Optimization in Neural Networks

How do neural networks learn?

July 19, 2025

Outline

Why Optimization?

Why Optimization in Neural Networks?

Optimization is the process of minimizing a loss function by adjusting the model's parameters.

- Allows neural networks to learn patterns from data
- Uses gradients to guide parameter updates
- Goal: Find parameter values that minimize the loss function

Loss Function Examples

Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Cross-Entropy Loss:

$$\text{CE} = -\sum_{i} y_i \log(\hat{y}_i)$$

- MSE is used in regression
- Cross-entropy is used in classification

Gradient Descent

Gradient Descent: Core Idea

Gradient Descent iteratively updates parameters in the direction of the negative gradient.

$$w := w - \eta \cdot \nabla L(w)$$

- η: Learning rate
- $\nabla L(w)$: Gradient of loss with respect to parameters

Types of Gradient Descent

- **Batch Gradient Descent**: Uses entire dataset
- **Stochastic Gradient Descent (SGD)**: One sample at a time
- **Mini-batch Gradient Descent**: Uses a subset (e.g., 32 samples)

GD Variant Comparison

Type	Pros	Cons
Batch	Stable convergence	Computationally expensive
SGD	Fast, online learning	Noisy updates
Mini-batch	Balanced speed	Requires tuning batch size

Popular Optimizers

SGD with Momentum

Momentum adds velocity to help accelerate convergence:

$$v_t = \beta v_{t-1} + \eta \nabla L(w), \quad w := w - v_t$$

- β: Momentum coefficient (e.g., 0.9)
- Helps escape saddle points and oscillations

RMSProp

Adapts learning rate for each parameter:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$

$$w := w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

- Good for RNNs
- Controls exploding gradients

Adam Optimizer

Combines momentum and RMSProp:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w := w - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

Optimizer Comparison

Optimizer	Pros	Cons	Use Case
SGD	Simple, low memory	Slow convergence	General baseline
Momentum	Escapes local minima	Needs tuning	Deep nets
RMSProp	Stable updates	Sensitive to hyperparams	RNNs
Adam	Fast, adaptive	May overfit	Most models

Challenges in Optimization

Optimization Challenges

- **Saddle Points** slow down learning
- **Vanishing/Exploding Gradients** in deep nets
- **Overfitting**: Excessive optimization on training data

Learning Rate Schedules

- **Step decay**: Reduce η after fixed steps
- **Exponential decay**: Multiply η each epoch
- **Reduce on plateau**: Monitor validation loss

Best Practices

Tips & Best Practices

- Start with Adam, tune learning rate
- Use mini-batches and validation sets
- Monitor training and validation loss
- Use TensorBoard or similar tools

Key Takeaways

- Optimization drives learning in neural networks
- Gradient descent is foundational
- Adam optimizer is robust and widely used
- Always validate, tune, and monitor

Thank you!

Questions?