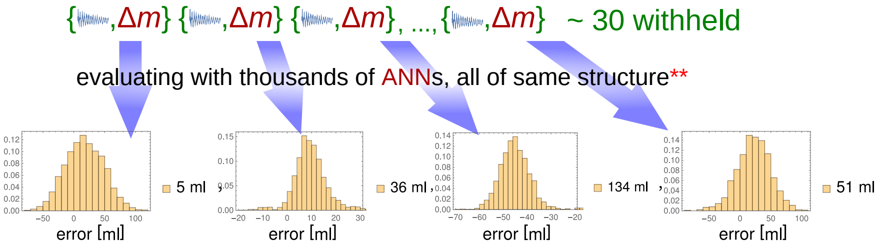
- withhold approximately 30 tests and use the rest to train ANN



- evaluate ANN on these withheld tests

- $n = 107$ is *very small* for machine learning applications, so repeat **thousands** of times for
  - fixed set of withheld tests
  - fixed ANN structure

Training and Evaluation Routine

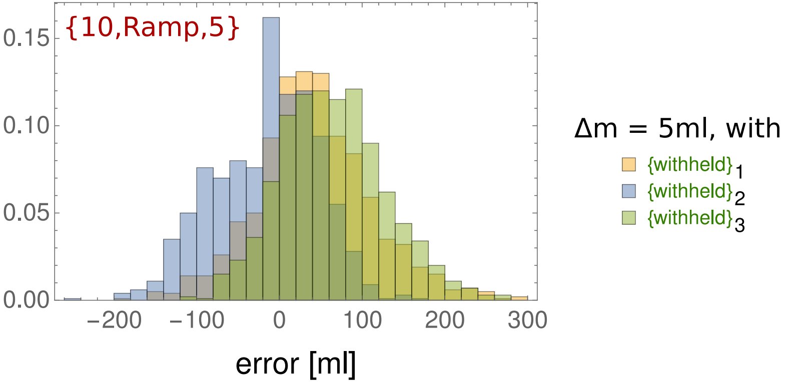


**nets differ in "trained" parameters*

- distribution mean for test i and fixed ANN, {withheld}:

$$< Z_i > \Big|_{ANN,\{withheld\}}$$

Training and Evaluation Routine

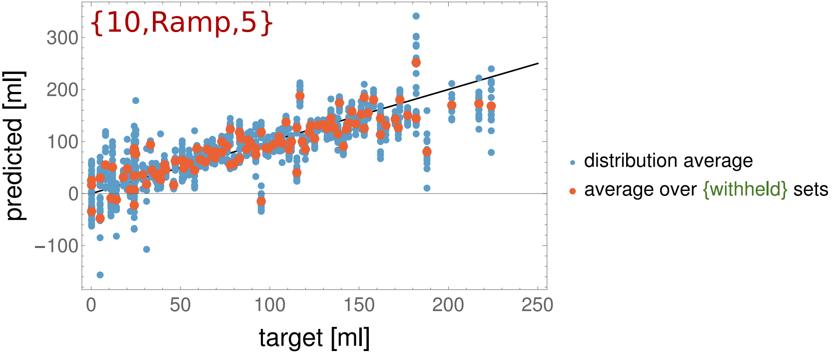


- distribution for test *i* depends on {withheld} set used for training

Training and Evaluation Routine

- average over over varying sets of {withheld} to get

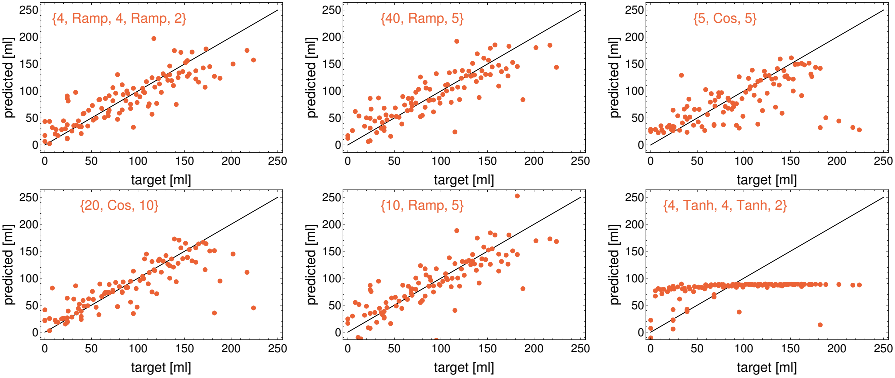
$$< Z_i > \Big|_{ANN}$$



- do not average over various structures of ANN **yet**

Constructing a Weighting Scheme

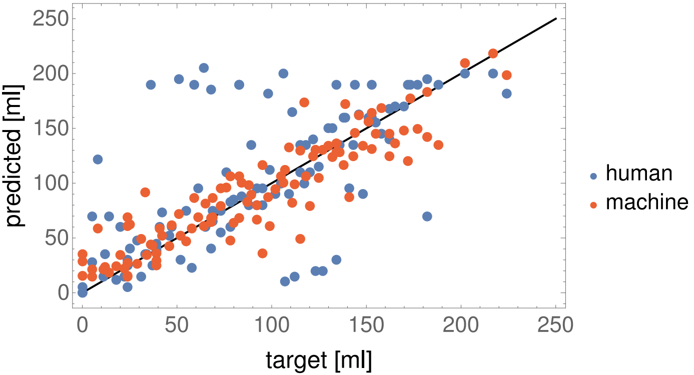
- predictions depend heavily on net structure



- use predictions from different nets to weight the average

Weighted Average for Final Result

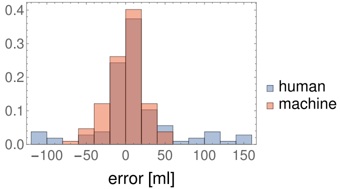
- weighted average, $< Z_i >$, from Machine Learning



- $< Z_i >$ compares favorably with human results

The Champion

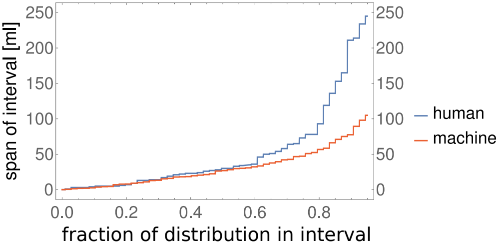
- error distributions based on 107 tests



human average error: 9.4 ml

machine average error: −0.2 ml

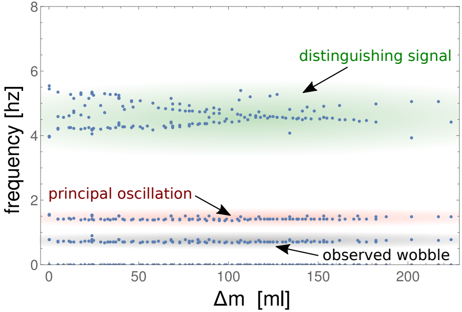
- measure of the span of the distribution



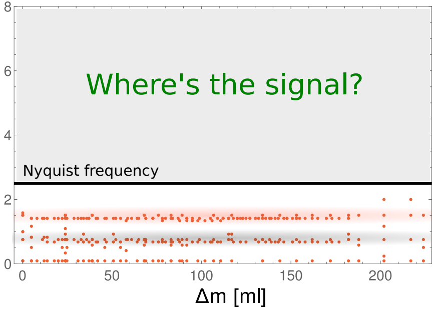
Winner: machine 

The Mystery

- Because of the averaging over 0.2 s windows,
  the machine cannot use the signal I used.



What I saw in the frequency



What the machine could see

Outline

1. Machine Learning in science
2. Experiment to benchmark
3. Classical analysis
4. Machine Learning analysis
5. Discussion

Discussion

- Small n and noise are representative of tabletop science.
- Hypothesis that many nets can be used in place of many data was verified qualitatively.
- Machine performance depends on input data (feature selection). Window average worked well; many did not.
- Machine seemed to handle uncertainty in the data better than did the human, though I have not quantified this yet.
- Training hundreds of thousands of nets requires several weeks but is not labor intensive. The labor intensive classical analysis requires less than a day.

Future Work

- Secure funding so I can negotiate more time with the equipment!

Thank you!

And special thanks to

Yuri Lira, Maria Moura, Jaione Tirapu-Azpiroz, Cicero Nogueira Dos Santos, Joel Luís Carbonera, Mathias Steiner

Contact: pbryant@br.ibm.com