Comprehensive Review: Neural Network Examples and Training Insights

DSC 255 - Machine Learning Fundamentals

Contents

1 Overview

This review illustrates how neural networks can be used in practice through concrete examples. The examples focus on classification tasks using 2D datasets and showcase how architectural choices (like the number of hidden units) affect the learned decision boundaries.

2 General Network Architecture for 2D Classification

- **Input Layer**: 2 nodes, one per feature (x_1, x_2)
- **Hidden Layer**: H nodes, where H is varied across experiments
- **Output Layer**: 1 node with sigmoid activation for binary classification
- **Activation Functions**:
  - Input to hidden: Linear function + ReLU
  - Hidden to output: Linear function + Sigmoid

3 Example 1: Simple Nonlinear Boundary

- Goal: Separate two distinct clusters using a nonlinear boundary
- Initial trial with $H = 2$ hidden nodes succeeded
- Random initialization and order of training examples can lead to different outcomes even with the same architecture
- Due to non-convexity, optimization may land in poor local minima

4 Example 2: Need for Redundancy

- Another 2D dataset with complex class structure
- Trials with $H = 4$ often underperformed: the network used only 2 out of the 4 hidden units effectively
- Performance improved significantly when $H = 8$ (overparameterized setting)
- Takeaway: Redundancy in hidden nodes often aids optimization

5 Example 3: Complex Nonlinear Boundary

- Highly intricate class boundaries required significantly more capacity
- Trials:
  - $H = 4$: Only a linear boundary was learned
  - $H = 8$: Slight improvement, but still linear-like
  - $H = 16$: Mixed results, moderate success

  - $H = 32$: Better nonlinearity and classification accuracy
  - $H = 64$: Final trial achieved zero classification error on training set
- Key insight: For complex boundaries, increase H until expressive capacity is sufficient

6 Randomness in Training

- Neural net training is highly stochastic:
  - Random initialization of weights
  - Random shuffling of training data
- These factors affect convergence and can yield drastically different models from the same architecture

7 Overparameterization Can Help

- Using more hidden units than necessary increases redundancy
- This often helps gradient descent avoid poor local optima
- Overparameterized nets generalize well in practice with regularization

8 PyTorch Implementation Snippets

Network Declaration

```
d, H = 2, 8
model = torch.nn.Sequential(
    torch.nn.Linear(d, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, 1),
    torch.nn.Sigmoid()
)
lossfn = torch.nn.BCELoss()
```

Training Step

```
ypred = model(x)
loss = lossfn(ypred, y)
model.zero_grad()
loss.backward()
with torch.no_grad():
    for param in model.parameters():
        param -= eta * param.grad
```

9 Conclusion

These examples illustrate:

- How architectural choices affect neural net behavior
- The benefits of overparameterization
- The impact of randomness on convergence
- The practicality of using PyTorch to experiment with model design and training

Neural nets are flexible but sensitive to training configuration. Trial-and-error experimentation is often essential to success.