Week 00a Basic Handout: ML Foundations - Learning from Data

Machine Learning for Smarter Innovation

1 Week 00a Basic Handout: ML Foundations - Learning from Data

1.1 For Students With: No prior ML knowledge, basic programming concepts

1.2 Overview

This handout introduces machine learning fundamentals without requiring mathematical background. Focus on concepts, real-world examples, and practical understanding.

1.3 Part 1: The Big Idea - Machines That Learn

1.3.1 What is Machine Learning?

Machine learning means teaching computers to find patterns in data instead of programming explicit rules.

Traditional Programming: - Programmer writes: IF spam word THEN mark as spam - Works for simple cases - Breaks with edge cases

Machine Learning: - Computer discovers: "These 500 emails are spam, these 500 are not spam" - Learns patterns automatically - Adapts to new patterns

1.3.2 Why Machine Learning Now?

Three factors came together: 1. **Data** - We generate billions of data points daily 2. **Computing** - GPUs can process massive datasets 3. **Algorithms** - Better learning methods discovered

---

1.4 Part 2: The Three Learning Styles

1.4.1 1. Supervised Learning (Learning with a Teacher)

Example: Spam detection - **Input**: Email text - **Label**: "Spam" or "Not Spam" - **Goal**: Learn to classify new emails

Real Applications: - Email spam filters (Gmail) - Fraud detection (credit cards) - Medical diagnosis (X-ray analysis) - Price prediction (real estate)

When to Use: - You have labeled examples (input + correct answer) - You want to predict or classify new data - Pattern is consistent over time

1.4.2 2. Unsupervised Learning (Learning without a Teacher)

Example: Customer segmentation - **Input**: Customer purchase history - **Label**: None - **Goal**: Discover natural customer groups

Real Applications: - Customer segmentation (marketing) - Anomaly detection (fraud, errors) - Recommendation systems (Netflix genres) - Document organization (Google News topics)

When to Use: - No labels available - Want to discover hidden patterns - Explore data structure

1.4.3 3. Reinforcement Learning (Learning by Trial and Error)

Example: Game playing - **Input**: Game state - **Action**: Make a move - **Reward**: Win/loss/draw - **Goal**: Learn winning strategy

Real Applications: - Game AI (AlphaGo, chess) - Robotics (walking, grasping) - Self-driving cars (navigation) - Resource optimization (data centers)

When to Use: - Sequential decisions matter - Learn from interaction - Delayed rewards

---

1.5 Part 3: How Does Learning Work?

1.5.1 The Learning Process

1. **Collect Data** - Gather examples (emails, images, sensor readings)
2. **Choose Model** - Pick learning algorithm (linear, tree, neural network)
3. **Train** - Show model examples, adjust internal parameters
4. **Evaluate** - Test on NEW data (not training data!)
5. **Deploy** - Use in real world

1.5.2 Key Insight: Generalization

Goal: Perform well on NEW data, not just training data

Bad: Memorizing training data (overfitting) **Good**: Learning underlying pattern (generalizing)

Analogy: Student preparing for exam - Memorizing exact practice problems = overfitting - Understanding concepts = generalizing

---

1.6 Part 4: Success Stories

1.6.1 Email Spam Detection

- **Before ML**: Rule-based filters caught 60% of spam
- **After ML**: Gmail catches 99.9% of spam
- **Why**: Learns new spam patterns automatically

1.6.2 Image Recognition

- **Before ML**: Hand-coded rules, poor accuracy
- **After ML**: 95%+ accuracy on ImageNet
- **Breakthrough**: Deep learning (2012)

1.6.3 Language Translation

- **Before ML**: Rule-based translation, awkward output
- **After ML**: Google Translate uses neural networks
- **Result**: Human-level quality for many language pairs

1.7 Part 5: When NOT to Use Machine Learning

1.7.1 ML is NOT the answer when:

1. **Simple rules work** - Don't use ML to add two numbers
2. **No data available** - Need thousands+ examples minimum
3. **Explainability critical** - Medical/legal may require transparent logic
4. **Data changes rapidly** - Model becomes outdated quickly
5. **Cost exceeds benefit** - Training can be expensive

1.7.2 Traditional Programming is Better For:

- Calculations (tax computation)
- Exact logic (password validation)
- Known algorithms (sorting, searching)
- Zero-tolerance errors (banking transactions)

1.8 Part 6: Common Pitfalls (What Can Go Wrong)

1.8.1 1. Not Enough Data

Problem: Model can't learn pattern from 10 examples **Solution**: Collect more data (thousands minimum)

1.8.2 2. Biased Data

Problem: Training on non-representative data **Example**: Facial recognition trained only on light-skinned faces **Solution**: Diverse, balanced datasets

1.8.3 3. Overfitting

Problem: Model memorizes training data, fails on new data **Symptom**: 100% training accuracy, 50% test accuracy **Solution**: Regularization, more data, simpler model

1.8.4 4. Wrong Metric

Problem: Optimizing accuracy when precision matters **Example**: Cancer detection needs high recall (catch all cancers) **Solution**: Choose metric matching business goal

1.8.5 5. Data Leakage

Problem: Test data accidentally in training set **Result**: Falsely optimistic performance **Solution**: Strict train/test separation

---

1.9 Part 7: Practical Checklist

1.9.1 Before Starting ML Project:

- ☐ Do I have enough labeled data? (1000+ examples minimum)
- ☐ Is there a pattern to learn? (not random noise)
- ☐ Can I measure success clearly? (accuracy, profit, etc.)
- ☐ Is data representative of real-world use?
- ☐ Do I have computational resources? (GPU for deep learning)
- ☐ Is model interpretability required?
- ☐ What happens if prediction is wrong? (risk assessment)

1.9.2 Good First ML Projects:

1. **Classification**: Email spam, sentiment analysis
2. **Regression**: Price prediction, demand forecasting
3. **Clustering**: Customer segmentation, document grouping

1.9.3 Avoid As First Project:

- Real-time systems (latency critical)
- Safety-critical applications (medical, automotive)
- Highly imbalanced data (fraud: 0.1% positive rate)
- Complex sequential decisions (reinforcement learning)

1.10 Key Takeaways

1. **Learning from Data**: ML discovers patterns automatically vs hand-coded rules
2. **Three Styles**: Supervised (with labels), Unsupervised (find patterns), Reinforcement (trial and error)
3. **Generalization**: Goal is new data performance, not memorization
4. **When to Use**: Abundant data, clear patterns, measurable outcomes
5. **When NOT to Use**: Simple rules work, no data, explainability critical
6. **Common Pitfalls**: Insufficient data, overfitting, bias, wrong metric

---

1.11 Next Steps

- **Week 00b**: Supervised Learning algorithms (regression, trees, ensembles)
- **Hands-On**: Try scikit-learn tutorials (spam detection, iris classification)
- **Reading**: "Machine Learning Yearning" by Andrew Ng (free PDF)

1.12 Glossary (Plain English)

- **Algorithm**: Step-by-step learning procedure
- **Feature**: Input variable (age, income, word count)
- **Label**: Correct answer for training example
- **Model**: Learned pattern from data
- **Training**: Process of learning from data
- **Testing**: Evaluating on new data

- **Overfitting**: Memorizing instead of learning
- **Generalization**: Working well on new data
- **Supervised**: Learning with labeled examples
- **Unsupervised**: Finding patterns without labels