

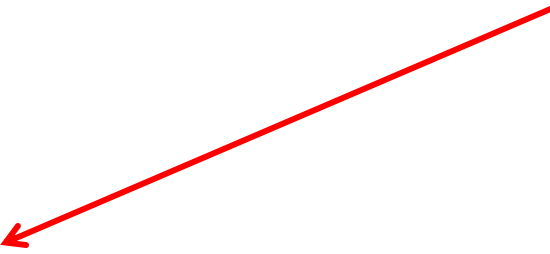
# Optimization for Training Deep Models

Sargur N. Srihari  
[srihari@cedar.buffalo.edu](mailto:srihari@cedar.buffalo.edu)

# Topics in Optimization

- Role of Optimization in Deep Learning
- How learning differs from optimization
  - Risk, empirical risk and surrogate loss
  - Batch, minibatch, data shuffling
- Challenges in neural network optimization
- Basic Algorithms
- Parameter initialization strategies
- Algorithms with adaptive learning rates
- Approximate second-order methods
- Optimization strategies and meta-algorithms <sup>2</sup>

# Optimization is essential for DL

- All Deep Learning is an instance of a recipe:
  1. Specification of a dataset
  2. A cost function
  3. A model
  4. An optimization procedure 
- The recipe for linear regression
  1. Data set :  $X$  and  $y$
  2. Cost function:  $J(\mathbf{w}) = -E_{x,y \sim \hat{p}_{data}} \log p_{\text{model}}(y | \mathbf{x}) + \lambda \|\mathbf{w}\|_2^2$  Includes regularization
  3. Model specification:  $p_{\text{model}}(y | \mathbf{x}) = N(y; \mathbf{x}^T \mathbf{w} + \mathbf{b}, 1)$
  4. Optimization algorithm
    - solving for where the cost is minimal

# Our focus is on one case of optimization

- To find parameters  $\theta$  of a neural network that significantly reduces a cost function  $J(\theta)$ 
  - It typically includes:
    - a performance measure evaluated on an entire training set as well as an additional regularization term

# Optimization in Deep Learning

- There are many contexts for optimization in DL
  1. Inference with PCA requires optimization
    - Encoding:  $f(\mathbf{x})=\mathbf{c}$ , Decoding:  $\mathbf{x} \approx g(f(\mathbf{x}))$ ,  $g(\mathbf{c})=D\mathbf{c}$
    - Optimal  $\mathbf{c}^*=\text{argmin}_{\mathbf{c}}\|\mathbf{x}-g(\mathbf{c})\|_2$ , Reconstruction:  $g(f(\mathbf{x}))=DD^T\mathbf{x}$
  2. Analytical Optimization to write proofs/design algorithms
    - Squared error objective is same as maximum likelihood
  3. Neural network training
    - Most difficult optimization of all is neural network training
      - Weight decay minimization:

$$J(\mathbf{w}) = -E_{\mathbf{x}, y \sim \hat{p}_{data}} \log p_{\text{model}}(y | \mathbf{x}) + \lambda \|\mathbf{w}\|_2^2$$

# Importance of Optimization

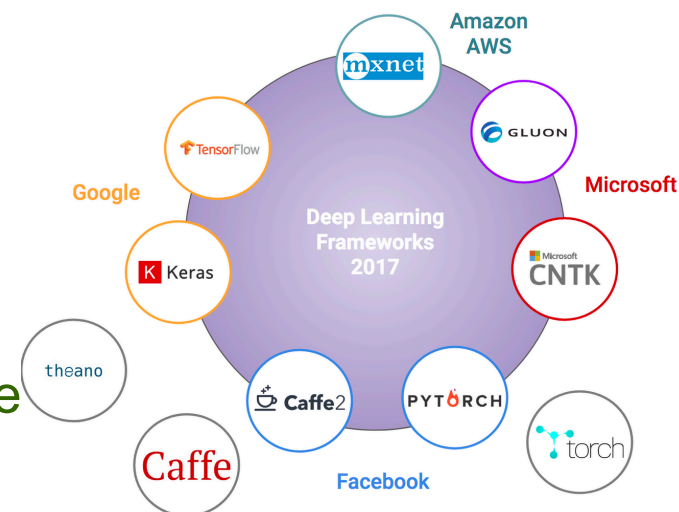
- Common to invest days to months of time on 100s of machines to solve a single instance of neural network training problem
- Because the problem is so important and so expensive
  - Specialized optimization techniques have been developed for solving it

# Plan of Discussion of Optimization

1. How training optimization differs from pure optimization
2. Challenges that make optimization of neural networks difficult
3. Several practical algorithms including
  1. Optimization algorithms
  2. Strategies for initializing parameters
    - Most advanced algorithms
      - adapt learning rates or
      - leverage *second derivatives* of cost function
4. Combine simple optimization algorithms into higher-level procedures

# DL Frameworks include Optimization

- Frameworks offer building blocks
  - For designing, training, validating deep nets through a high level programming interface
- Optimized for performance
  - Provide parallelization for GPUs
  - Visualization of network modeling & interface
- Example: Keras
  - An open source neural network library written in Python
  - It is capable of running on top of TensorFlow
  - Contains implementations of building blocks, such as
    1. Layers
    2. Objectives
    3. Activation functions
    4. Optimizers
    5. Tools to make working with image and text data easier





# Keras for MNIST Neural Network

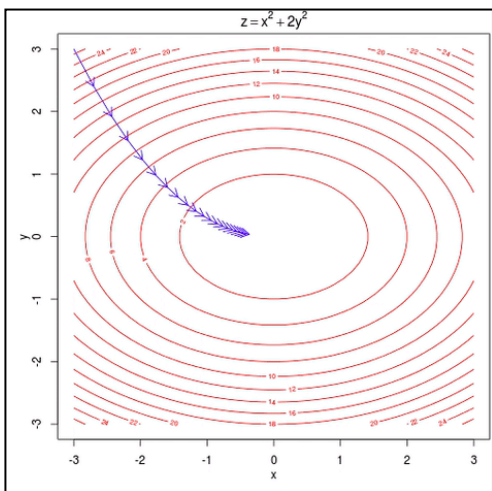
- # Neural Network
- import keras
- from keras.datasets import mnist
- from keras.layers import Dense
- from keras.models import Sequential
- (x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()
- num\_classes=10
- image\_vector\_size=28\*28
- x\_train = x\_train.reshape(x\_train.shape[0], image\_vector\_size)
- x\_test = x\_test.reshape(x\_test.shape[0], image\_vector\_size)
- y\_train = keras.utils.to\_categorical(y\_train, num\_classes)
- y\_test = keras.utils.to\_categorical(y\_test, num\_classes)
- image\_size = 784 model = Sequential()
- model.add(Dense(units=32, activation='sigmoid', input\_shape=(image\_size,)))
- model.add(Dense(units=num\_classes, activation='softmax'))
- **model.compile(optimizer='sgd', loss='categorical\_crossentropy', metrics=['accuracy'])**
- history = model.fit(x\_train, y\_train, batch\_size=128, epochs=10, verbose=False, validation\_split=.1)
- loss, accuracy = model.evaluate(x\_test, y\_test, verbose=False)

# Summary of Optimization Methods

- Movies:

<http://hduongtrong.github.io/2015/11/23/coordinate-descent/>

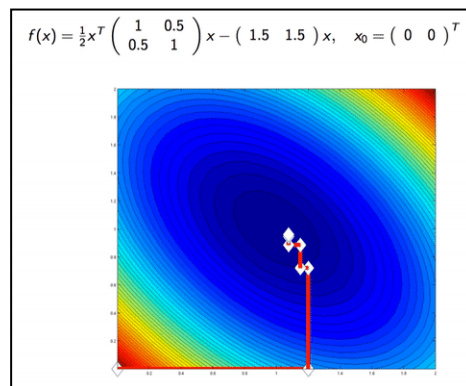
## Gradient Descent



$$g = \frac{1}{M} \nabla_{\theta} \sum_{i=1}^M L(x^{(i)}, y^{(i)}, \theta)$$

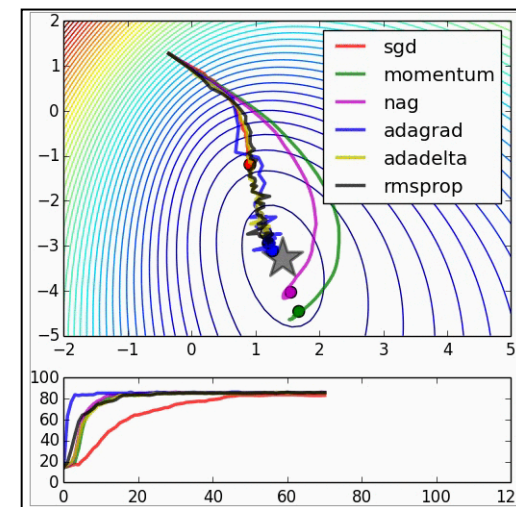
$$\theta \leftarrow \theta - \varepsilon g$$

## Coordinate Descent



Minimize  $f(x)$  wrt a single variable,  $x_i$ , then wrt  $x_j$  etc

## SGD



$$g = \frac{1}{m'} \nabla_{\theta} \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta)$$

$$\theta \leftarrow \theta - \varepsilon g$$