Methodology of Learning

Thanks to Jude Shavlik

Proper Experimental Methodology Can Have a Huge Impact!

A 2002 paper in *Nature* (a major journal) needed to be corrected due to "training on the testing set"

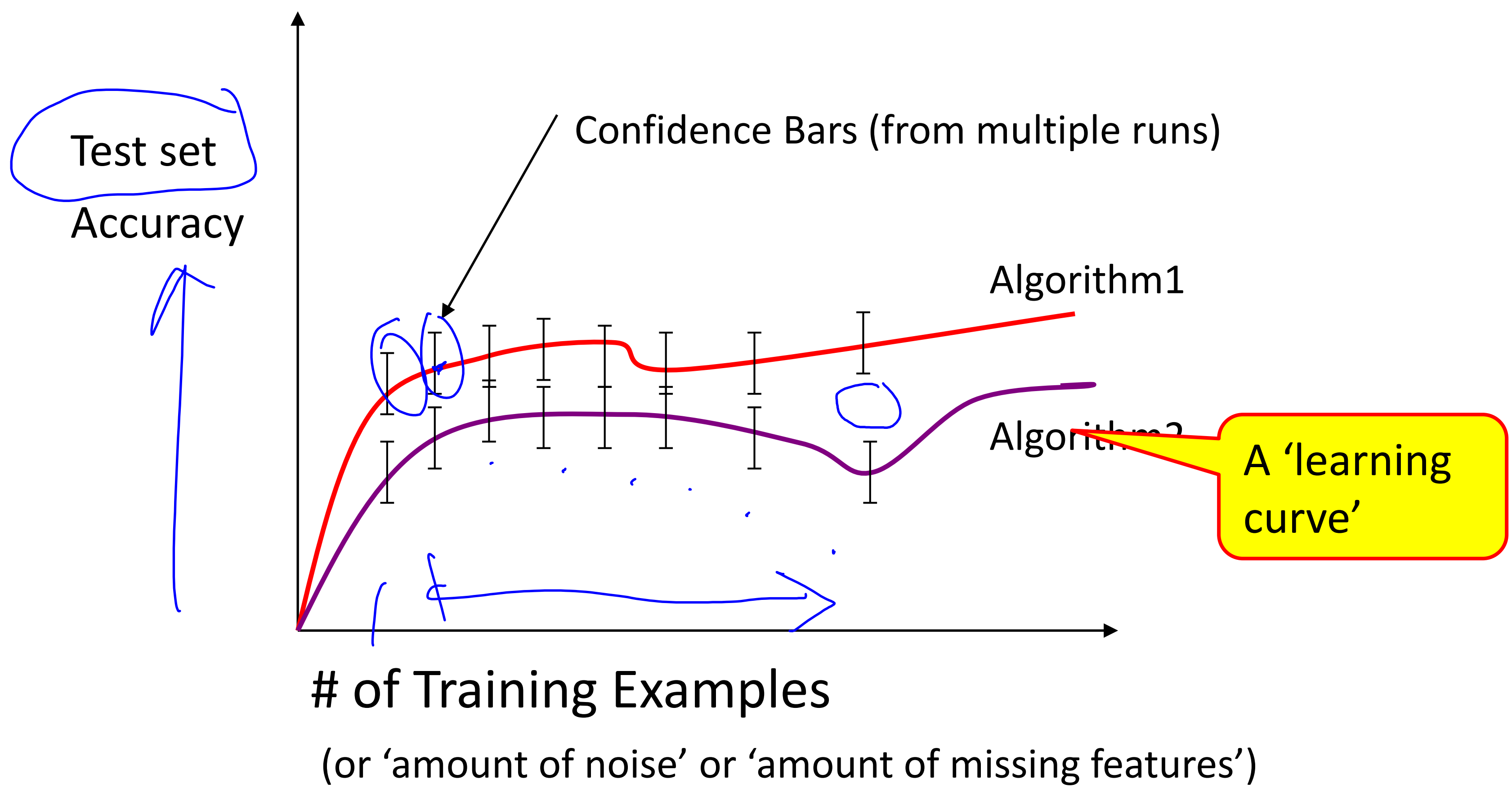
Original report : 95% accuracy (5% error rate)

Corrected report (which still is buggy):
73% accuracy (27% error rate)

Error rate increased over 400%!!!

Some Typical ML Experiments – Empirical Learning

no stat sig



Typical Experiments

	Testset Performance
Full System	80%
Without Module A	75%
Without Module B	62%

Experimental Methodology

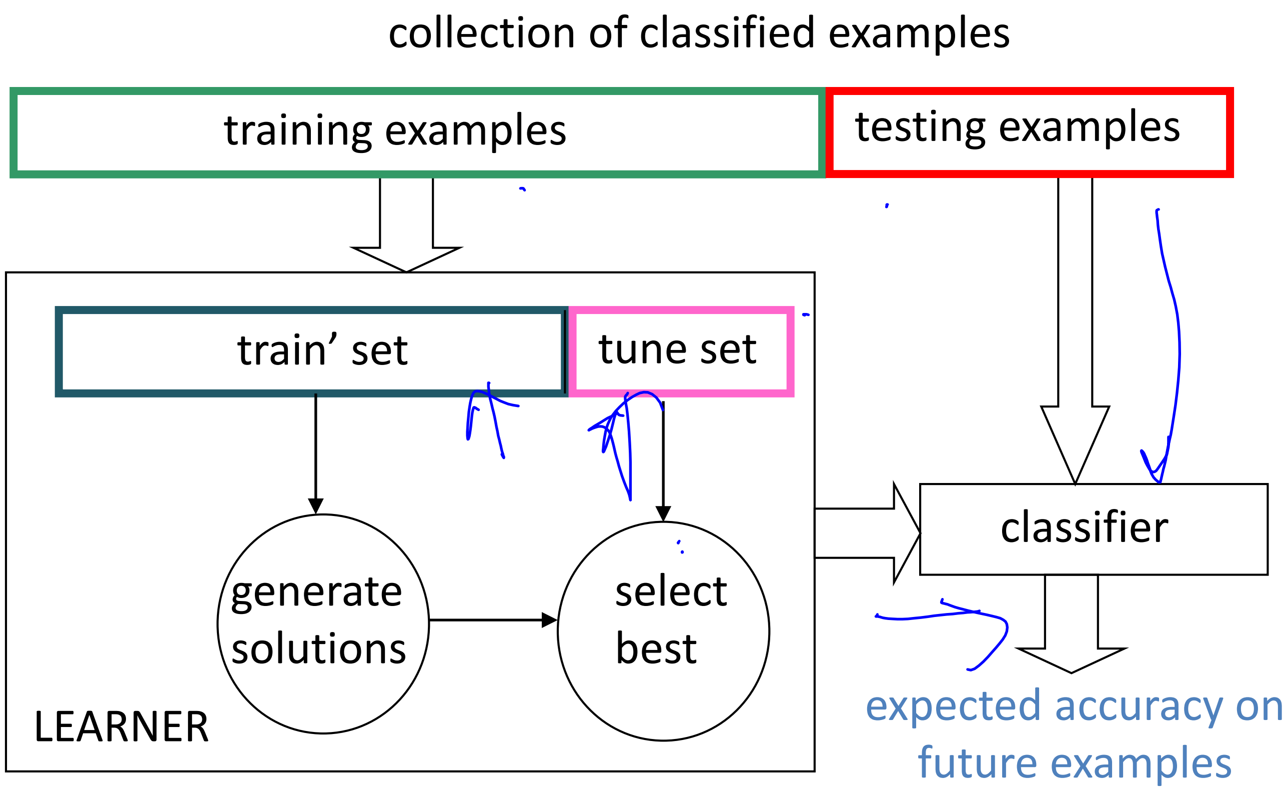
1) Start with a dataset of labeled examples
2) Randomly partition into *N* groups
3a) *N* times, combine *N* -1 groups into a train set
3b) Provide train set to learning system
3c) Measure accuracy on "left out" group (the test set)

Called *N*-fold cross validation (typically *N* =10)

Tuning Set

- Often, an ML system has to choose when to stop learning, tuning
  select among alternative answers, etc.
- One wants the model that produces the highest accuracy on **future** examples ("overfitting avoidance")
- It is a **"cheat"** to look at the **test** set while still learning
- Better method
  - Set aside part of the training set
  - Measure performance on this "tuning" data to estimate future performance for a given set of parameters
  - Use best parameter settings, train with **all** training data (except **test** set) to estimate future performance on **new** examples

A typical Learning system



Statistical techniques such as 10-fold cross validation and *t*-tests are used to get meaningful results

Parameters

Notice that each train/test fold may get <u>different</u> parameter settings!

That's fine (and proper)

I.e. , a "parameterless"* algorithm internally sets parameters for **each data set** it gets

* Usually, though, some parameters have to be externally fixed (e.g. knowledge of the data, range of parameter settings to try, etc)

Multiple Tuning sets

Using a **single** tuning set can be unreliable predictor, plus some data "wasted."
Hence, often the following is done:

1) For each possible set of parameters
   a) Divide <u>training</u> data into **train'** and **tune** sets, using ***N*-fold cross validation**
   b) Score this set of parameter values: average **tune** set accuracy over the *N* folds
2) Use **best** set of parameter settings and **<u>all</u> (train' + tune)** examples
3) Apply resulting model to **test** set

Number of Features and Performance

- Too many features can hurt test set performance
- Too many irrelevant features mean many spurious correlations for a ML algorithm to detect

  "Curse of dimensionality"

Feature Selection and ML
(general issue for ML)

Filtering-Based Feature Selection (FS)

all features

↓

FS algorithm

↓

subset of features

↓

ML algorithm

↓

model

Eg, use 'info provided' to score features

Wrapper-Based Feature Selection

all features → FS algorithm

calls ML algorithm many times, uses it to help select features

ML algorithm

→ model

Feature Selection as a Search Problem

<u>State</u> = set of features

Start state = *empty* (forward selection)
or *full* (backward selection)

Goal test = highest scoring state

<u>Operators</u>

add/subtract a feature

<u>Scoring function</u>

accuracy on training (or tuning) set of
ML algorithm using this state's feature set