Machine Learning in Formal Verification

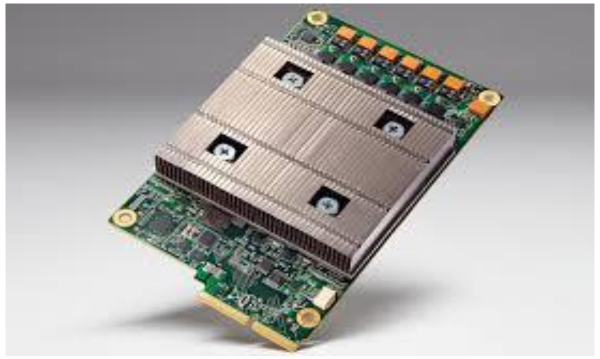
FMCAD 2016 Tutorial

Manish Pandey, PhD

Chief Architect, New Technologies

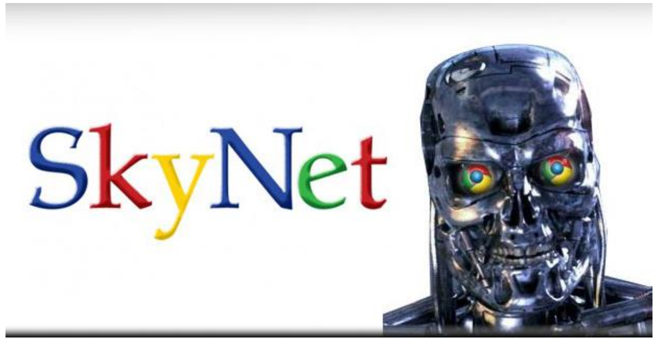
Synopsys Verification Group

Race for AI



nervana

Google DeepMind
Challenge Match
MATCH 2
AlphaGo vs Lee Sedol
Google DeepMind


SkyNet

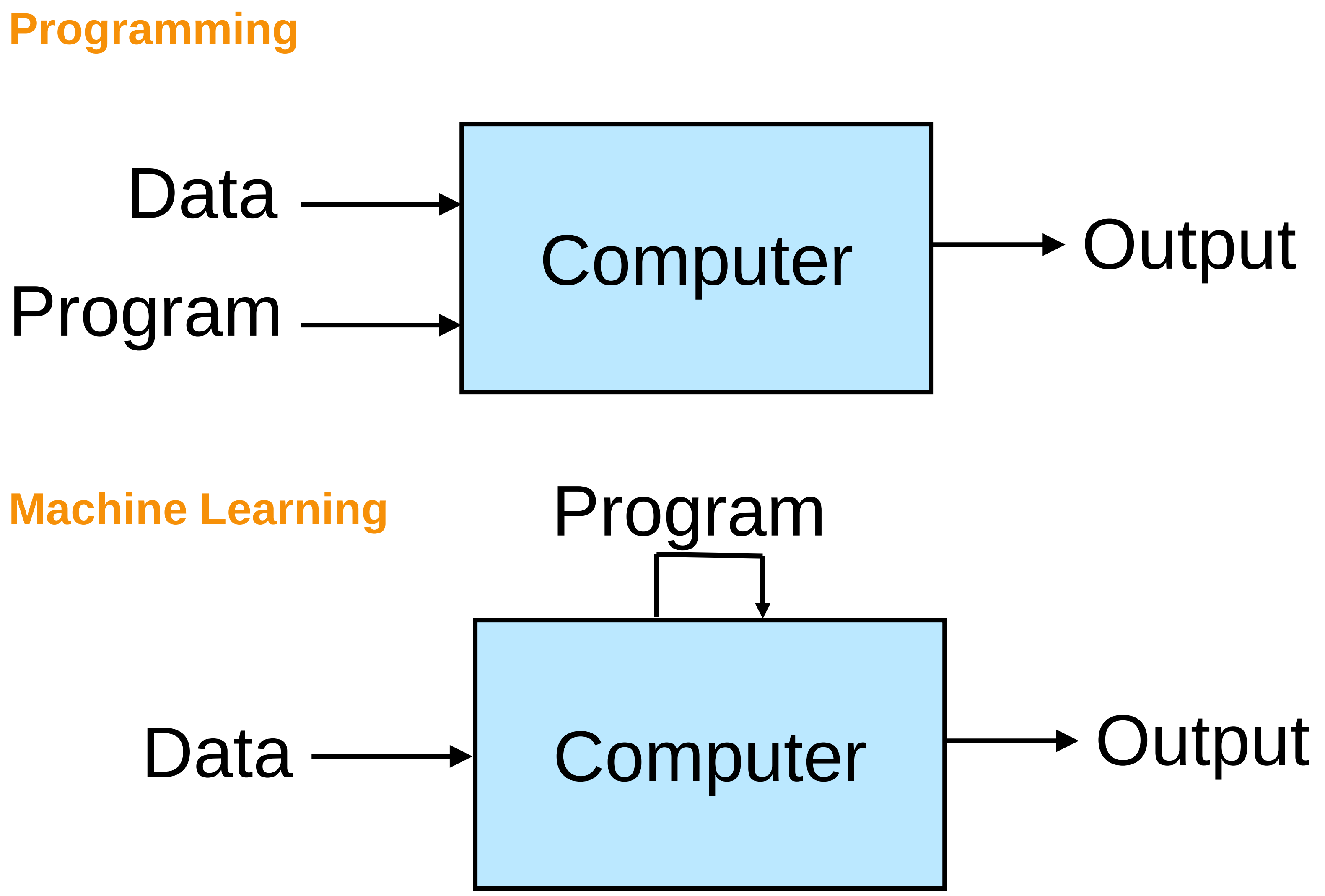
What is Machine Learning?

“Learning is any process by which a system improves performance from experience”
Herbert Simon

“The complexity in traditional computer programming is in the code (programs that people write). In machine learning, algorithms (programs) are in principle simple and the complexity (structure) is in the data. Is there a way that we can automatically learn that structure? That is what is at the heart of machine learning.”
Andrew Ng

What is Machine Learning?

- Algorithms that can improve performance using training data
- Typically, a large number of parameter values learned from data
- Applicable to situations where challenging to define rules manually



Question

Can we build software that learns from ‘experience’ and enables designers and verification engineers to become more productive?

- Enhance current formal verification tools
- Enable development of new tools

Outline

- Basics of Machine Learning
- Infrastructure for Machine Learning
- Machine Learning in Formal Verification

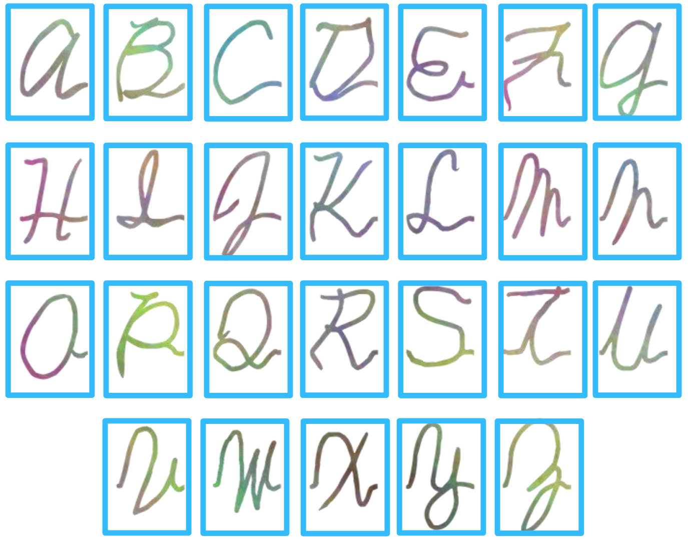
Basics of Machine Learning

Part 1

Canonical Machine Learning Problems

- **Supervised Learning**
  - Inferring a function from training data consisting of input object and desired output value (labeled data).
  - Learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.
  - **Regression**
  - **Classification**
- **Unsupervised Learning** – modeling and learning data
  - Infer a function to describe hidden structure from unlabeled data.
  - **Clustering**

Supervised Example: Classification



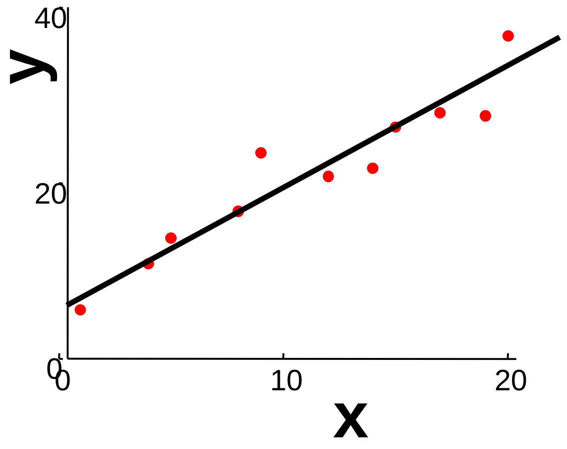
- Each character is represented by a 20x25 pixels. $x \in R^{500}$
- Character recognition machine learning task:

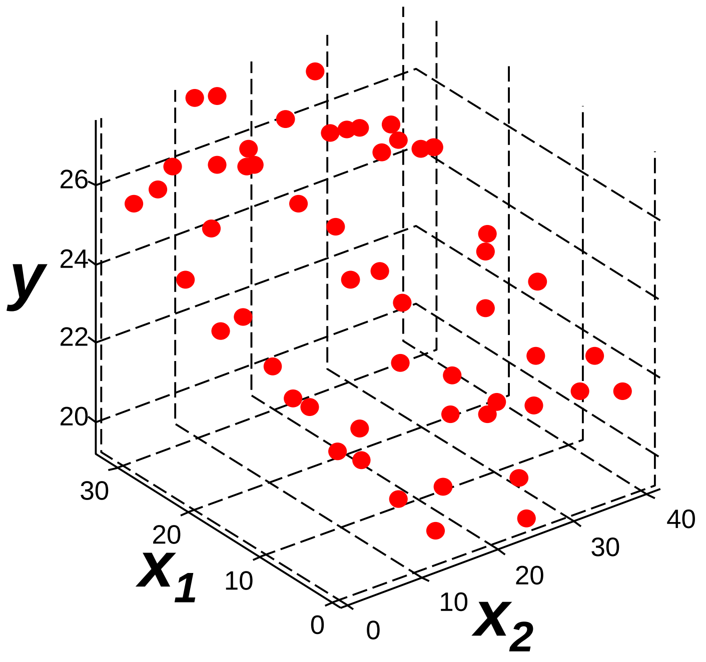
  Find a classifier f(x) such that

$$f : x \rightarrow \{a, b, c, \ldots, z\}$$

? ?

Supervised Example: Regression

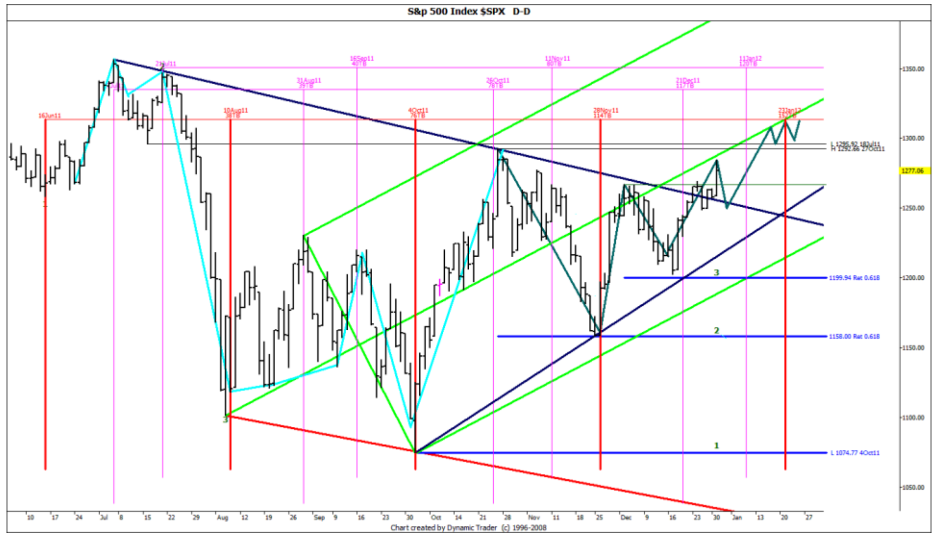




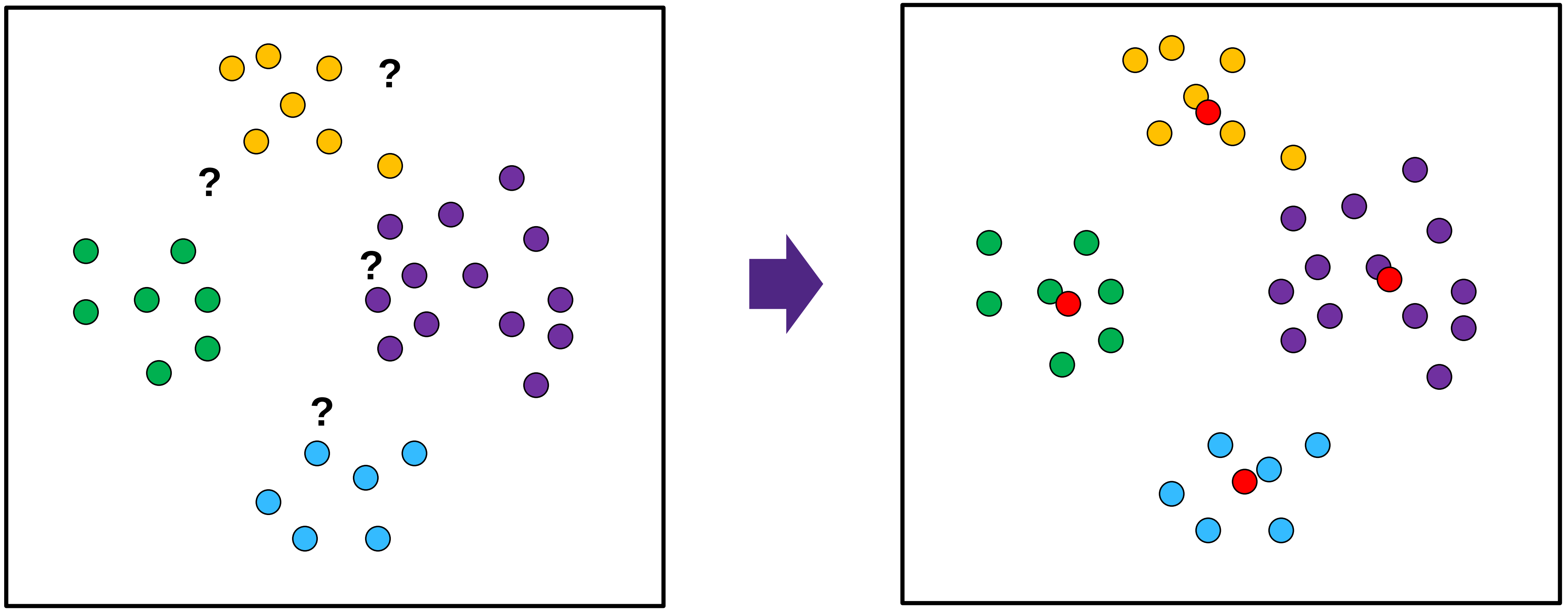
- Find a function f(x) such that

$$f : x \rightarrow y$$

$$f : \{x_1, x_2\} \rightarrow y$$



Unsupervised Example K-means Clustering

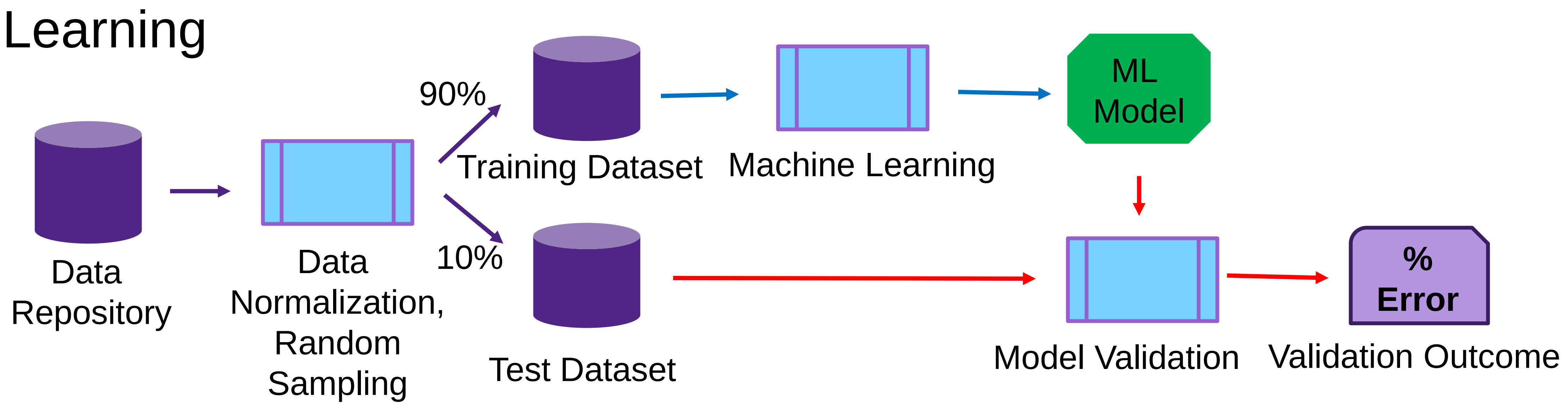


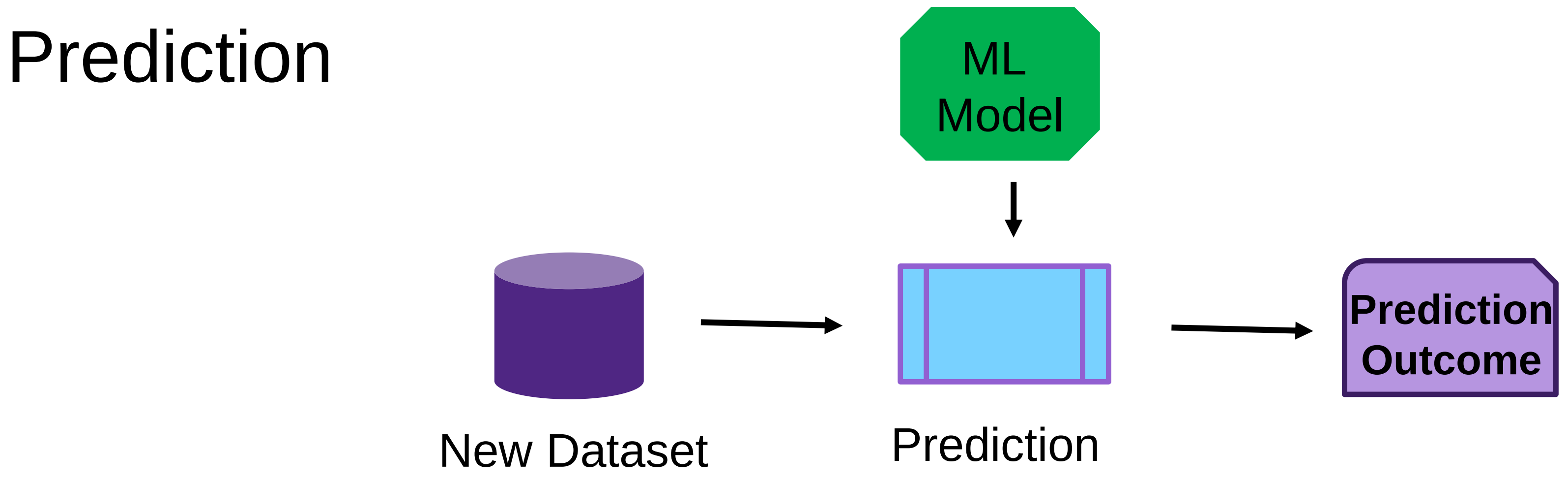
- Initialize k cluster centroids
- Associate each data instance with nearest centroid

Supervised Learning Process

- Determine the type of training examples and what kind of data is to be used as a training set.
- Gather a training set
  - A set of input objects and corresponding outputs are also gathered
  - Manually or from automated measurements.
- Determine the structure of the learned function and corresponding learning algorithm.
- Complete the design. Run the learning algorithm on the gathered training set.
- Evaluate the accuracy of the learned function.
  - Do parameter adjustment and learning
  - Measure performance of the resulting function on a test set that is separate from the training set.

Supervised Learning Process Flow





Process Flow (continued)

- Testing: Test the model using unseen test data to assess the model accuracy

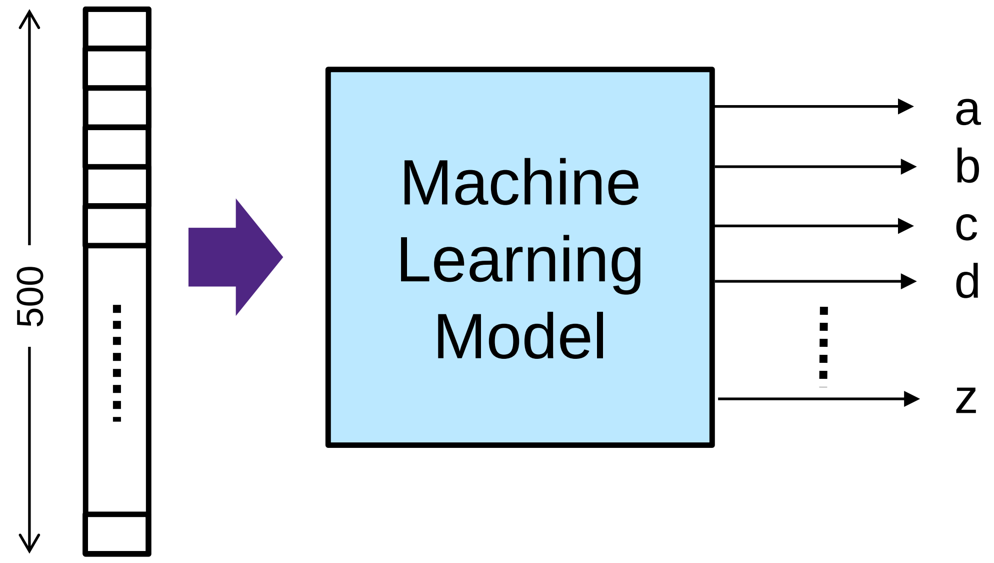
$$\text{Accuracy} = \frac{\text{No. of correct classifications}}{\text{Total no of test cases}}$$

Logistic Classification Details

- Each character is represented by a 20x25 pixels. $x \in R^{500}$
- Character recognition machine learning task:

  Find a classifier f(x) such that

$f : x \rightarrow \{a, b, c, \ldots, z\}$ f( ) = v



Logistic Classification Details Continued

$$\underset{\boxed{\text{26x500}}}{W}\,\underset{\boxed{\text{500x1}}}{x} + \underset{\boxed{\text{26x1}}}{b} = \underset{\boxed{\text{26x1}}}{y}$$

- Machine Learning Task
  - Compute weights W[26x500] and bias b[26x1]
  - For a given x,
    - Compute S(y_i) $= \dfrac{e^{y\_i}}{\sum_j e^{y\_j}}$
    - S(y_i) yields the probability that x is the ith character
- Minimize Learning Error
- How to define learning error

Logistic Classification Details Continued

Given input x, and associated label L

- Compute y = Wx + b

x = L = [0,0,0,….,0,1,0,0,0,0]

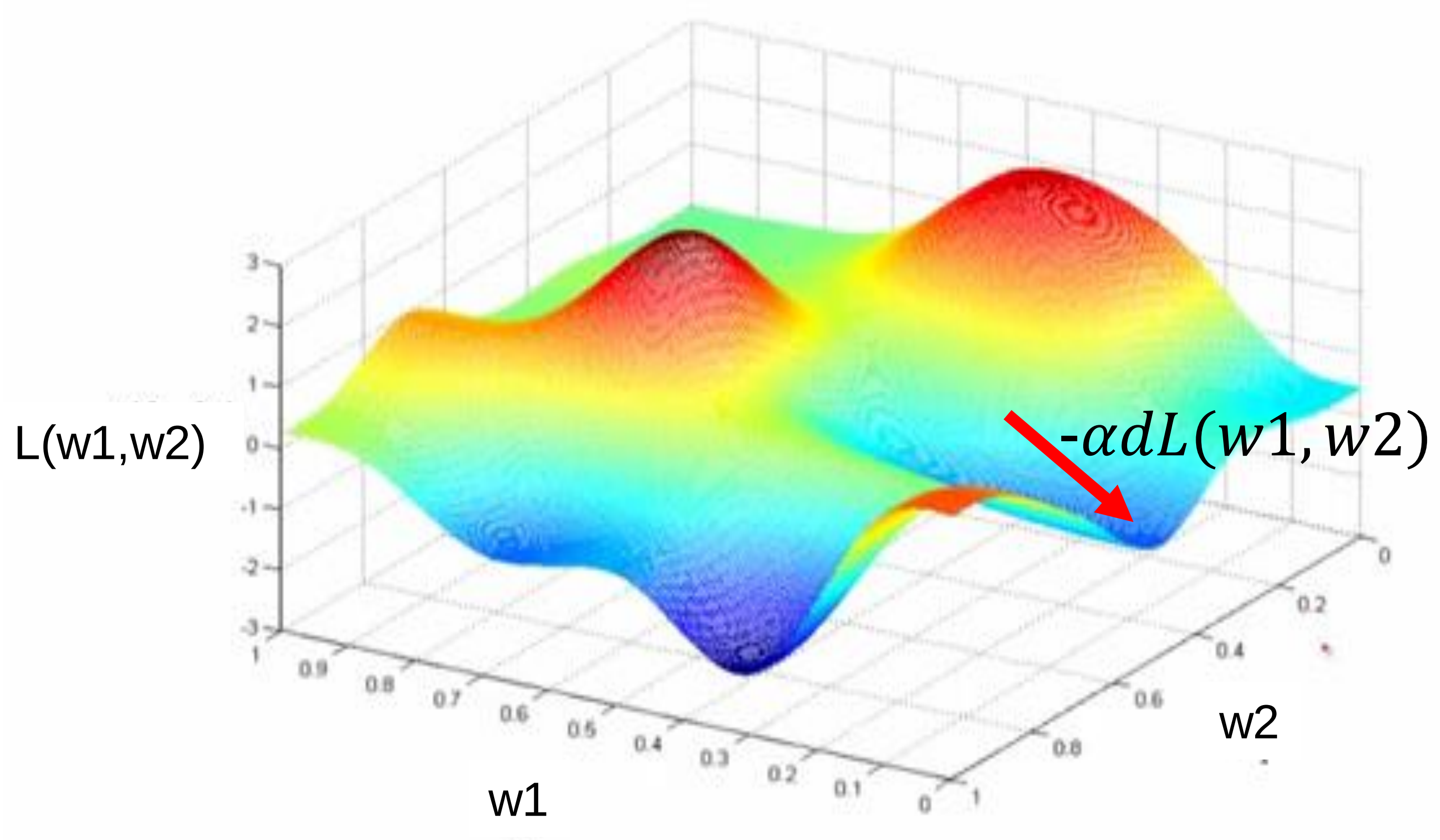
- Compute S(y)
- Cross entropy is

$$D(S, L) = -\sum_i L_i \log(S_i)$$

- Loss function

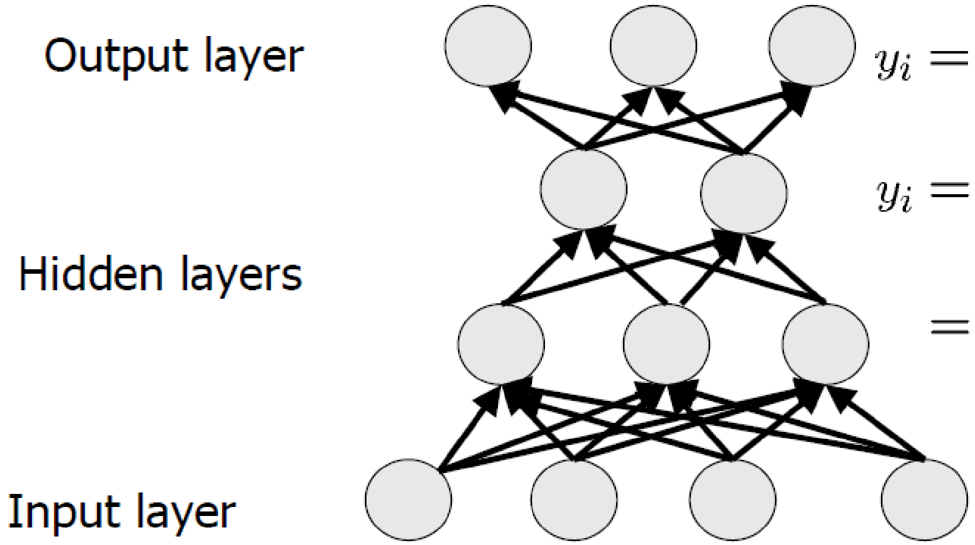
$$L = \frac{1}{N}\sum_i D(S(Wx_i + b), L_i)$$

Logistic Classification: Gradient Decent



Deep Learning Neural Networks

- Deep multi-layer networks inspired by neural structures in the brain.
- The network aims at learning feature hierarchies
  - Features from higher levels of the hierarchy are formed by lower level features

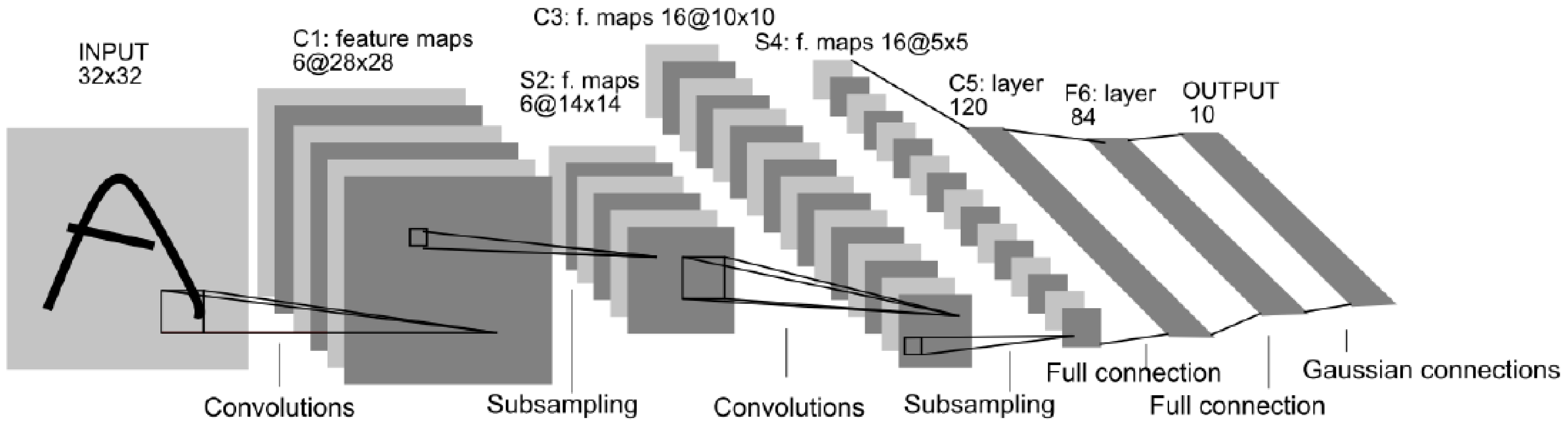


$$y_i = \tanh(\sum_j W_{ij}x_j + W_{i0})$$

$$y_i = \sigma(\sum_j W_{ij}x_j + W_{i0})$$

$$= \frac{1}{1 + \exp[-(\sum_j W_{ij}x_j + W_{i0})]}$$

Example Convolutional Neural Network (LeCun5)



Input: 32x32 pixel image.
- Cx: Convolutional layer
- Sx: Subsample layer
- Fx: Fully connected layer

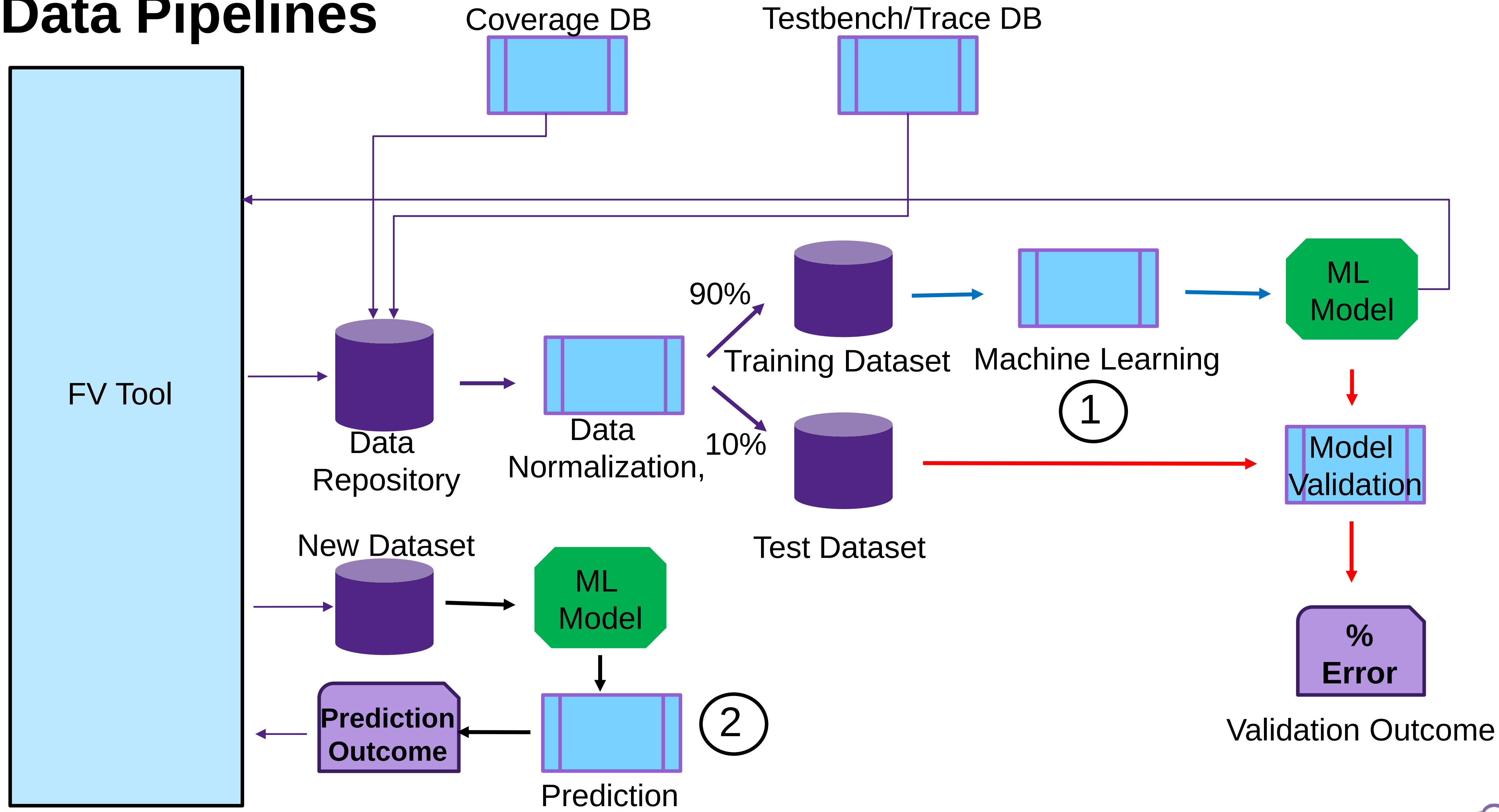
Training Parameters: weights of connections.
- C1: 28x28x(5x5+1)*6 = 122304
- S2: 14x14x(2x2+1)*6 = 5880

Weight update: Backpropagation

Infrastructure for Large Scale Machine Learning

Part 2

Data Pipelines



On-line vs Off-line

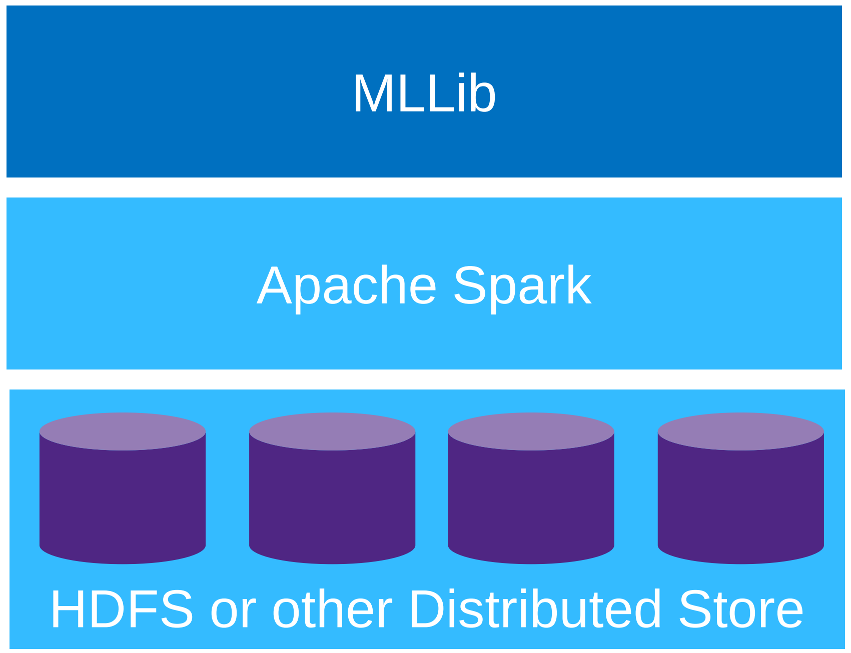
- Tool choices
  - –Learning – On-line or Off-line
  - –Prediction – On-line or Off-line
- Choices to be made at every phase of the tool operation
  - –Compilation/Model Creation
  - –Sequential Analysis/Solver
  - –Debug

Machine Learning at Scale

- Off-line and on-line machine learning
  - –Data volume
  - –Learning speed
  - –Prediction speed
- Managing data at scale is hard
  - –Distributed data storage
  - –Distributed computation
  - –Deployment and Operational considerations

Apache Spark

- Distributed in-memory computation platform
- Underlying distributed storage
- Key idea – compute pipelines with
  - Parallel computation model
  - In-memory parallelization support
  - Checkpointing
- MLlib -- Parallel Machine Learning Library implements most common ML algorithms

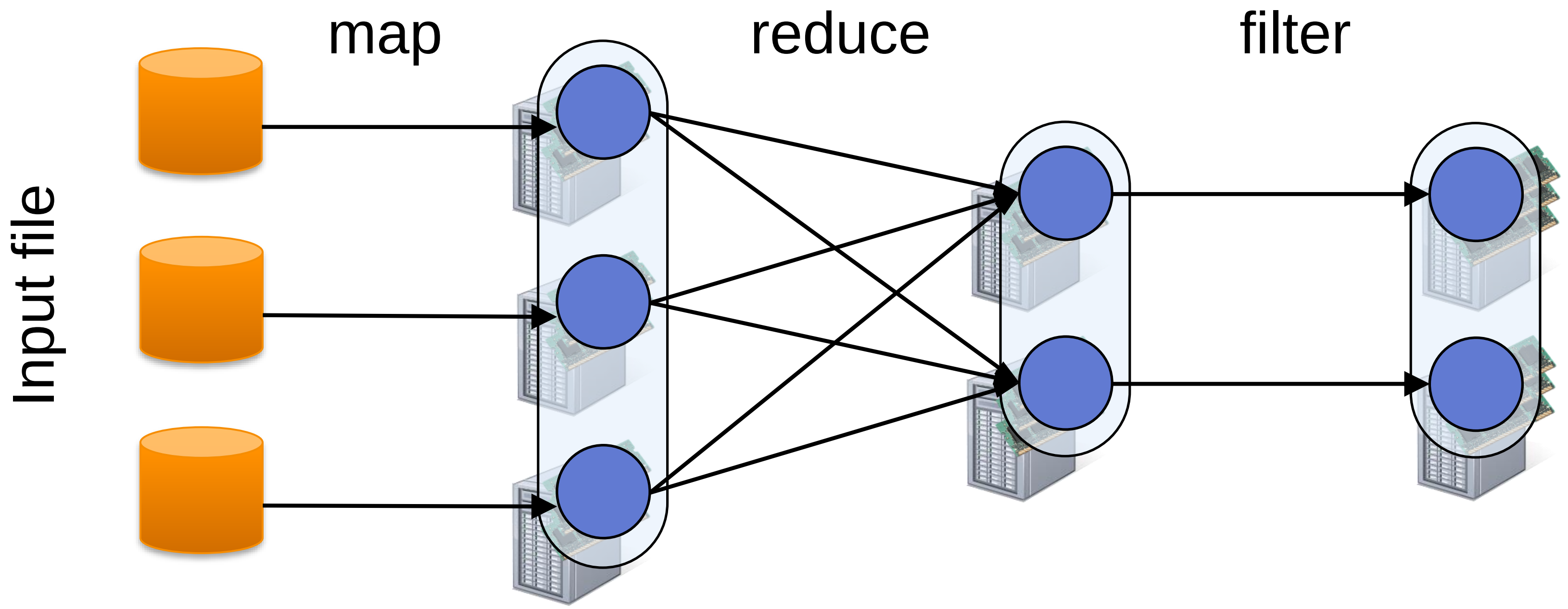


[Zaharia et.al. 2013]

Apache Spark for In-memory computation at scale

RDDs track *lineage* info to rebuild lost data

- file.map(record => (record.type, 1))
  .reduceByKey((x, y) => x + y)
  .filter((type, count) => count > 10)

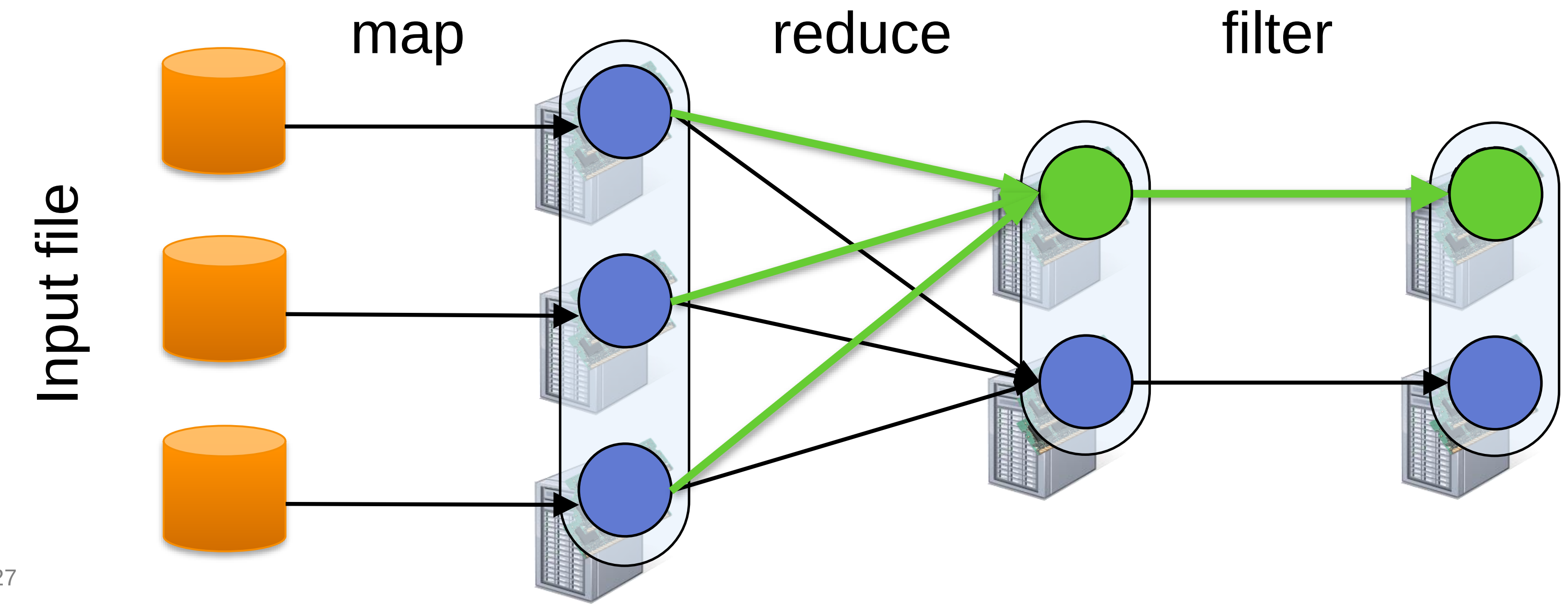


[Zaharia etal 2013]

Fault Tolerance

RDDs track *lineage* info to rebuild lost data

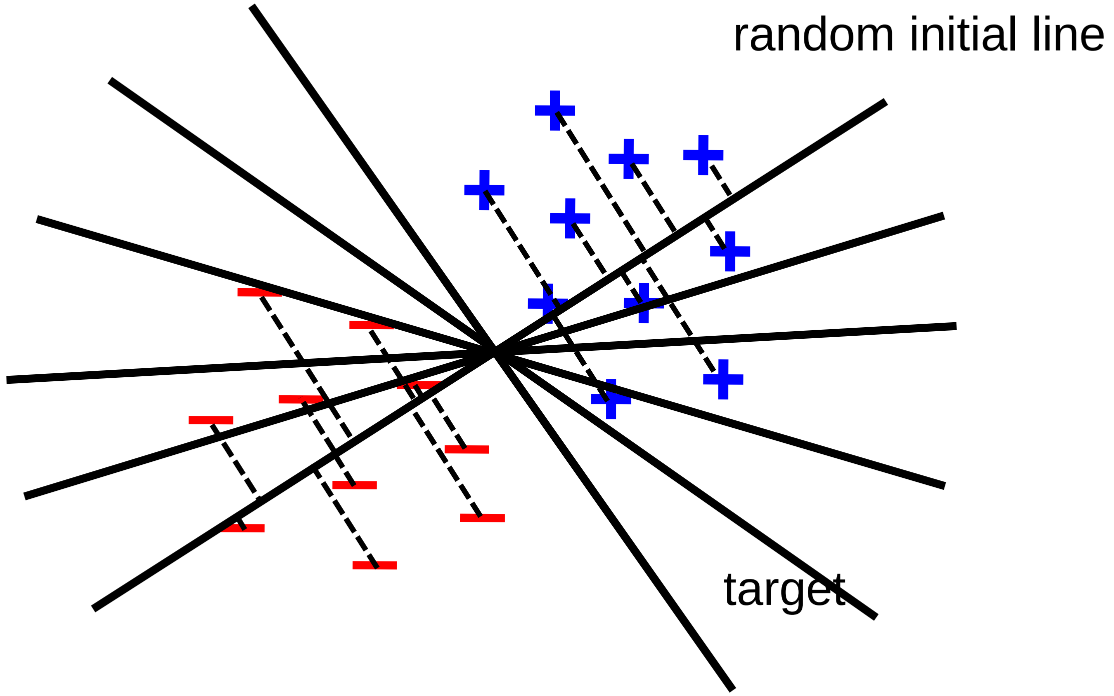
- file.map(record => (record.type, 1))
  .reduceByKey((x, y) => x + y)
  .filter((type, count) => count > 10)



[Zaharia etal 2013]

Mllib Example: Logistic Regression

Goal: find best line separating two sets of points



[Zaharia etal 2013]

Mllib Example: Logistic Regression

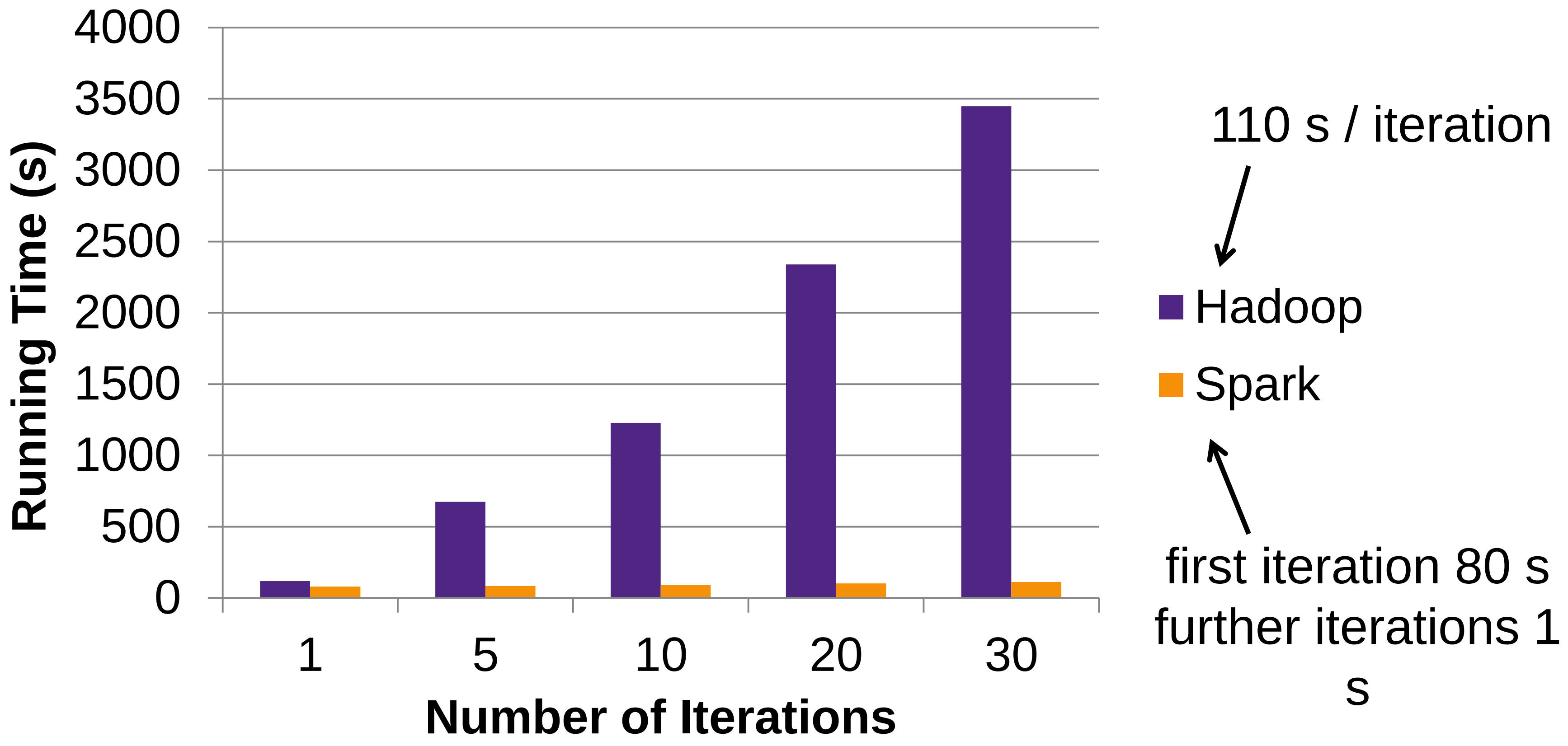
```
data = spark.textFile(...).map(readPoint).cache()

w = Vector.random(D)

for (i <- 1 to iterations) {
  gradient = data.map(p =>
    (1 / (1 + exp(-p.y * w.dot(p.x)))) * p.y * p.x
  ).reduce((x, y) => x + y)
  w -= gradient
  }

println("Final w: " + w)
```

Logistic Regression Results



Machine Learning applications for Formal Verification

Part 3

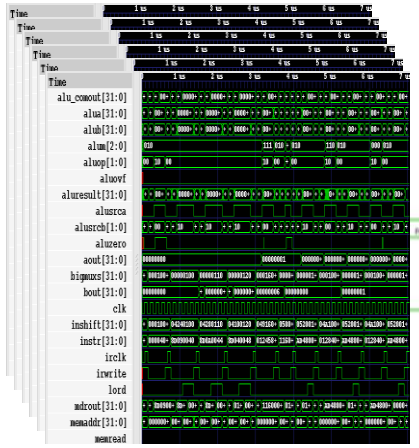
Formal Application Areas

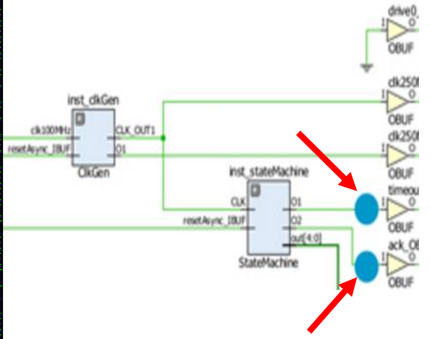
- Broad Categories
  - –Enhance current formal verification tools
  - –Enable development of new tools
- Application Areas
  - –Specification Mining
  - –Automate Troubleshooting – Expert helper
  - –Debugging and Root cause identification
  - –Aid theorem proving - Develop Solving Strategy
  - –Learning Theory -- Learn a concept from a concept class using positive and negative instances of the concept

FV Debugging Session

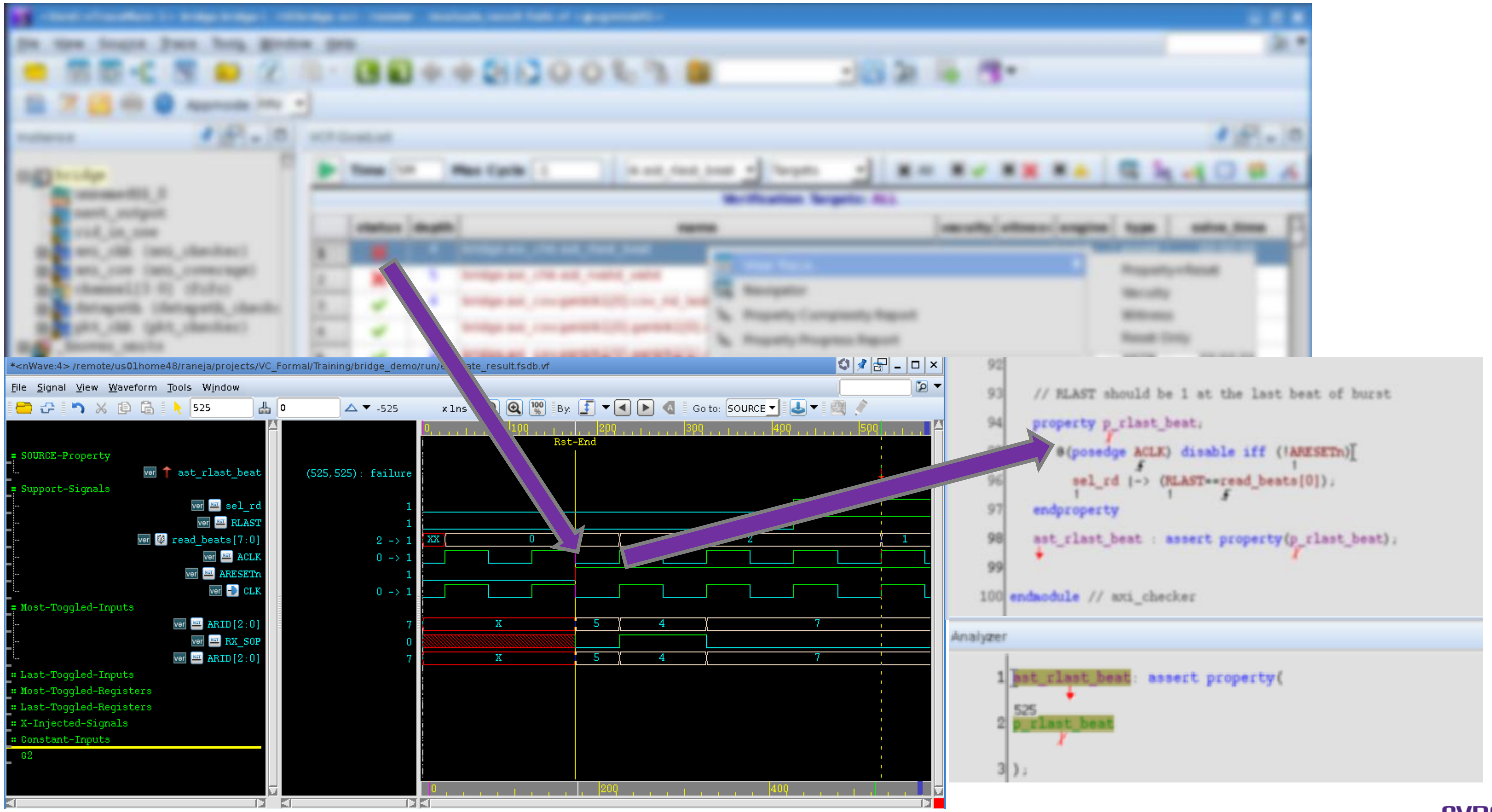
Describe your problem: I have having a failure in assertion I0/A1. The values of the Ack signal seems to be invalid in the displayed waveform.

Machine: hi
Human: hello
Machine: could you please let me know what is your run-time session?
Human: 200451
Machine: ok, I can access your FV session. What is the signal name?
Human: I0/Ack
Machine: Please specify the time at which you see the Ack signal is invalid.
Machine: What is value you expect?
Human: At time t = 105, I expect that the value to be 1.
Machine: Can you specify why this is expected?
Human: My constraints imply that a request should be successfully accepted. I0/REQ is high.
Machine: Upon analysis of your design and constraints, Ack can be 1, if reset is 0 in your constraints and I0/REQ stays high from time 25 to 55? Would you like me to modify your constraints and re-run the design?
Human: Yes, thanks!





Debugging and Root Cause Identification



*<nWave:4> /remote/us01home48/raneja/projects/VC_Formal/Training/bridge_demo/run/e[illegible]ate_result.fsdb.vf
File Signal View Waveform Tools Window
525
0
-525
x 1ns
By:
Go to: SOURCE
Rst-End
Analyzer

```
≡ SOURCE-Property
      ast_rlast_beat        (525,525): failure
≡ Support-Signals
      sel_rd                1
      RLAST                 1
      read_beats[7:0]       2 -> 1
      ACLK                  0 -> 1
      ARESETn               1
      CLK                   0 -> 1
≡ Most-Toggled-Inputs
      ARID[2:0]             7
      RX_SOP                0
      ARID[2:0]             7
≡ Last-Toggled-Inputs
≡ Most-Toggled-Registers
≡ Last-Toggled-Registers
≡ X-Injected-Signals
≡ Constant-Inputs
  G2
```

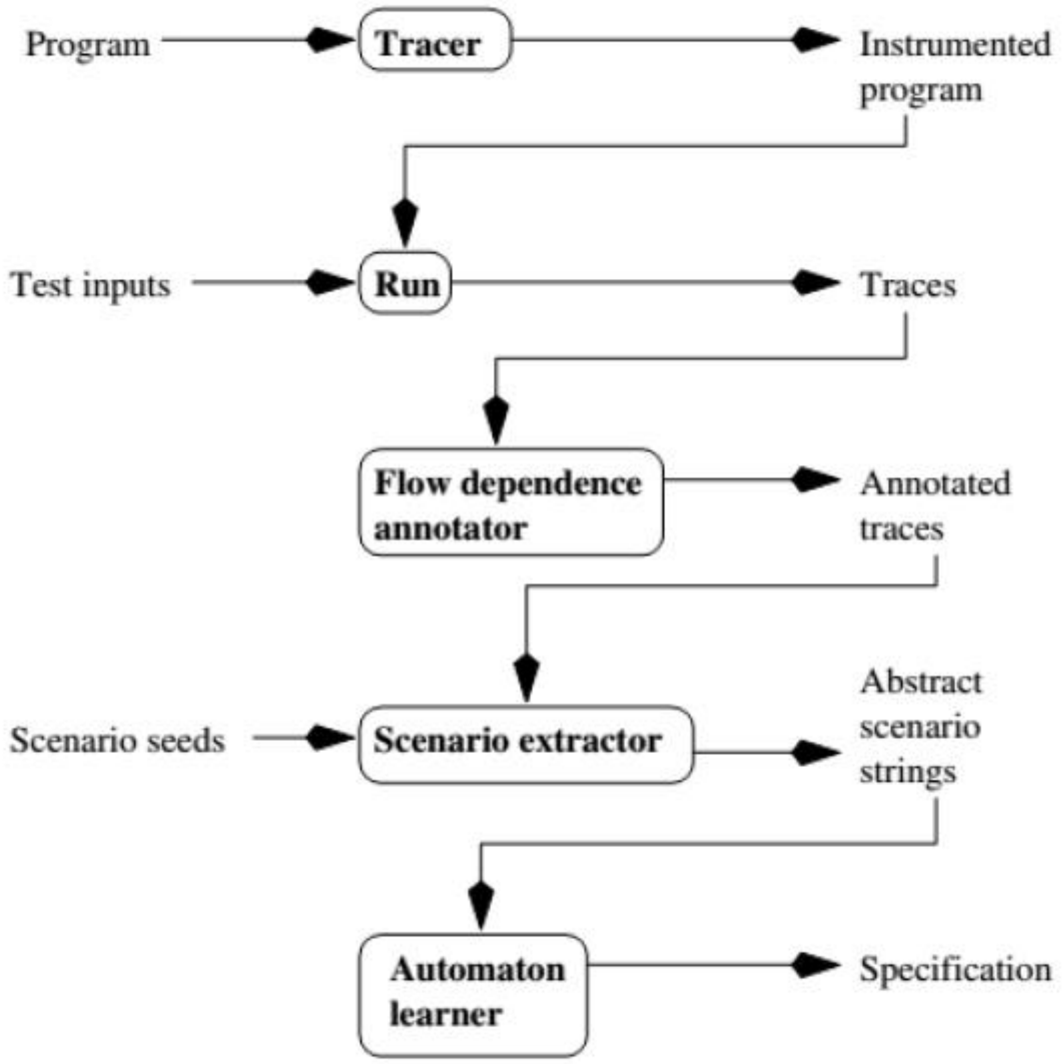
```
// RLAST should be 1 at the last beat of burst
property p_rlast_beat;
  @(posedge ACLK) disable iff (!ARESETn)
    sel_rd |-> (RLAST==read_beats[0]);
endproperty
ast_rlast_beat : assert property(p_rlast_beat);

endmodule // axi_checker
```

```
ast_rlast_beat: assert property(
525
p_rlast_beat
);
```

Specification Mining (Fellner 2015)

- Manually writing specifications is complicated and error prone
- Learn specifications from runtime traces
  - –Specification as probabilistic finite automata
  - –Learn with similarity version of k-tails Algorithm



Machine Learning aided Theorem Proving (Bridge 2014)

- ML applied to the automation of heuristic selection in a first order logic theorem prover.
  - Heuristic selection based on features of the conjecture to be proved and the associated axioms is shown to do better than any single heuristic.
- Heuristic selection amenable to machine learning.
  - The connection between input feature values and the associated preferred heuristic is too complex to be derived manually
  - For any given sample problem the preferred heuristic may be found by running all heuristics. Obtaining labelled training data is simple.
  - thus straightforward given a good selection of trial problems.The approach taken is to
- Demonstrates ML techniques should be able to find a more sophisticated functional relationship between the conjecture to be proved and the best method to use for the proof search.
  - Theorem proving more accessible to non-specialists

Computation Learning Theory (Madhusudan 2007)

- Generic theme: Learn a concept from a concept class using positive and negative instances of the concept.
  - Can we learn a Boolean function given sample evaluations?
  - Learning in presence of noise
- Probably Approximately Correct Learning (Valiant'84)
  - *For any concept δ, ϵ we can, with probability* $1-\delta$ *, efficiently learn using samples an* ϵ*-approximation of the concept.*
  - Conjunctions of Boolean literals is PAC-learnable.
- *Learn to mine* - Examples: simple loop invariants; simple predicates that control flow; simple agreements between components; simple concurrency conventions.
- Active learning [Angluin'86 , Rivest'93]
  - Learner allowed to *ask* questions:
    - Membership questions: Is $w \epsilon T$?
    - Equivalence question: Is $T = L(C)$?

Backups

Credits

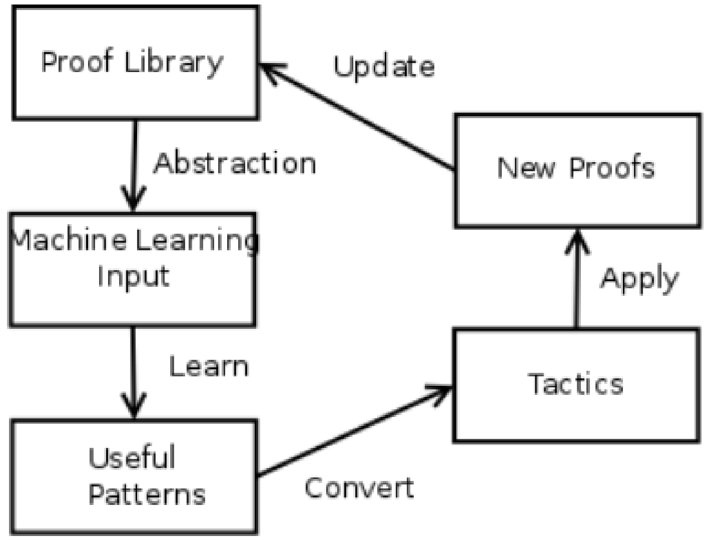
- Images/results from:
  - Ruslan Salakhutdinov
  - Joshua Bengio
  - Geoffrey Hinton
  - Yann LeCun
  - Zaharia/UCB Amplab
  - Andrew Ng
  - (Misc images on Skynet etc)

Unsupervised Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density Estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

Automating Theorem Proving - Gransden

- Theorem provers problem is that they still require a large amount of human intervention to successfully guide the proof.
- Solution involves using data mining techniques to mine proof tactics from successful and unsuccessful proofs with the aim that they can be more widely applied.


Proof Library
Update
Abstraction
New Proofs
Machine Learning Input
Apply
Learn
Tactics
Useful Patterns
Convert