Assignment 4 Report: Perceptron Learning Approaches Name: Shivam Pawar Course: MATH-485 Introduction This assignment explores two approaches to Perceptron learning: the Heuristic Perceptron and the Gradient Descent Perceptron. Both are implemented and analyzed on a 2D binary classification dataset. The dataset contains 100 samples with two numerical features and a binary Part 1: Heuristic Perceptron Code Snippet class HeuristicPerceptron: def __init__(self, eta=0.1, max_iter=1000): self.eta = eta self.max_iter = max_iter self.w = None self.b = None self.boundaries = [] def _init_weights(self, X): n_features = X.shape[1] self.w = np.random.normal(loc=0.0, scale=0.01, size=n_features) self.b = np.random.normal(loc=0.0, scale=0.01) self.boundaries.append((self.w.copy(), self.b, 'initial')) def fit(self, X, y): self._init_weights(X) iterations = 0errors = True while errors and iterations < self.max_iter:</pre> errors = False for xi, target in zip(X, y): prediction = self.predict(xi.reshape(1, -1))[0] if prediction != target: update = self.eta * (target - prediction) self.w += update * xi self.b += update errors = **True** if errors: self.boundaries.append((self.w.copy(), self.b, 'intermediate')) iterations += 1 self.boundaries.append((self.w.copy(), self.b, 'final')) return iterations Visualizations and Analysis Figure 1: Dataset Visualization **Dataset Visualization** Class 0 1.0 • • Class 1 0.8 -0.6 Feature 2 0.2 -0.0 -0.2 0.8 0.0 0.4 1.0 0.6 Feature 1 • Red points: Class 0, Blue points: Class 1 • Data is linearly separable and suitable for a perceptron. Figure 2: Heuristic Perceptron (Ir=0.01) Perceptron Decision Boundaries Learning rate: 0.01, Iterations: 1000 2.0 Class 0 Class 1 Initial boundary Final boundary Feature 2 0.5 0.0 -1.00.0 1.0 0.5 1.5 -0.5 2.0 Feature 1 • Very slow learning. Many small updates. • Final boundary is accurate but convergence is slow. Figure 3: Heuristic Perceptron (Ir=0.1) Perceptron Decision Boundaries Learning rate: 0.1, Iterations: 1000 2.0 Class 0 Class 1 Initial boundary 1.5 --- Final boundary Feature 2 0.0 -0.5-1.01.0 -0.5 0.0 0.5 1.5 2.0 Feature 1 • Faster convergence. Fewer intermediate updates. • Final boundary is clear and balanced. Optimal result. Figure 4: Heuristic Perceptron (Ir=1.0) Perceptron Decision Boundaries Learning rate: 1.0, Iterations: 1000 2.0 Class 0 Class 1 Initial boundary 1.5 Final boundary Feature 2 -0.5 -1.0-0.50.0 0.5 1.0 2.0 1.5 Feature 1 • Aggressive jumps in boundaries. • Converged quickly, but less stable during training. Results Table Learning Rate Iterations Final Weights Final Bias [-0.0885, -0.1132] 0.0891 0.01 1000 0.1 1000 [-0.8096, -1.0734] 0.8111 1.0 [-7.7065, -10.8941] 8.0098 1000 Part 2: Gradient Descent Perceptron Code Snippet class GradientDescentPerceptron: def __init__(self, eta=0.1, n_epochs=100): self.eta = eta self.n_epochs = n_epochs self.w = None self.b = None self.boundaries = [] self.cost_history = [] self.epoch_records = [] def fit(self, X, y): self._init_weights(X) $n_samples = X.shape[0]$ self.cost_history.append(self.compute_cost(X, y)) self.epoch_records.append(0) for epoch in range(1, self.n_epochs + 1): output = self.activation(X) grad_w = np.dot(X.T, error) / n_samples grad_b = np.sum(error) / n_samples self.w -= self.eta * grad_w self.b -= self.eta * grad_b **if** epoch % 10 == 0: self.cost_history.append(self.compute_cost(X, y)) self.epoch_records.append(epoch) self.boundaries.append((self.w.copy(), self.b, 'final')) Visualizations and Analysis Figure 5-6: GD Perceptron (Ir=0.01) Perceptron Decision Boundaries Learning rate: 0.01, Epochs: 100 • Class 0 Class 1 Initial boundary — Final boundary 1.0 Feature 2 0.0 -0.5 1.0 0.5 -0.50.0 1.5 2.0 Feature 1 Error Plot (Every 10 Epochs) 0.750 0.745 Binary Cross-Entropy 0.735 0.725 0.720 80 20 60 100 Epochs Boundary updates are slow and subtle. Loss decreases slowly but steadily. Figure 7–8: GD Perceptron (Ir=0.1) Perceptron Decision Boundaries Learning rate: 0.1, Epochs: 100 Class 0 Class 1 Initial boundary 1.5 — Final boundary 1.0 Feature 2 0.0 -0.5 0.5 1.0 1.5 -0.5 2.0 0.0 Feature 1 Error Plot (Every 10 Epochs) 0.80 Binary Cross-Entropy 0.72 0.70 0.68 40 60 100 20 80 Epochs • Good trade-off between speed and stability. • Fast and stable convergence. Figure 9–10: GD Perceptron (Ir=1.0) Perceptron Decision Boundaries Learning rate: 1.0, Epochs: 100 2.0 Class 0 Class 1 Initial boundary 1.5 - Final boundary 1.0 Feature 2 0.0 -0.5-1.00.5 1.0 -0.5 0.0 1.5 2.0 Feature 1 Error Plot (Every 10 Epochs) 1.1 -1.0 -0.9 Binary Cross-Entropy 0.5 0.4 -0.3 -60 100 20 40 80 Epochs • Fast learning, but less smooth. Risk of instability if data were noisy. Figure 11-14: Epoch Comparison (Ir=0.1) Perceptron Decision Boundaries Learning rate: 0.1, Epochs: 50 2.0 Class 0 Class 1 Initial boundary 1.5 Final boundary 1.0 Feature 2 0.0 -0.5 0.5 1.0 0.0 1.5 2.0 -0.5Feature 1 Error Plot (Every 10 Epochs) 0.73 0.72 Binary Cross-Entropy 0.69 0.68 20 30 10 50 40 ò Epochs Perceptron Decision Boundaries Learning rate: 0.1, Epochs: 200 Class 0 Class 1 Initial boundary 1.5 — Final boundary 1.0 Feature 2 -0.5-1.00.5 0.0 1.0 2.0 -0.51.5 Feature 1 Error Plot (Every 10 Epochs) 0.675 0.650 0.625 Binary Cross-Entropy 0.600 0.575 -0.550 0.525 0.500 0.475 100 125 150 175 25 75 Epochs • Epoch 50: Underfitting, high loss. • Epoch 100: Balanced learning. • Epoch 200: Slight improvement, diminishing returns. Results Table Learning Rate Epochs Final Weights Final Bias Final Loss 0.01 [1.4314, -1.1076] -0.0625 0.7213 0.1 [0.8021, -0.9818] -0.1373 0.6821 [-3.5249, -3.1071] 3.3478 1.0 0.3234 0.6808 0.1 [-0.4394, 0.1654] -0.1184

• Heuristic Perceptron is simple and intuitive, suitable for small datasets.

• Gradient Descent Perceptron offers smoother convergence and loss tracking. • Learning rate and number of epochs significantly affect convergence quality.

Conclusion

[-1.9050, -1.2881] 1.4938 0.1 0.4734

| Gradient descent outperforms heuristic in terms | rms of training stability and loss reduction. | | |
|---|---|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |