UNIT 3
Backpropagation

CONTENTS

- Backpropagation and Differentiation
- Hessian Matrix
- Generalization
- Cross Validation
- Network Pruning Techniques
- Virtues of Backpropagation
- Limitations of BackpropagationAccelerated Convergence
- Supervised Learning

Backpropagation and Differentiation

- **Definition**: Backpropagation is the algorithm used to train neural networks by minimizing error between predicted and actual outputs.
- **Process**:
- **Forward pass**: Inputs flow through the network, producing an output.
- **Loss calculation**: Compare predicted output with the true label using a loss function.
- **Backward pass**: Errors are propagated backward through the network.
- **Weight update**: Gradients are used to adjust weights via gradient descent.

Backpropagation and Differentiation

- **Differentiation**
- Backpropagation relies on **calculus**, specifically the **chain rule of differentiation**.
- The chain rule allows us to compute how a small change in a weight affects the final loss.
- Example: If the loss L depends on output y, and y depends on weight w, then:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w}$$

- This principle is applied layer by layer, enabling efficient gradient computation across deep networks.

Backpropagation and Differentiation

- Why It Matters
- Differentiation makes backpropagation **computationally efficient** — we don’t need to recompute everything from scratch.
- It ensures that **every parameter** in the network gets updated in the right direction to reduce error.
- Without differentiation, training deep networks would be practically impossible.
- In short: **Backpropagation = chain rule in action.**
- It’s the elegant way neural networks learn by systematically adjusting weights to minimize error.

.

Hessian Matrix

- The **Hessian matrix** is a square matrix of **second-order partial derivatives** of the loss function with respect to the model parameters (weights and biases).
- If the loss function is L(θ), where θ represents all parameters, then the Hessian is:

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial \theta_1^2} & \frac{\partial^2 L}{\partial \theta_1 \partial \theta_2} & \cdots \\ \frac{\partial^2 L}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 L}{\partial \theta_2^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Hessian Matrix

- **Role in Neural Networks**
- **Curvature Information**: The Hessian tells us about the curvature of the error surface.
  - Positive definite → local minimum.
  - Negative definite → local maximum.
  - Mixed signs → saddle point.
- **Optimization**: Second-order methods (like Newton's method) use the Hessian to adjust weights more intelligently than gradient descent.
- **Learning Rate Adaptation**: Eigenvalues of the Hessian can indicate whether the learning rate is too high or too low.

Hessian Matrix

Applications

- **Accelerated Convergence**: Newton's method or quasi-Newton methods (e.g., BFGS, L-BFGS) use Hessian approximations for faster training.
- **Regularization**: Hessian-based techniques can help detect flat minima (better generalization) vs sharp minima (overfitting).
- **Pruning**: Some pruning methods use second-order derivatives (Hessian-based pruning) to identify which weights can be removed with minimal impact.

Hessian Matrix

- **Limitations**
- **Computational Cost**: For large networks with millions of parameters, the Hessian is enormous and impractical to compute directly.
- **Approximation Needed**: In practice, we use approximations like:
  - **Diagonal Hessian** (only second derivatives of each parameter).
  - **Gauss-Newton approximation**.
  - **Fisher Information Matrix** (closely related to Hessian).

Generalization in Neural Networks

Generalization = the ability of a trained model to **perform well on unseen data** (not just the training set).

- **Why It Matters**
- A model that only memorizes training data has **low generalization**.
- Good generalization means the model captures **underlying patterns** rather than noise.

Generalization in Neural Networks

Backpropagation & Generalization

- Backpropagation optimizes weights to minimize training error.
- But without safeguards, it can lead to **overfitting** (excellent training accuracy, poor test accuracy).

Techniques to Improve Generalization

- **Regularization**: Adds penalty terms (like L1/L2) to discourage overly complex models.
- **Dropout**: Randomly “turns off” neurons during training to prevent co-adaptation.
- **Early Stopping**: Stops training when validation error starts increasing, preventing overfitting.
- **Data Augmentation**: Expands training data with transformations (images, text, etc.).
- **Cross Validation**: Ensures robustness by testing on multiple splits of data.

Generalization in Neural Networks

- **Poor Generalization = Overfitting**
- Symptoms:
  - Very high training accuracy, low test accuracy.
  - Model complexity far exceeds dataset size.
- Solutions: Simplify the model, add regularization, or gather more data.
- In short: **Generalization is the ultimate goal of supervised learning.** Backpropagation gets us to low training error, but techniques like dropout and early stopping ensure the model actually works in the real world.

Cross Validation

- A **model evaluation technique** used to assess how well a machine learning model performs on unseen data.
- It helps detect **overfitting** and ensures the model is not just memorizing training data
- **How It Works**
- The dataset is split into **k subsets (folds)**.
- The model is trained on k−1 folds and tested on the remaining fold.
- This process repeats **k times**, each time with a different fold as the test set.
- The average performance across all folds = **cross-validation score**.

Cross Validation

- Types of Cross Validation
- **k-Fold Cross Validation**: Most common; dataset divided into k equal parts.
- **Stratified k-Fold**: Ensures class proportions are preserved in each fold (important for imbalanced datasets).
- **Leave-One-Out (LOO)**: Each sample is used once as a test set; very thorough but computationally expensive.
- **Holdout Method**: Simple train/test split; less robust than k-fold.

Cross Validation

- Benefits
- Provides a **more reliable estimate** of model performance.
- Reduces bias compared to a single train/test split.
- Helps in **hyperparameter tuning** (choosing learning rate, regularization strength, etc.).
- Improves confidence in the model's ability to **generalize**.
- **Limitations**
- Computationally expensive for large datasets or complex models.
- Not always necessary if you already have a very large dataset.
- For time-series data, standard k-fold isn't suitable (need specialized methods like rolling-window CV).

Types of Pruning Techniques

Weight Pruning

- Removes individual weights that have little impact on the output.
- Often based on magnitude (e.g., prune weights close to zero).
- Results in a sparse weight matrix.

Neuron Pruning

- Entire neurons (or filters in CNNs) are removed if they contribute little to the network's performance.
- More structured than weight pruning, leading to easier hardware acceleration.

Structured Pruning

- Removes larger components like channels, layers, or blocks.
- Maintains regular structure, making the pruned network more efficient for deployment.

Pruning Techniques

Gradient-Based Pruning

- Uses sensitivity analysis: prune weights/neurons that have minimal effect on loss gradient.

Hessian-Based Pruning

- Second-order methods: use the Hessian matrix to identify parameters that can be removed with minimal impact on error.

Pruning Techniques

Benefits of Pruning

- **Efficiency**: Faster inference and reduced memory footprint.
- **Deployment**: Easier to run on edge devices (mobile, IoT).
- **Generalization**: Smaller networks may avoid overfitting.

Limitations

- Risk of **accuracy drop** if pruning is too aggressive.
- Requires careful fine-tuning after pruning.
- Computational overhead during pruning process itself.

Virtues of Backpropagation

- **Efficiency in Gradient Computation**
- Uses the **chain rule** to compute gradients quickly and systematically.
- **Scalability**
- Works for small and large networks, including deep architectures.
- **Versatility**
- Applicable to many types of neural networks (CNNs, RNNs, Transformers).
- **Automation**
- Eliminates the need for manual feature engineering by learning representations directly from data.
- **Foundation of Modern Deep Learning**
- Backpropagation is the backbone of supervised learning in ANNs.

Virtues of Backpropagation

Limitations of Backpropagation

Vanishing/Exploding Gradients

- Gradients can shrink or grow uncontrollably in deep networks, slowing or destabilizing training.

Local Minima & Saddle Points

- May get stuck in suboptimal solutions on complex error surfaces.

Sensitivity to Initialization

- Poor weight initialization can hinder convergence.

Computationally Expensive

- Training deep networks requires significant resources (time, memory, hardware).

Overfitting Risk

- Without regularization, networks may memorize training data instead of generalizing.

Requires Differentiable Functions

- Backpropagation only works when activation and loss functions are differentiable.

Accelerated Convergence

Definition

- Accelerated convergence refers to **techniques that speed up the training process** of neural networks, helping them reach optimal solutions faster and more reliably.

Why It's Needed

- Standard gradient descent can be **slow** and may get stuck in local minima or saddle points.
- Deep networks often suffer from **vanishing/exploding gradients**, making convergence unstable.
- Faster convergence = reduced training time, lower computational cost, and quicker deployment.

Accelerated Convergence

Techniques for Accelerated Convergence

- **Momentum**
- Adds a fraction of the previous update to the current update.
- Helps smooth out oscillations and accelerates movement in consistent gradient directions.
- **Adaptive Learning Rates**
- Algorithms like **AdaGrad, RMSProp, Adam** adjust learning rates dynamically for each parameter.
- Speeds up convergence by scaling updates based on gradient history.
- **Batch Normalization**
- Normalizes activations within each mini-batch.
- Reduces internal covariate shift, stabilizes training, and allows higher learning rates.

Accelerated Convergence

- **Second-Order Methods**
- Use curvature information (e.g., Hessian approximations) to make smarter updates.
- Newton’s method and quasi-Newton methods (like L-BFGS) can converge faster than first-order methods.
- **Learning Rate Scheduling**
- Gradually decreases learning rate during training (step decay, exponential decay, cosine annealing).
- Prevents overshooting minima and fine-tunes convergence.
- **Weight Initialization**
- Proper initialization (e.g., Xavier, He initialization) reduces vanishing/exploding gradients.
- Leads to faster and more stable convergence.

Accelerated Convergence

- Benefits
- **Reduced training time** → faster experimentation and deployment.
- **Improved stability** → fewer issues with divergence.
- **Better generalization** → smoother optimization paths often lead to more robust models.
- **Summary**
- Accelerated convergence is about **making backpropagation smarter and faster**.
- Momentum and adaptive learning rates push training forward efficiently.
- Batch normalization and good initialization stabilize learning.
- Second-order methods and learning rate schedules refine the optimization process.

Supervised Learning

- **Definition**: A machine learning paradigm where models are trained using **labeled data** (input-output pairs).
- The algorithm learns to map inputs → outputs by minimizing the error between predictions and true labels.
- Example: Given images (input) and their categories (output), the model learns to classify new images.

Supervised Learning

Process of Supervised Learning

- **Data Preparation**
  - Collect labeled dataset (e.g., features + target labels).
- **Model Training**
  - Use backpropagation to adjust weights based on training data.
- **Validation**
  - Evaluate performance on unseen validation data.
- **Testing**
  - Assess generalization ability on test data.

Supervised Learning

Examples of Supervised Learning

- **Image Classification**: Cat vs. dog recognition.
- **Speech Recognition**: Converting audio signals into text.
- **Sentiment Analysis**: Classifying text as positive, negative, or neutral.
- **Medical Diagnosis**: Predicting disease presence from patient data.

Supervised Learning

Benefits

- Produces highly accurate models when sufficient labeled data is available.
- Easy to evaluate performance using metrics like accuracy, precision, recall, F1-score.
- Well-suited for tasks where clear input-output relationships exist.
- **Limitations**
- Requires **large amounts of labeled data**, which can be costly to obtain.
- Risk of **overfitting** if the model memorizes training data instead of generalizing.
- Computationally intensive for large datasets and deep networks.

Thank You..