STAT 4830: Numerical optimization for data science and ML

Lecture 0: Introduction

Professor Damek Davis

Course Overview

- **Focus**: Numerical optimization for data science and ML
- **Tools**: PyTorch, Python, occasional use of other libraries
- **Format**: Lecture-based
- **Final Project**: Incrementally developed throughout semester

Prerequisites

- Basic calculus and linear algebra (Math 2400)
- Basic probability (Stat 4300)
- Python programming experience
- No advanced optimization/ML background needed

Why PyTorch?

- Modern auto-differentiation frameworks drive deep learning success
- Enables rapid experimentation with:
  - New model architectures
  - Novel optimization algorithms
- More flexible than traditional solver-based tools

Preview: Spam Classification

Let's start with a practical example:

- How do we automatically filter spam emails?
- Demonstrates core optimization concepts
- Shows PyTorch in action

How Computers Read Email

```
email1 = """
Subject: URGENT! You've won $1,000,000!!!
Dear Friend! Act NOW to claim your PRIZE money!!!
"""

email2 = """
Subject: Team meeting tomorrow
Hi everyone, Just a reminder about our 2pm sync.
"""
```

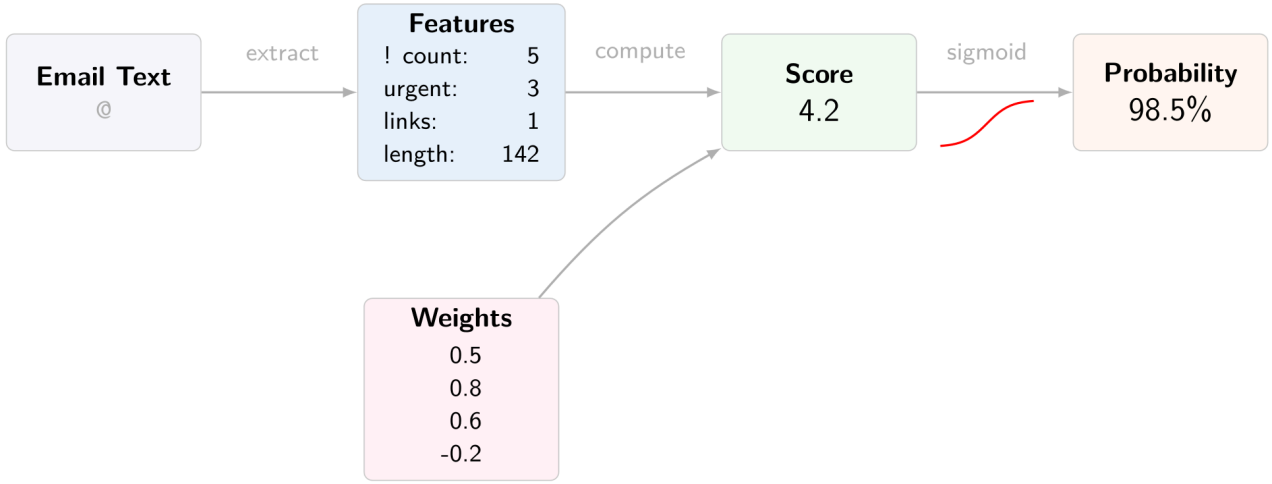
Feature Extraction

Convert text to numbers:

```
def extract_features(email):
    features = {
        'exclamation_count': email.count('!'),
        'urgent_words': len(['urgent', 'act now', 'prize']
                            & set(email.lower().split())),
        'suspicious_links': len([link for link in email.split()
                                 if 'www' in link]),
        'time_sent': email.timestamp.hour,
        'length': len(email)
    }
    return features
```

Classification Process

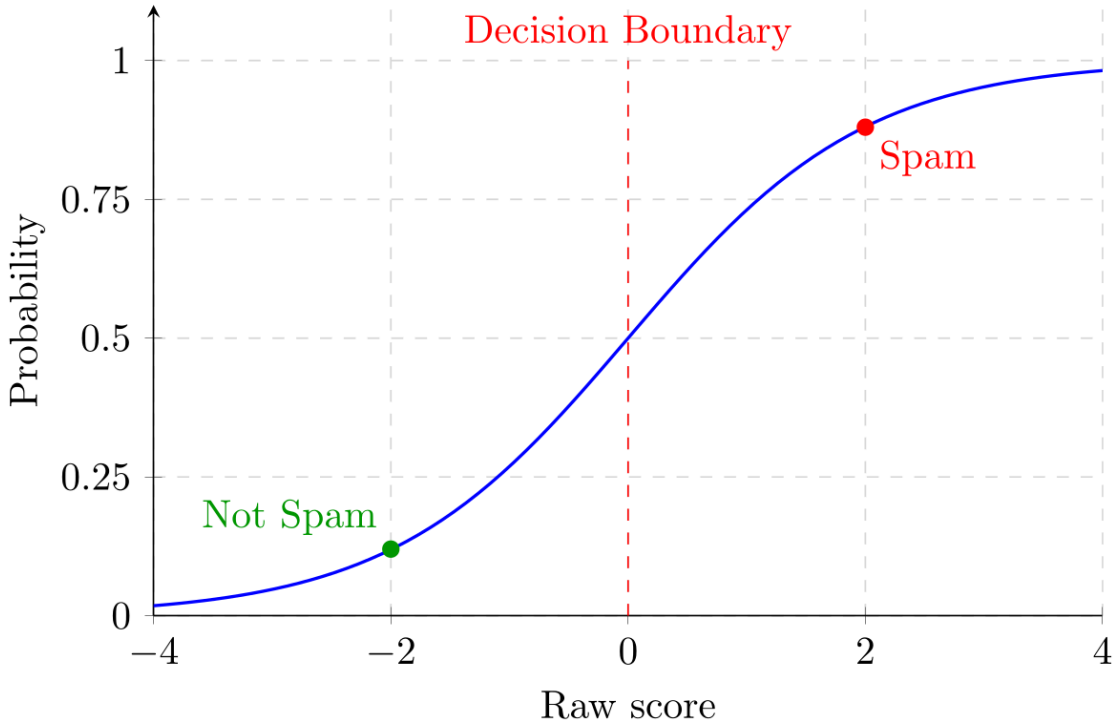
1. Extract numeric features
2. Multiply by weights
3. Sum weighted features
4. Convert to probability



The Sigmoid Function

Converts any number into a probability (0-1):

```
def sigmoid(x):
    return 1 / (1 + torch.exp(-x))
```



Mathematical Formulation

Our optimization problem:

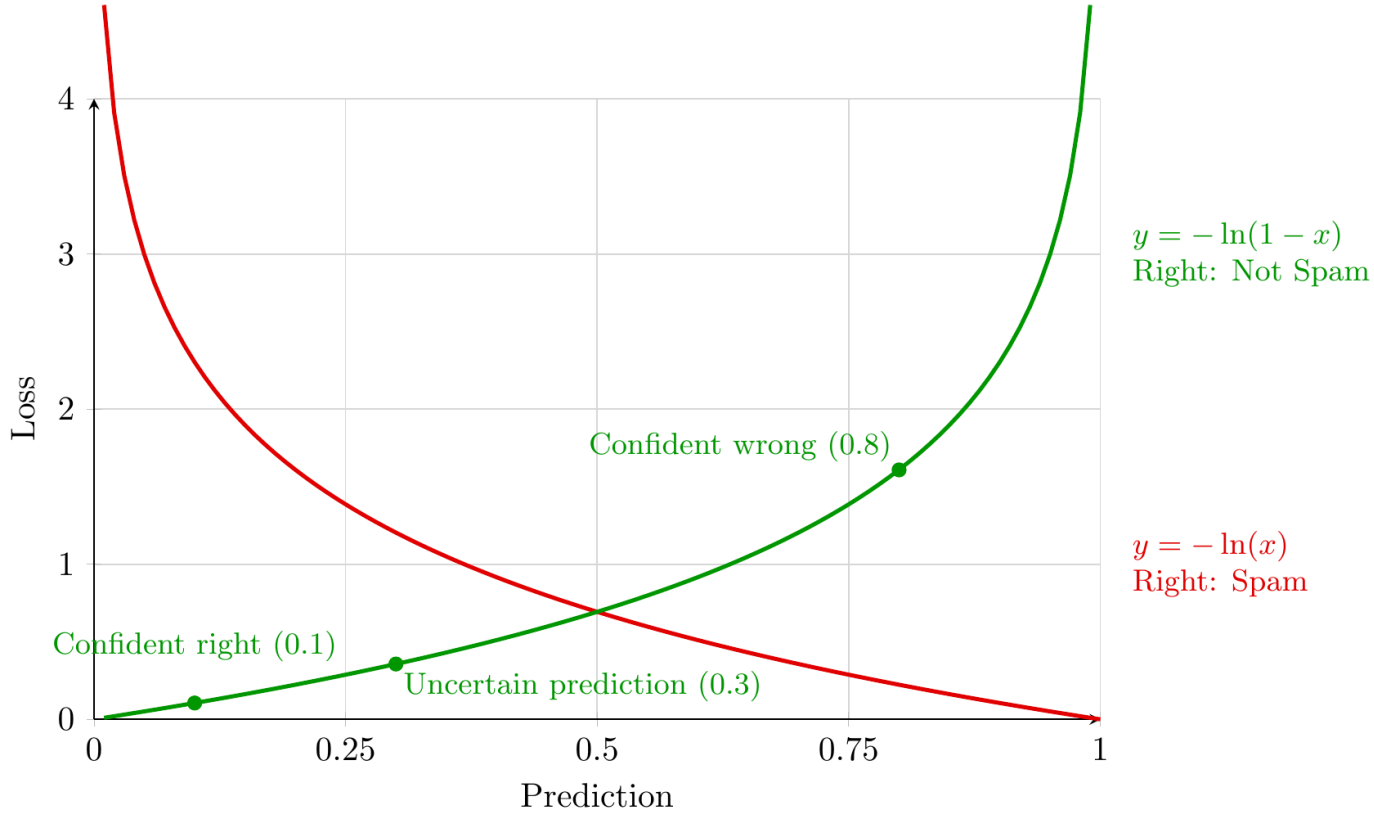
$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} \left[-y_i \log(\sigma(x_i^\top w)) - (1 - y_i) \log(1 - \sigma(x_i^\top w)) \right]$$

Where:

- w = weights vector
- x_i = feature vector
- y_i = true label (0/1)
- σ = sigmoid function

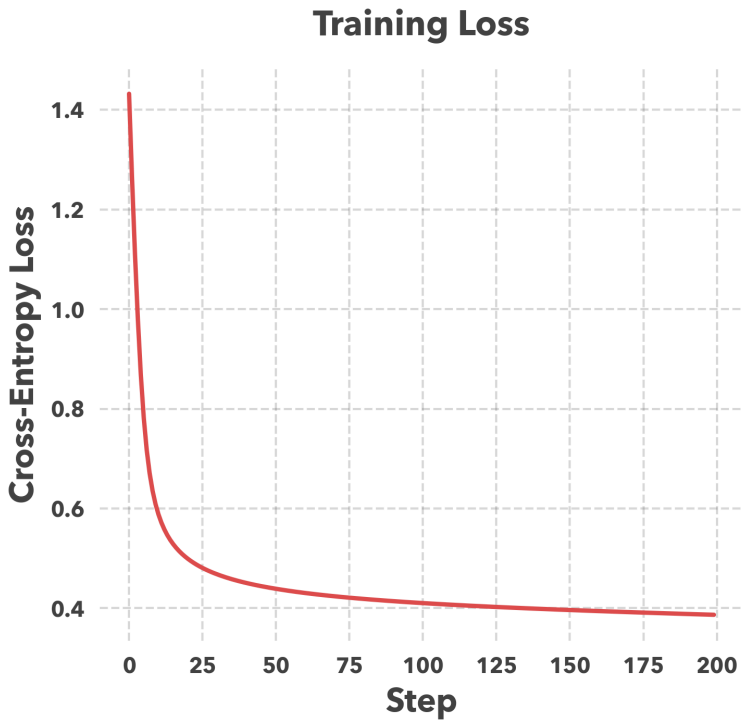
Cross-Entropy Loss

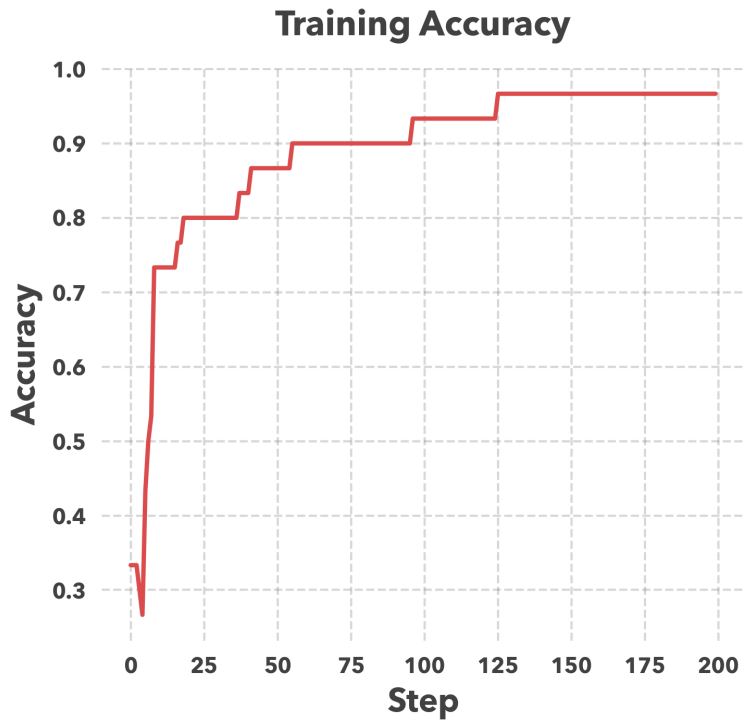
- Penalizes wrong predictions
- Rewards confident correct predictions
- Creates balanced learning

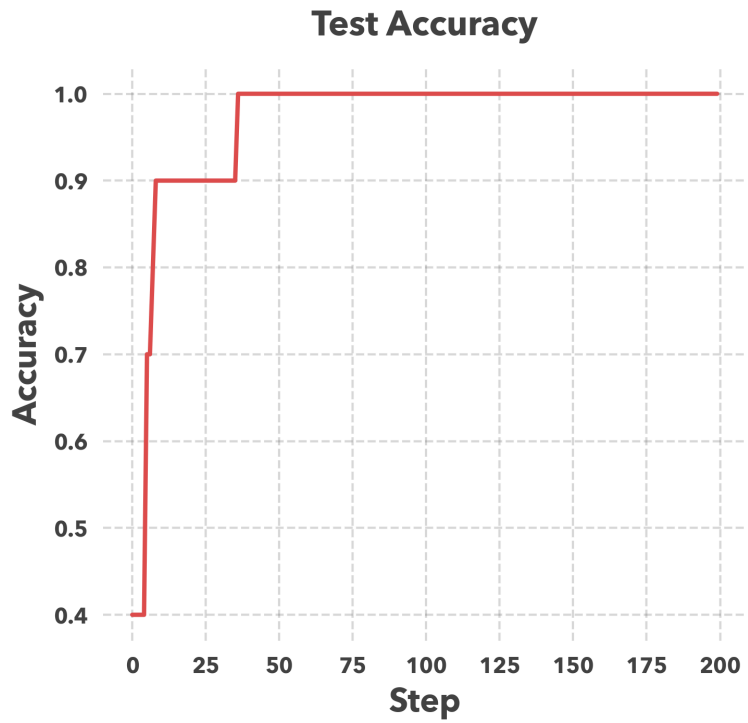


Implementation in PyTorch

```
# Initialize
weights = torch.randn(5, requires_grad=True)
learning_rate = 0.01

for _ in range(1000):
    # Forward pass
    predictions = spam_score(features, weights)
    loss = cross_entropy_loss(predictions, true_labels)

    # Backward pass
    loss.backward()

    # Update weights
    with torch.no_grad():
        weights -= learning_rate * weights.grad
        weights.grad.zero_()
```


Training Loss
Cross-Entropy Loss
1.4
1.2
1.0
0.8
0.6
0.4
0
25
50
75
100
125
150
175
200
Step


Training Accuracy
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
25
50
75
100
125
150
175
200
Step


Test Accuracy
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
25
50
75
100
125
150
175
200
Step

Course Structure

1. Linear algebra & direct methods
2. Problem formulations & classical software
3. Calculus for optimization
4. Automatic differentiation & PyTorch
5. First-order methods
6. Second-order methods
7. Advanced topics
8. Modern deep learning practice

Learning Outcomes

By course end, you'll be able to:

1. Model real problems as optimization problems
2. Select appropriate algorithms
3. Implement solutions in PyTorch
4. Apply optimization to practical problems
5. Conduct optimization research

Getting Started

- Review the syllabus
- Set up Python environment
- Try the Colab notebook
- Start thinking about project ideas

Questions?

- Course website: URL
- Office hours: Listed on the course website
- Email: Address
- Discord: Check email for invite.