Module 4: Logistic Regression

Overview

Logistic regression is the foundational algorithm for binary classification—predicting which of two classes an example belongs to. This module introduces the sigmoid function for converting linear outputs to probabilities, decision boundaries for separating classes, and the cross-entropy loss function designed for classification. The gradient descent training procedure is nearly identical to linear regression.

---

1. Classification vs. Regression

The Difference

Regression: Predict a continuous value - Example: Predict house price ($350,000) - Output: Any real number

Classification: Predict a discrete class - Example: Predict spam or not spam - Output: One of k categories

Binary Classification

The simplest classification: exactly two classes. - Spam / Not spam - Pass / Fail - Cat / Dog - Positive / Negative

We typically encode classes as 0 and 1.

Why Not Linear Regression?

You might try: train a linear regression and round the output.

Problems: - Outputs aren't bounded (could predict -5 or 100) - Squared error is wrong (predicting 0.9 vs 1.0 shouldn't be penalized much) - Gradients are poor far from decision boundary

We need a different approach.

---

2. The Sigmoid Function

Definition

The sigmoid (logistic) function maps any real number to the interval (0, 1):

```
(z) = 1 / (1 + e^(-z))
```

Properties

- **Output range:** Always between 0 and 1 (exclusive)
- **Monotonic:** Higher z higher output

- **Symmetric:** σ(-z) = 1 - σ(z)
- **Midpoint:** σ(0) = 0.5
- **Smooth:** Differentiable everywhere

Key Values

z	σ(z)
-∞	□ 0
-5	0.0067
0	0.5
+5	0.9933
+∞	□ 1

Interpretation

The sigmoid output is interpreted as a probability:

```
P(y = 1 | x) =  (w · x + b)
```

“Given features x, the probability that the class is 1.”

---

3. The Logistic Regression Model

Two Steps

1. **Linear combination:** z = w · x + b
2. **Sigmoid activation:** p = σ(z)

Combined:

```
P(y = 1 | x) =  (w · x + b)
```

Making Predictions

Compare probability to a threshold (default 0.5):

```
If p   0.5: predict class 1
If p < 0.5: predict class 0
```

Example

Features: study hours = 6 Parameters: w = 0.5, b = -2.5

```
z = 0.5 × 6 + (−2.5) = 0.5
p =  (0.5) = 0.62
Prediction: 1 (pass) since 0.62   0.5
```

---

4. Decision Boundaries

What Is a Decision Boundary?

The decision boundary is the set of points where the model is undecided—where P(y=1) = 0.5.

Since σ(z) = 0.5 when z = 0, the decision boundary is:

$$w \cdot x + b = 0$$

Geometric Interpretation

For 2D features (x_1, x_2), the boundary is a line:

$$w_1 x_1 + w_2 x_2 + b = 0$$

For 3D, it's a plane. In general, a hyperplane.

Linear Separability

Logistic regression can only separate classes with a linear boundary. If the true boundary is curved, logistic regression will underfit. This is addressed by: - Feature crosses (create curved boundaries in original space) - Neural networks (learn non-linear boundaries)

---

5. Classification Threshold

Adjusting the Threshold

The default threshold of 0.5 can be changed based on requirements.

Lower threshold (e.g., 0.3): - More examples classified as positive - More true positives, but also more false positives - Use when missing positives is costly (disease detection)

Higher threshold (e.g., 0.7): - Fewer examples classified as positive - Fewer false positives, but also fewer true positives - Use when false positives are costly (spam filter)

Threshold Selection

Choose based on: - Cost of false positives vs. false negatives - Required precision or recall - ROC curves and precision-recall curves (Module 5)

---

6. Logistic Loss (Cross-Entropy)

Why Not MSE?

MSE for classification has problems: - Gradient is small when predictions are very wrong - Doesn't properly measure probability predictions

Binary Cross-Entropy

The proper loss for binary classification:

```
L = -[y × log(p) + (1-y) × log(1-p)]
```

For the entire dataset:

```
L = -(1/n) × Σ[y  × log(p ) + (1-y ) × log(1-p )]
```

Intuition

When y = 1: Loss = -log(p) - If p □ 1: Loss □ 0 (correct with confidence) - If p □ 0: Loss □ ∞ (wrong with confidence)

When y = 0: Loss = -log(1-p) - If p □ 0: Loss □ 0 (correct) - If p □ 1: Loss □ ∞ (wrong)

Cross-entropy heavily penalizes confident wrong predictions.

---

7. Training with Gradient Descent

The Gradient

Remarkably, the gradient of logistic loss has the same form as linear regression:

```
L/ w = (1/n) × X  @ (predictions - labels)
L/ b = (1/n) × Σ(predictions - labels)
```

The sigmoid “absorbs” into the math nicely.

Training Algorithm

```
Initialize w = 0, b = 0

For each iteration:
    # Forward pass
    z = X @ w + b
    predictions = sigmoid(z)

    # Compute loss (for monitoring)
    loss = cross_entropy(labels, predictions)

    # Backward pass
    errors = predictions - labels
    grad_w = (1/n) × X.T @ errors
    grad_b = (1/n) × sum(errors)

    # Update
    w = w - learning_rate × grad_w
    b = b - learning_rate × grad_b
```

Same Concepts Apply

- Learning rate matters
- Batch/SGD/mini-batch all work
- Convergence criteria similar

---

8. The Sigmoid Derivative

Used in Backpropagation

While not needed for simple logistic regression, the sigmoid derivative is important for neural networks:

```
'(z) =  (z) × (1 −  (z))
```

The Vanishing Gradient Problem

Notice that σ'(z) is maximized at z=0 (where σ'(0) = 0.25) and approaches 0 for large |z|. This causes the "vanishing gradient" problem in deep networks, motivating alternative activations like ReLU (Module 6).

Key Takeaways

1. **Classification predicts classes**, not continuous values
2. **Sigmoid squashes** any number to (0, 1), interpretable as probability
3. **Logistic regression** = linear combination + sigmoid
4. **Decision boundary** is where w·x + b = 0
5. **Threshold** can be adjusted based on error costs
6. **Cross-entropy loss** properly measures classification errors
7. **Training uses same gradient descent** as linear regression
8. **Linear boundaries** are the main limitation

Connections to Future Modules

- **Module 5**: Multiple classes □ softmax and multiclass
- **Module 5**: Better metrics □ precision, recall, F1
- **Module 6**: Non-linear boundaries □ neural networks
- **Module 12**: When classification is unfair □ fairness

Further Reading

- [Links to be added]
- Introduction to Statistical Learning, Chapter 4

- Andrew Ng's Machine Learning Course: Logistic Regression