Deep Learning: Optimizers

Bert Gollnick

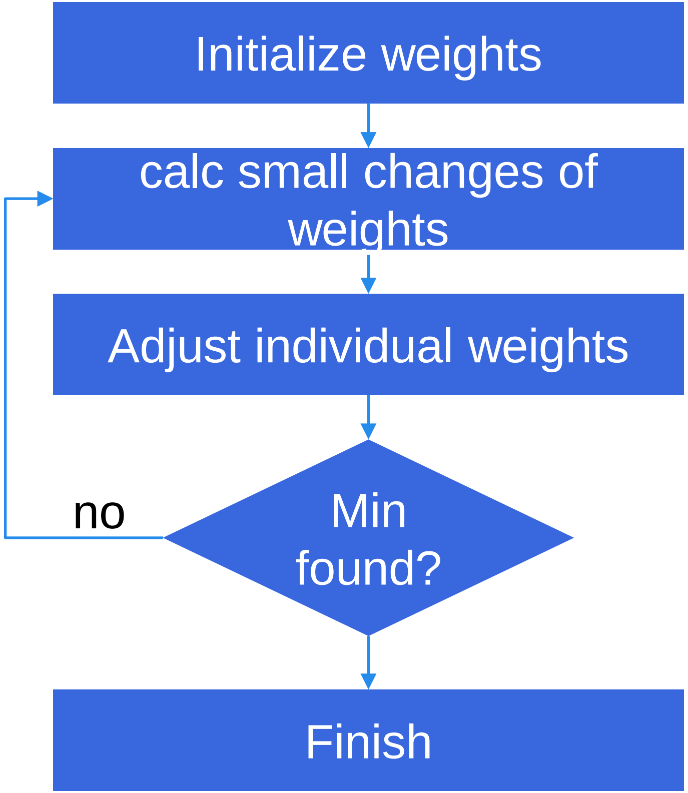
Deep Learning:Optimizer

Overview

- During training weights of the model updated to **minimize loss** function
- But how? → **Optimizer**
- Calculates **updates** of **weights** based on Loss Function
- Brute force (check all combinations) → bad idea!
- Educated trial and error → good

Deep Learning:Optimizer

Gradient Descent



Initialize weights

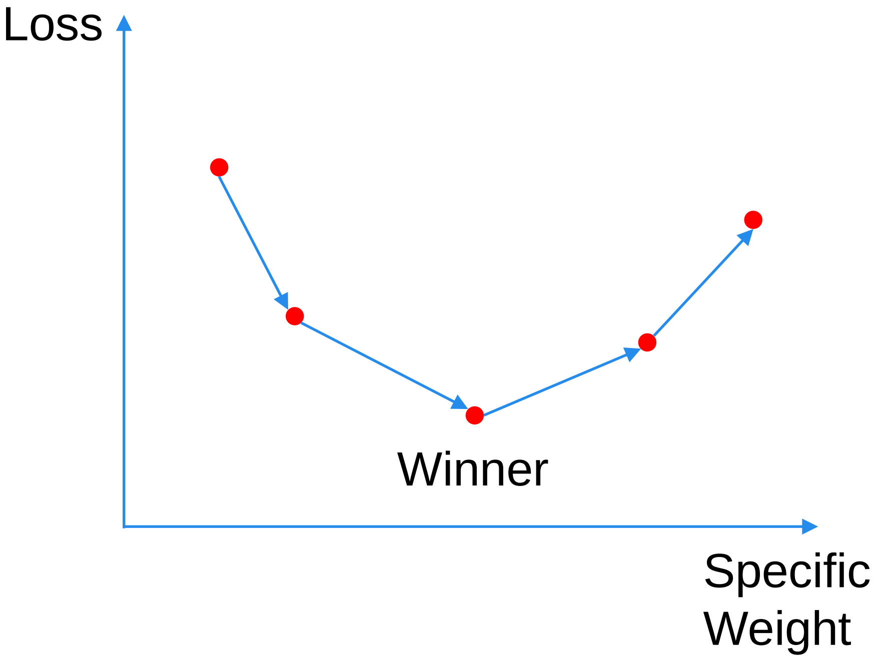
calc small changes of weights

Adjust individual weights

Min found?

no

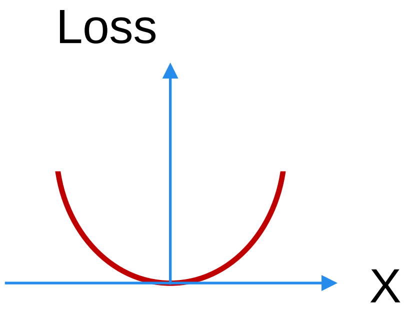
Finish



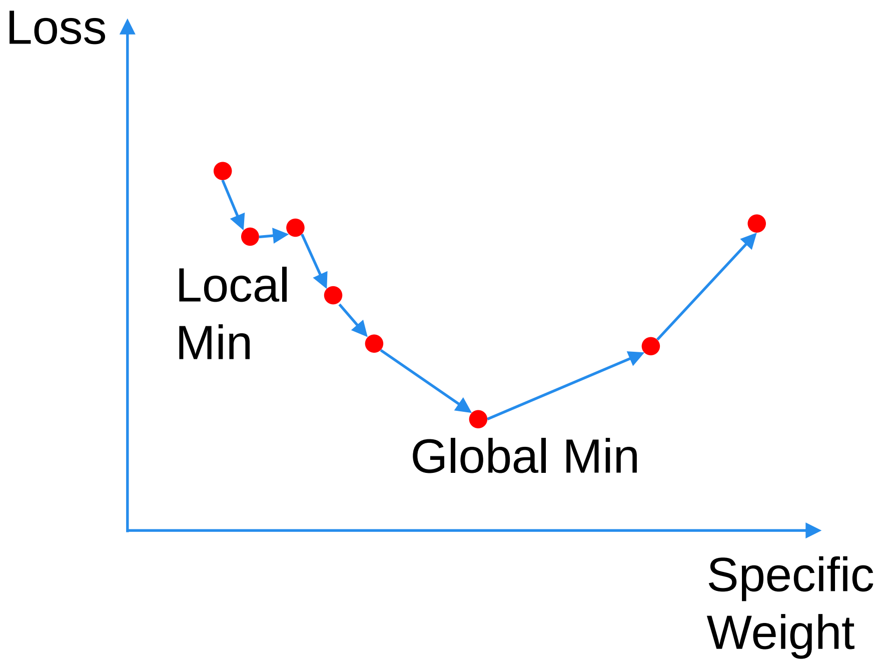
Deep Learning:Optimizer

Gradient Descent

- Problem: local minima
- Solution:
  - convex loss function



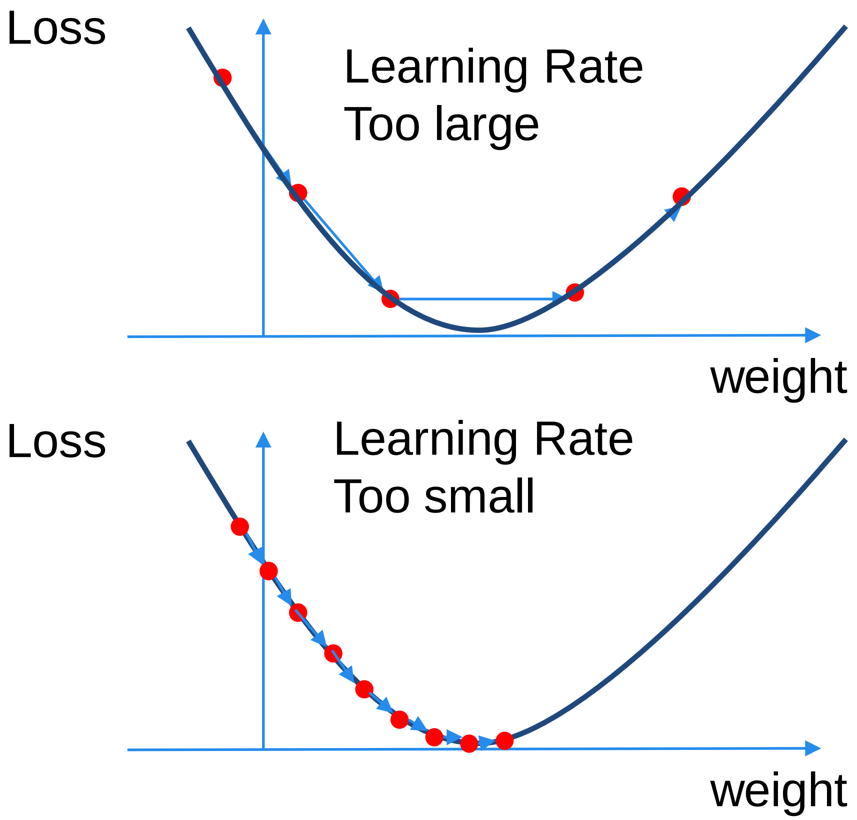
  - Learning rate



Deep Learning:Optimizer

Learning Rate

- Size of weight changes
- High learning rate
  - Large steps
  - Risk of overshooting the minimum
- Low learning rate
  - Very precise
  - Time-consuming



Deep Learning:Optimizer

Other Optimizers

Adagrad

- Adapts learning rate to features → learning rate = f(weights)
- Works well for sparse datasets
- Learning rate decreases with time and gets sometimes too small
- Adaprop, RMSprop supposed to solve this

Adam

- **Ada**ptive **m**omentum estimation
- Applies momentum → includes previous gradients into current gradient calculation
- Widespread

More Optimizers

- Stochastic Gradient Descent, Batch gradient descent, …