Federated Learning Project - Complete Implementation Guide

Project Overview

This project implements federated learning algorithms using different SGD variants, comparing them against the FedAvg baseline.

Team Setup

- **Team size**: 2 people (3 with extra workload)
- **Deliverables**: 5 federated learning algorithm implementations

---

1. Project Structure

```
federated-learning-project/
├── src/
│   ├── federated_framework.py      # Base framework
│   ├── optimizers.py               # All optimizer implementations
│   ├── models.py                   # CNN and NanoGPT models
│   ├── data_utils.py               # Dataset loading and splitting
│   └── experiments.py              # Experiment runners
├── experiments/
│   ├── femnist_experiments.py      # FEMNIST experiments
│   └── nanogpt_experiments.py      # NanoGPT experiments
├── results/
│   ├── plots/                      # Generated plots
│   └── logs/                       # Training logs
├── notebooks/
│   └── analysis.ipynb              # Results analysis
├── requirements.txt
└── README.md
```

---

2. Implementation Strategy

Phase 1: Core Framework (Week 1)

1. Implement base classes (`FederatedOptimizer`, `FederatedClient`, `FederatedServer`)
2. Implement FedAvg baseline
3. Test with simple dataset

Phase 2: Algorithm Implementation (Week 2-3)

Implement 5 algorithm variants (choose from):

Server-Side Optimizers (SGD on clients + optimizer on server):

- FedAdam
- FedAdagrad
- FedRMSprop
- FedMomentum
- FedNAdam
- FedYogi
- FedAdamW

Client-Side Optimizers (Optimizer on clients + averaging on server):

- FedAdam-Client (Adam on clients)
- FedAdagrad-Client
- FedRMSprop-Client
- FedMomentum-Client

Phase 3: Experiments (Week 3-4)

1. Run FEMNIST experiments with CNN
2. Run NanoGPT experiments with custom dataset
3. Test different Dirichlet alpha values (0.1, 0.5, 1.0, 10.0)

Phase 4: Analysis (Week 4)

1. Compare convergence speed
2. Compare final accuracy
3. Analyze communication efficiency
4. Study non-IID robustness

---

3. Algorithm Selection Guide

Recommended 5 Algorithms:

1. FedAvg (Baseline) ✓

- SGD on clients + weighted averaging on server
- Baseline for all comparisons

2. FedAdam-Server

- SGD on clients + Adam on server
- Better convergence in heterogeneous settings

3. FedAdam-Client

- Adam on clients + averaging on server
- Adaptive learning rates help with non-IID data

4. FedMomentum-Server

- SGD on clients + Momentum on server
- Accelerates convergence

5. FedNAdam-Server or FedYogi-Server

- Most sophisticated adaptive method
- Test state-of-the-art performance

---

4. FEMNIST Dataset Setup

Loading FEMNIST

python

```python
from datasets import load_dataset
from torch.utils.data import DataLoader, Subset

# Load FEMNIST
dataset = load_dataset("flwrlabs/femnist", split="train")

# Properties:
# - 62 classes (10 digits + 52 letters, upper and lower case)
# - 28x28 grayscale images
# - Naturally partitioned by writer (user)
```

Non-IID Distribution

The Dirichlet distribution controls data heterogeneity:

- **α = 0.1**: Highly non-IID (each client has few classes)
- **α = 0.5**: Moderately non-IID
- **α = 1.0**: Balanced non-IID
- **α = 10.0**: Nearly IID

```python
# Example: Create non-IID split
client_indices = create_non_iid_split(
    dataset,
    num_clients=10,
    alpha=0.5
)
```

---

5. NanoGPT Setup (Alternative Track)

Creating Custom Dataset

```python
# Collect your own text data
texts = [
    "essay1.txt",
    "essay2.txt",
    "journal1.txt",
    # ... different writing samples
]

# Each user's text becomes their local dataset
# This naturally creates a user-private federated setting
```

NanoGPT Model

```python
```

```python
import torch.nn as nn

class NanoGPT(nn.Module):
    def __init__(self, vocab_size, n_embd=384, n_head=6, n_layer=6):
        super().__init__()
        self.token_embedding = nn.Embedding(vocab_size, n_embd)
        self.position_embedding = nn.Embedding(1024, n_embd)
        self.blocks = nn.ModuleList([
            TransformerBlock(n_embd, n_head)
            for _ in range(n_layer)
        ])
        self.ln_f = nn.LayerNorm(n_embd)
        self.lm_head = nn.Linear(n_embd, vocab_size, bias=False)

    def forward(self, idx):
        # Implementation...
        pass
```

---

6. Experiment Configuration

Standard Hyperparameters

```python
config = {
    'num_clients': 10,
    'num_rounds': 50,
    'local_epochs': 5,
    'batch_size': 32,
    'learning_rate': 0.01,  # Adjust per optimizer
    'alpha': 0.5,           # Dirichlet parameter
    'device': 'cuda'
}
```

Grid Search Suggestions

Test variations:

- **Number of clients**: [5, 10, 20]
- **Local epochs**: [1, 5, 10]
- **Alpha values**: [0.1, 0.5, 1.0]
- **Learning rates**: [0.001, 0.01, 0.1]

---

7. Evaluation Metrics

Primary Metrics

1. **Test Accuracy**: Final and best accuracy
2. **Test Loss**: Convergence behavior
3. **Convergence Speed**: Rounds to reach target accuracy
4. **Communication Cost**: Total parameters transmitted

Analysis to Include

1. **Learning Curves**: Plot accuracy/loss vs rounds
2. **Comparative Tables**: Final performance summary
3. **Ablation Studies**: Effect of α, local epochs, etc.
4. **Statistical Significance**: T-tests between methods

---

8. Expected Results

Typical Observations

1. **FedAvg**: Good baseline, stable but may be slow
2. **Adaptive methods (Adam, NAdam)**:
   - Faster initial convergence
   - Better with non-IID data
   - May need careful learning rate tuning
3. **Momentum**:
   - Accelerates convergence
   - Reduces oscillations
4. **Client-side vs Server-side**:
   - Client-side: More flexible, adapts locally
   - Server-side: Better coordination, less client compute

---

9. Report Structure

Suggested Sections

1. **Introduction**
   - Federated learning overview
   - Motivation for SGD variants
2. **Background**
   - FedAvg baseline
   - SGD variants description
   - Related work
3. **Methodology**
   - Algorithm implementations
   - Experimental setup
   - Datasets and models
4. **Results**
   - Convergence analysis
   - Accuracy comparisons
   - Non-IID robustness
   - Ablation studies
5. **Discussion**
   - Why certain methods work better
   - Trade-offs (speed vs stability)
   - Practical recommendations
6. **Conclusion**
   - Summary of findings
   - Future work

---

10. Implementation Tips

Best Practices

1. **Reproducibility**

```python
torch.manual_seed(42)
np.random.seed(42)
```

2. **Logging**

```python
import wandb  # or tensorboard
wandb.log({
    'round': round_num,
    'test_acc': accuracy,
    'test_loss': loss
})
```

3. **Checkpointing**

```python
torch.save({
    'round': round_num,
    'model_state_dict': model.state_dict(),
    'optimizer_state': optimizer.state,
}, f'checkpoint_round_{round_num}.pt')
```

4. **Parallel Experiments**

```python
# Use multiprocessing for multiple configs
from multiprocessing import Pool
with Pool(4) as p:
    results = p.map(run_experiment, configs)
```

Common Pitfalls

❌ **Don't**: Use same learning rate for all optimizers ✅ **Do**: Tune learning rates per optimizer (Adam needs smaller LR)

❌ **Don't**: Ignore client sampling ✅ **Do**: Consider partial client participation

❌ **Don't**: Test only IID data ✅ **Do**: Vary Dirichlet α to test robustness

---

11. Additional Resources

Papers to Read

1. **FedAvg**: "Communication-Efficient Learning of Deep Networks from Decentralized Data"
2. **FedAdam**: "Adaptive Federated Optimization"
3. **Dirichlet Distribution**: "Measuring the Effects of Non-Identical Data Distribution"

GitHub References

- FedML: https://github.com/FedML-AI/FedML
- Flower: https://github.com/adap/flower
- PyTorch Examples: https://github.com/pytorch/examples

Datasets

- FEMNIST: https://huggingface.co/datasets/flwrlabs/femnist
- LEAF Benchmark: https://leaf.cmu.edu/

---

12. Timeline (4 Weeks)

Week 1

- [ ] Set up environment
- [ ] Implement core framework
- [ ] Test FedAvg baseline

Week 2

- [ ] Implement 3 additional algorithms
- [ ] Verify correctness
- [ ] Start FEMNIST experiments

Week 3

- [ ] Implement remaining algorithms
- [ ] Run comprehensive experiments
- [ ] Test different configurations

Week 4

- [ ] Analyze results
- [ ] Create visualizations
- [ ] Write report

---

13. Deliverables Checklist

- [ ] 5 federated learning algorithm implementations
- [ ] Working code with documentation
- [ ] Experimental results on FEMNIST or NanoGPT
- [ ] Comparison plots and tables
- [ ] Written report (8-12 pages)
- [ ] README with instructions
- [ ] (Optional) Presentation slides

Quick Start Commands

```bash
# Install dependencies
pip install torch torchvision datasets numpy matplotlib pandas tqdm

# Run baseline experiment
python experiments/femnist_experiments.py --algorithm fedavg

# Run comparison
python experiments/femnist_experiments.py --compare-all

# Analyze results
jupyter notebook notebooks/analysis.ipynb
```

Good luck with your project! 🚀