أكاديمية كاوست
KAUST ACADEMY

Day 3: Fundamentals Of Deep Learning Summary

Note: Do Not depend entirely on this and study from the official slides

Contributed by: Hassan Mohammed Nasr

- Deep Learning (DL) is machine learning powered by Neural Networks with "deep" layers
- Deep Learning (DL) became powerful as we got access to more data, faster computations, and better algorithms.
- The core building block of neural network is the neuron where stacking neurons give us a universal approximation function that can fit to any pattern.

Single neuron do:
- Linear combination $z = w^T x + b$
- Non Linear activation function: to add non linearity as stacking linear layers will result in linear function at the end of $a = \sigma(z)$

Some activation functions:

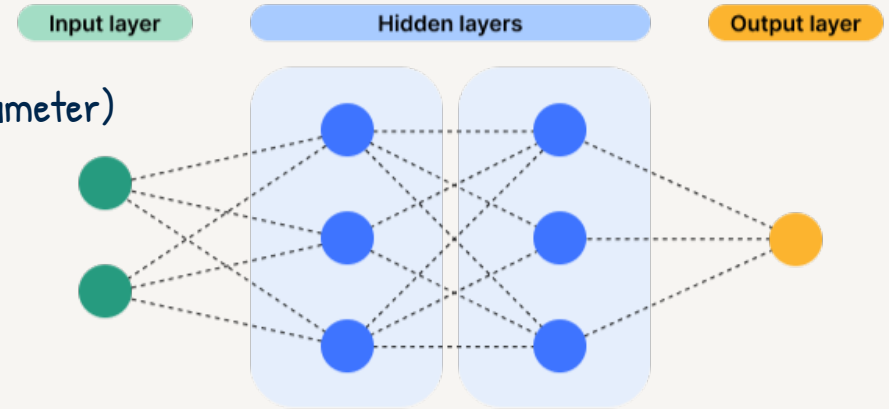
1. Relu (max(0,a)),
   Most popular.

2. Sigmoid
   Output range
   from 0 to 1

3. Tanh
   Output range
   from-1 to 1

Three parts of Neural Networks

- Input layer: Receives your data (# of neurons is the same as features size)
- Hidden layer: Extract patterns and features (# of neurons is design choice/hyperparameter)
- Output layer: Final prediction (# of neurons is the same as desired output size)



Training Neural Networks

Forward Pass

Input Data → Make Predictions → Compute Loss

Backward Pass (back propagation)

Calculate Gradients(derivatives) ← Update Parameters (Chain rule) ← Gradient Descent

$$\omega_{new} = \omega_{old} - a\nabla J$$

- Single update is called iteration
- epoch is one full pass over entire dataset

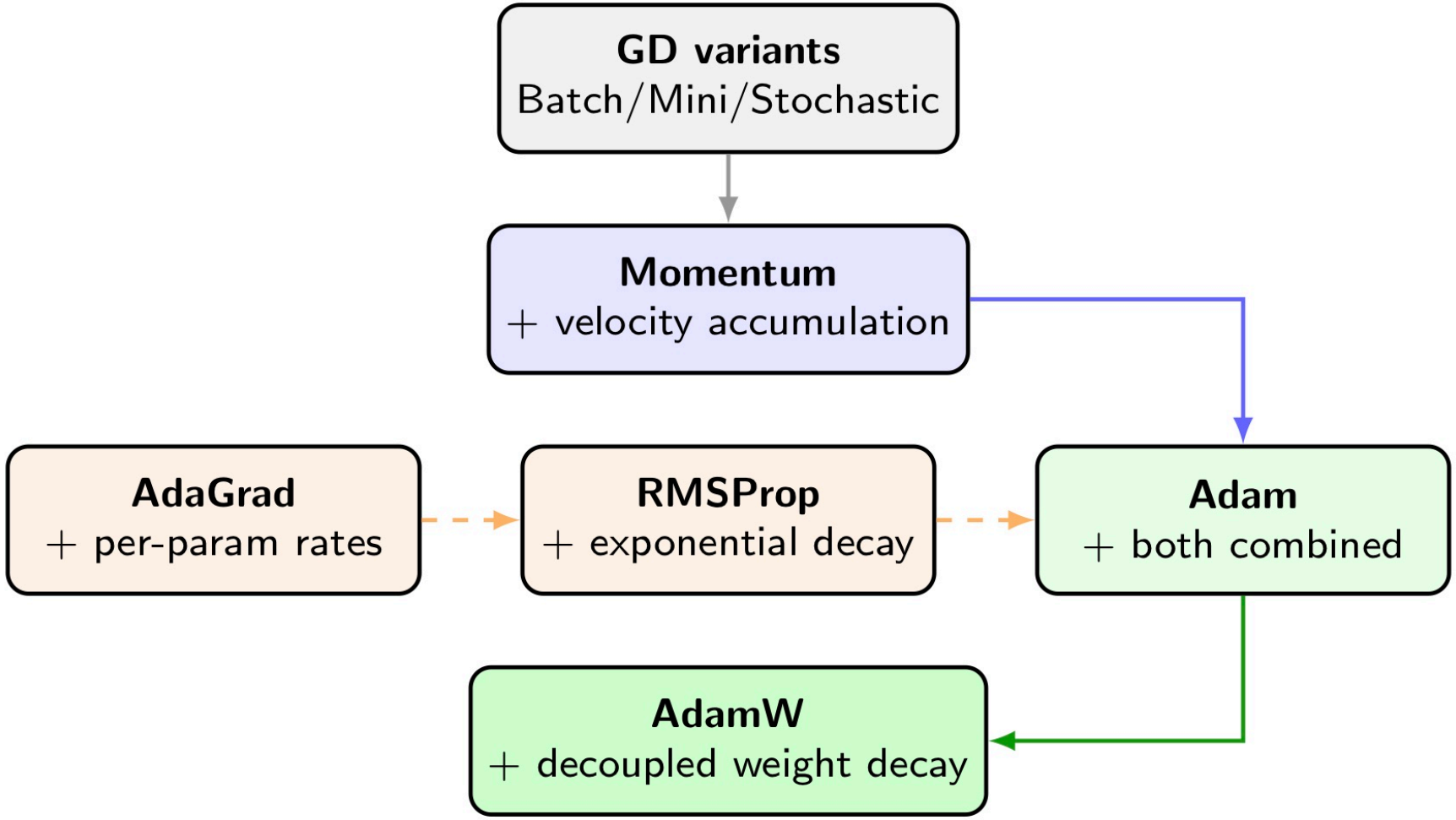
Data used for one update step

	Data	Pros	Cons
• Batch Gradient Descent	Entire dataset	Stable Update	Expensive and memory intensive
• Stochastic GD (SGD)	One sample	Escape local minima and fast	Very noisy and slow
• Mini batch GD	Small Batch	Stable Update and fast	Needs Tuning batch size

Advanced Optimizers

Optimizer	Concept	Why use
• Momentum	Adds "velocity" to the update rule	Accelerates through flat areas, cuts through oscillations
• AdaGrad	Adapts learning rate per parameter based on gradient history	Same learning rate for all parameters is not optimal, can prevent overfitting. However LR shrinks to zero too fast
• RMSProp	Fixes AdaGrad using exponential decay	Stabilizes training and prevents LR from vanishing
• Adam	Momentum + RMSProp + Bias correction	Fast and Stable
• AdamW	Adam + Decoupled Weight decay	Better generalization (current best optimizer)

Evolution of Optimizers:



Rule of thumb: Start with **AdamW** (it's the most robust default choice!)