Deep Learning - III

Training: Tricks of the Trade

Review

Where are we?

Program Curriculum

This program comprises of three modules:

Module 1: Introduction to Machine Learning

- ✓ 1. Motivation
- ✓ 2. The Classification Problem
- ✓ 3. Representation of World
- ✓ 4. Visualization and Unsupervised Learning
- ✓ 5. Data preparation with three problems

Duration: 5 weeks

Program Highlights

Where are we?

Module 2: Supervised Learning

- 6. Simple Linear Algorithms and Training
- 7. Linear non-separability and More Algorithms
- 8. Decision Trees
- 9. Training, Validation and Testing
- 10. Support Vector Machines

Duration: 5 weeks

Module 3: Introduction Deep Learning

- 11. Introduction to DL and Toolchain
- 12. Gradient Descent and Backpropagation
- 13. MLP as a classifier
- 14. Convolutional Neural Networks
- 15. Recurrent Neural Networks

- 15 Weeks Hybrid Program
- Weekend Contact Sessions
- 24x7 Online Labs
- Action Workshops by Industry Mentors
- Programming Experience Required

(Naïve) Summary from DL!!

- Decide Input and Output
- Choose the Architecture
- Initialize Weights
- Update Weight (eg. BP) in Iterations
- Find a Stop Criteria and Stop

Concept Map Revisited

Review: Pipeline/Concept Map (fromL4)

DATA

Structured (numerical, categorical attributes)
Digital Logs (Tweets, SMS)
RawData/Sensors (Image/Speech)
User behaviors
Etc.

FEATURE

Intuitive User defined
Raw data itself

Statistics (Histograms, PCA)

Signal Process (Fourier Xform)

FEATURE XFORMATIONS

Feature Selection

Feature Extraction

Dimensionality Reduction

Eg. PCA

ML PROBLEM

1. Classification
   a. Binary
   b. Multiclass
2. Regression
3. Clustering
4. Prediction (time series)

ALGORITHMS

1. KNN
2. Naïve Bayes
3. Perceptron
4. Linear

PERFORM. METRICS

Accuracy
Confusion Matrix
Precision
Recall
AP
True Positive
Etc.

Review: Pipeline/Concept Map

?

World

How can AI help in improving the traffic?

✔

Avatars

Sentiment Analysis

Product Rating

Abnormality Detection

Spam Filter

Etc.

ML Problem

ML Algorithms

Features and Representation

Loss, Objective Optimization Evaluation

✔

Solution

+Algorithmic Tricks
+Experimental Design
+Problem Constraints
+Practical Tricks
+Coding Tricks (?)

Nail

Hammer

Problem:
How can AIML be used for Understanding Karnataka Elections?

Can AIML really do this?

Too High Level to Start

Let us have a closer look at the problem

Goals: What and Why?

- Decide if the elections are fair and representative
- Forecast the Outcomes
- Decide if the campaigns are legal
- How much is each politician spending
- Are the media biased?
- ??
- (Real world problems; You should “smell” the ML here)

Non-Intrusive Data? (we sit in glass houses! ☹)

- Newspaper articles
- Advertisements
- Posters and Hoardings
- Debates between candidates/representatives
- Previous election results
- Social media comments
- Data from Election Commission
- Census Data

How do we start? Where is the problem?

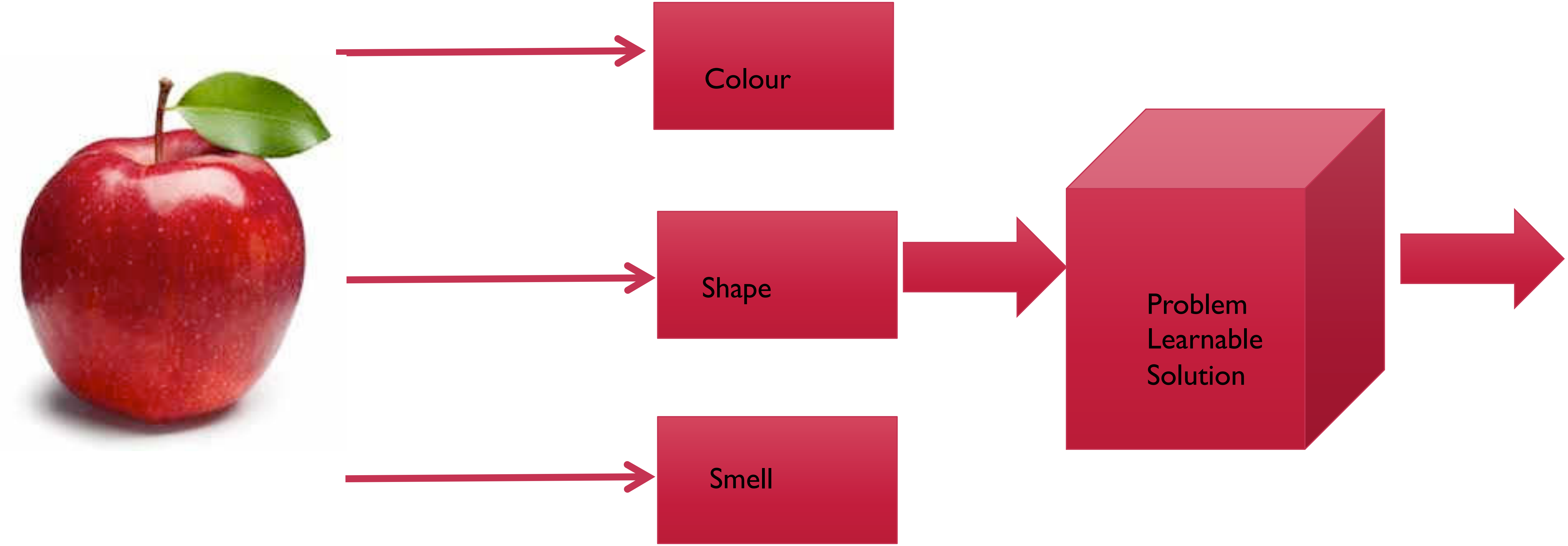
Zoom to Four Data Streams

- Broadcast Video Stream
- A Video Documentary of Roads/Public Spaces
- Talks/Scripts of Political Talks/Campaigns
- Tweets
- Do we (and how do we) use these today in "understanding elections"?

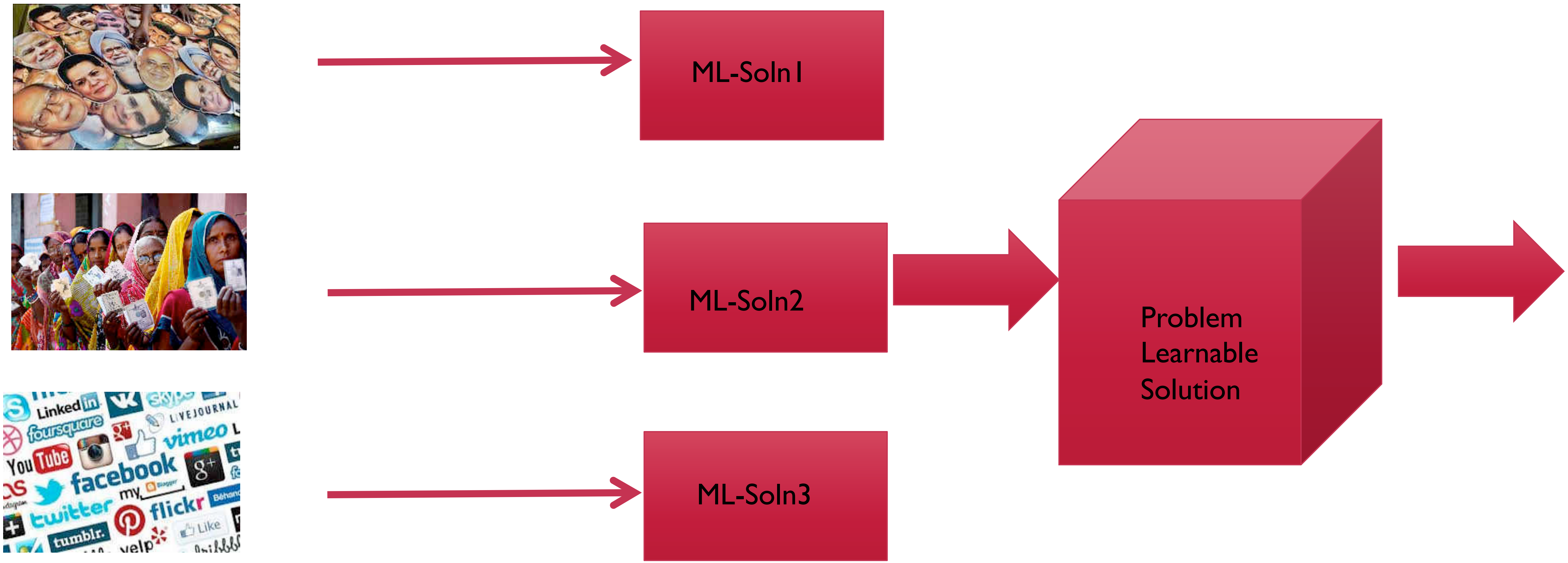
Other end of the spectrum

- Do we know how to detect and count the faces of the political leaders on posters and hoardings?
- Do we know how to detect "Symbols" on posters and hoardings?
- Do we know how to find sentiments from tweets?
- Can we count "objectionable" words in speech/Talks?
- Can we find and characterize trends over days?
- ***Common sense reasoning with "ML" as the basic block.***

Representation



Representation



An Open Problem (Your Startup Idea!!)

- Find Sentiment of a “Media Clip” in a broadcast video stream.
- What should be the labels?
- What should be the representation?
- What are the challenges?

Summary

- A large domain/space is too bad to start. Find multiple specific problems that we want to solve.
- Specify the data. Goals. Supervisory Signals.
  - Eg. Can we use "google" as a supervisor? (See todays lab)
- A number of modules/subroutines can be built directly.
- Solve larger problem by combining them.

Review: Pipeline/Concept Map

?

✔

World

How can AI help in improving the traffic?

Avatars

Sentiment Analysis

Product Rating

Abnormality Detection

Spam Filter
Etc.

ML Problem

ML Algorithms

Features and Representation

Loss, Objective Optimization Evaluation

✔

Solution

+Algorithmic Tricks
+Experimental Design
+Problem Constraints
+Practical Tricks
+Coding Tricks (?)

Nail

Hammer

Simple Summary of DL!!

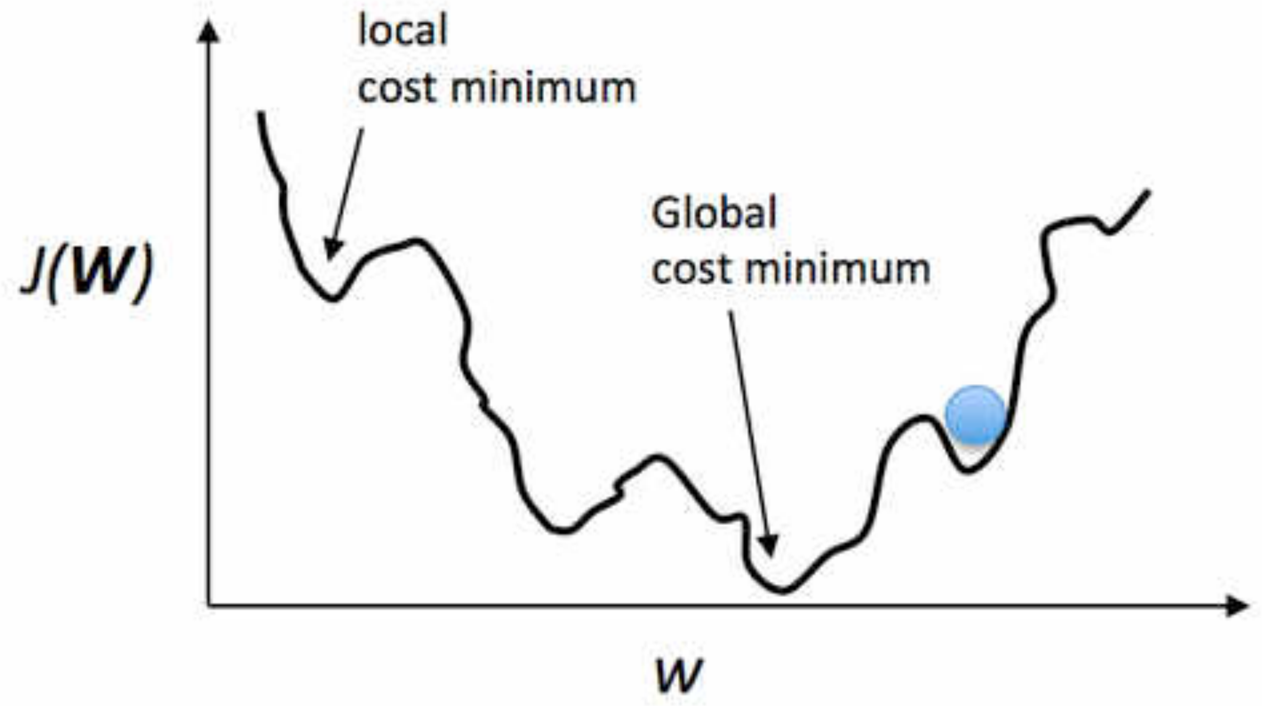
- Decide Input and Output
- Choose the Architecture
- Initialize Weights
- Update Weight (eg. BP) in Iterations
- Find a Stop Criteria and Stop

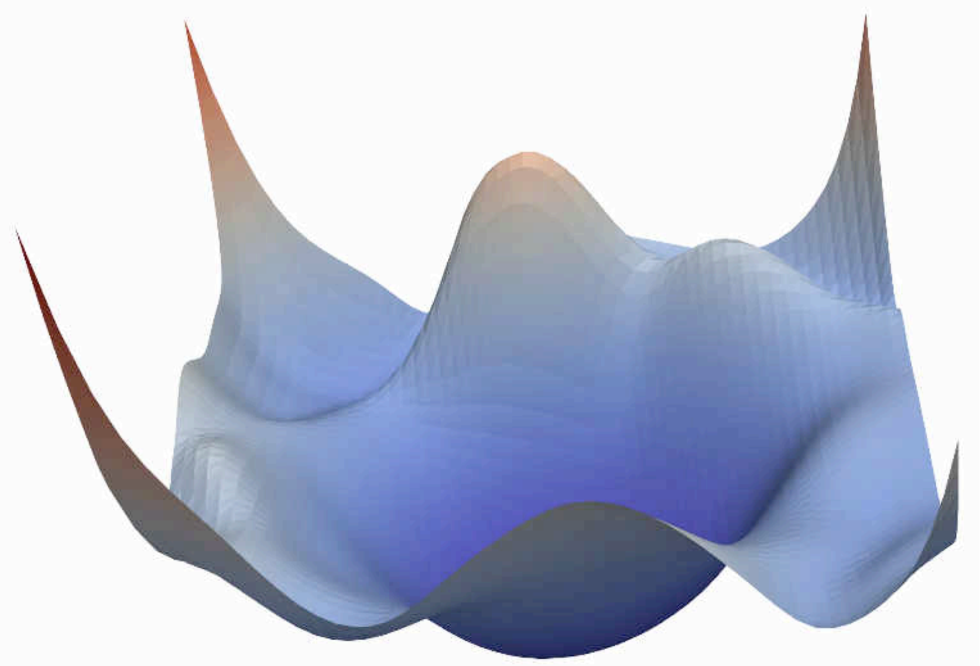
Is it this simple?

Challenges in BP: Why and How?

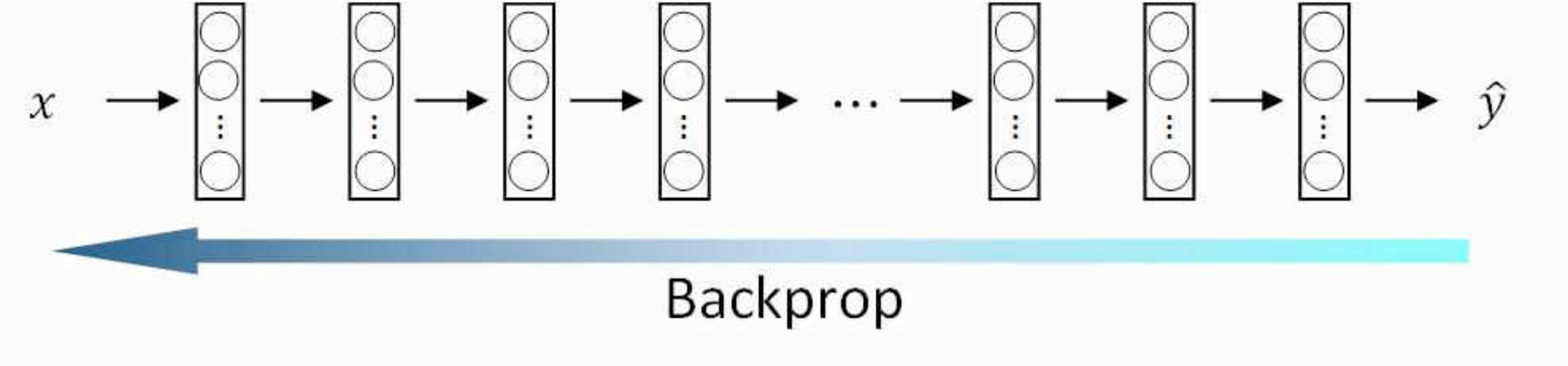
Back to classroom!!

Why is it hard?





Vanishing Gradients



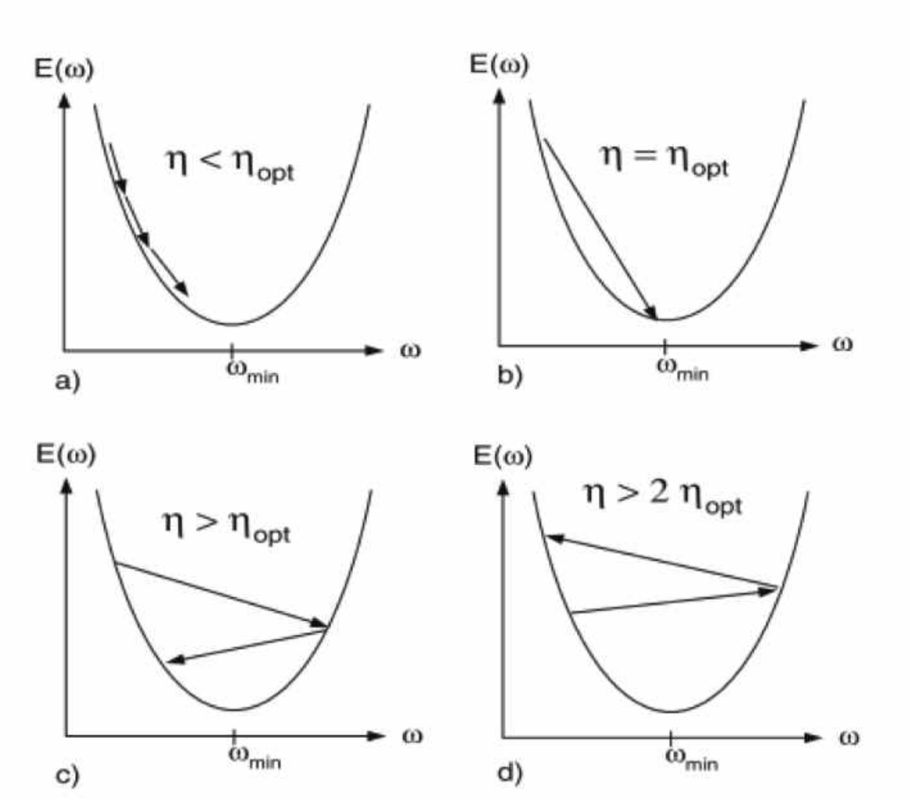
Product of a series of small numbers is very small.

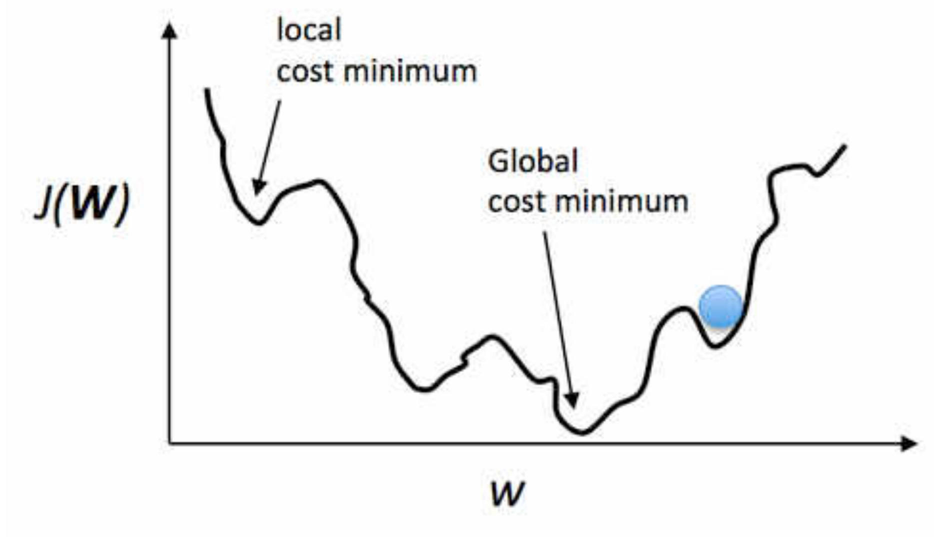
Error correction signal will not reach the initial layers

GD: Variations

- **Batch GD:** Update the parameters after the gradients are computed for the entire training set
- **Stochastic GD:** Randomly shuffle the training set, and update the parameters after gradients are computed for each training example
- **Mini-Batch Stochastic GD:** Update the parameters after gradients are computed for a randomly drawn mini-batch of training examples (this is the default option today)

Is there an optimal learning rate?





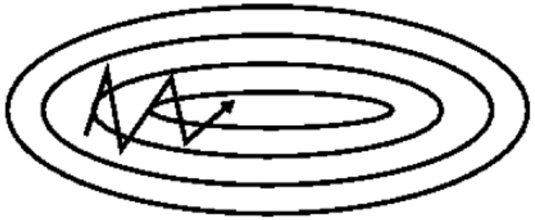
In reality, loss functions are quite complex.
(not simple quadratic to have "optimal" learning rates.

Momentum

Weight update given by:

$$\Delta\theta_{t+1} = \alpha\nabla_\theta J(\theta_t; x^{(i)}, y^{(i)}) + \overset{\text{Momentum Term}}{\boxed{\gamma\Delta\theta_t}}$$

Without momentum



With momentum

Regularization

Weight Decay

- Add a term corresponding to weights into the objective function.
- Smaller the weight (or even zero), the better.

$$C = E + \frac{\lambda}{2}\sum_i w_i^2$$

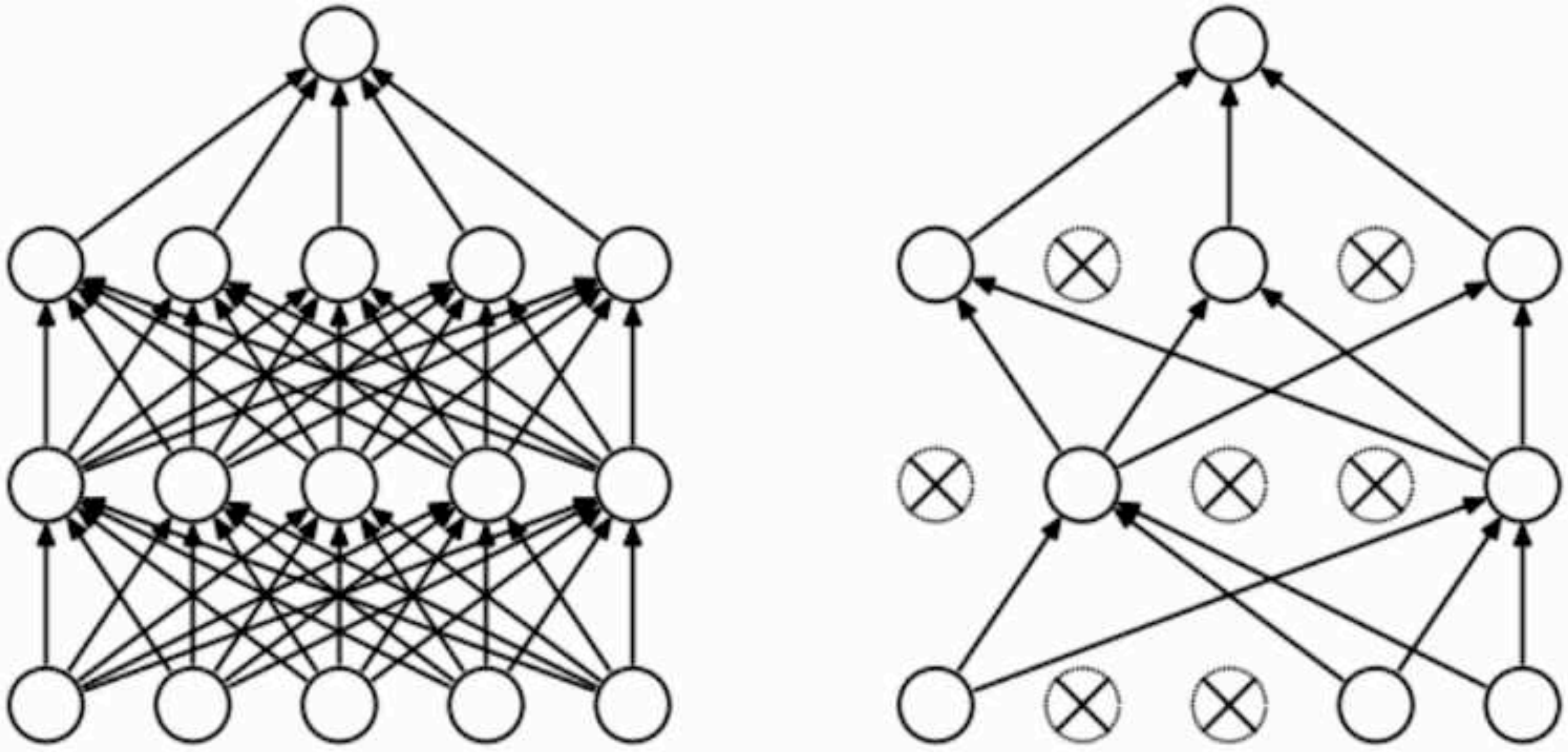
Ensemble Techniques (more later)

- We know how to build machine learning solutions.
  - Can they learn certain concepts by "accident"?
- Let us train many solutions.
  - We can fuse their predictions to obtain a better (more reliable) solution.
- Popular schemes:
  - Bagging, Boosting

Dropout

- Ignore/delete/mask certain neurons while training.
  - Get a simpler network.
  - Eg. Multiply the outputs by 0 or 1 at random.
- Equivalent to creating many neural networks.
- Reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.
- Force to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

Dropout



(a) Standard Neural Net

(b) After applying dropout.

Dropout

- Set the output of each hidden neuron to zero with a probability of p (say 0.5).
- The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in backpropagation.
- Thus neural network samples a different architecture, but all these architectures share weights.
- At test time, scale outputs by probability p.

Data Augmentation

- Data Jittering
- Rotations (transformations)
- Mirroring
- Pre-Training
- Synthetic Data

Add Noise

- Add small amount of noise.
  - Data noise
  - Gradient noise
  - Label Noise
- Bishop: ``Training with noise is equivalent to Tikhonov regularization”, 1995

More Tricks (Over to Dr. Girish)

- Good
  - Nice implementations in all frameorks
  - Fast incorporation of research/results into implementation
- Challenge
  - Experience required in getting the best.
  - Skills gets developed only by practice ☺

Thanks. Questions?