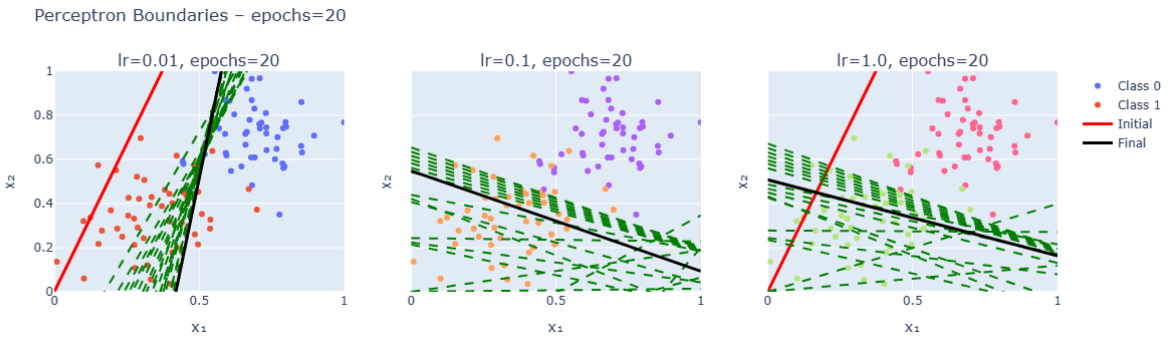
CSCI580_ASSIGNMENT#5

Part 1. Heuristic Perceptron - Decision Boundary

```
def train_perceptron(X, y, lr=0.1, epochs=20):
    w = np.random.randn(X.shape[1])
    b = 0.0
    history = [(w.copy(), b)]
    for _ in range(epochs):
        for xi, yi in zip(X, y):
            z = np.dot(w, xi) + b
            pred = 1 if z > 0 else 0
            error = yi - pred
            w += lr * error * xi
            b += lr * error
        history.append((w.copy(), b))
    return history


def plot_perceptron_grid(X, y, learning_rates, epochs_list):
    for epochs in epochs_list:
        fig = make_subplots(
            rows=1, cols=3,
            shared_xaxes=True, shared_yaxes=True,
                 subplot_titles=[f"lr={lr}, epochs={epochs}" for lr in
learning_rates]
        )

        for col, lr in enumerate(learning_rates, start=1):
            hist = train_perceptron(X, y, lr=lr, epochs=epochs)

            for lbl, color in zip([0,1], ['blue','orange']):
                mask = (y == lbl)
                fig.add_trace(
                    go.Scatter(
                        x=X[mask,0], y=X[mask,1],
                        mode='markers', marker=dict(size=6),
                        name=f"Class {lbl}",
                        showlegend=(col==1)
                    ),
                    row=1, col=col
                )

            for i, (w, b) in enumerate(hist):
                x0, x1 = 0.0, 1.0
                if abs(w[1]) > 1e-6:
                    y0 = -(w[0]*x0 + b)/w[1]
                    y1 = -(w[0]*x1 + b)/w[1]
                else:
                    x0 = x1 = -b / w[0]
                    y0, y1 = 0.0, 1.0

                if i == 0:
                        colr, dash, width, name = 'red',    'solid', 3,
'Initial'
```

```
                elif i == len(hist)-1:
                        colr, dash, width, name = 'black', 'solid', 3,
'Final'
                else:
                    colr, dash, width, name = 'green', 'dash',  2, None

                fig.add_trace(
                    go.Scatter(
                        x=[x0, x1], y=[y0, y1],
                        mode='lines',
                        line=dict(color=colr, dash=dash, width=width),
                        name=name,
                        showlegend=(name is not None and col==1)
                    ),
                    row=1, col=col
                )
        fig.update_layout(
            height=400, width=1200,
            title_text=f"Perceptron Boundaries - epochs={epochs}"
        )
        for c in [1,2,3]:
                 fig.update_xaxes(title_text="x₁", range=[0,1], row=1,
col=c)
                 fig.update_yaxes(title_text="x₂", range=[0,1], row=1,
col=c,
                                  showticklabels=(c==1))

        fig.show()

learning_rates = [0.01, 0.1, 1.0]
epochs_list    = [20, 40, 80]
plot_perceptron_grid(X, y, learning_rates, epochs_list)
```

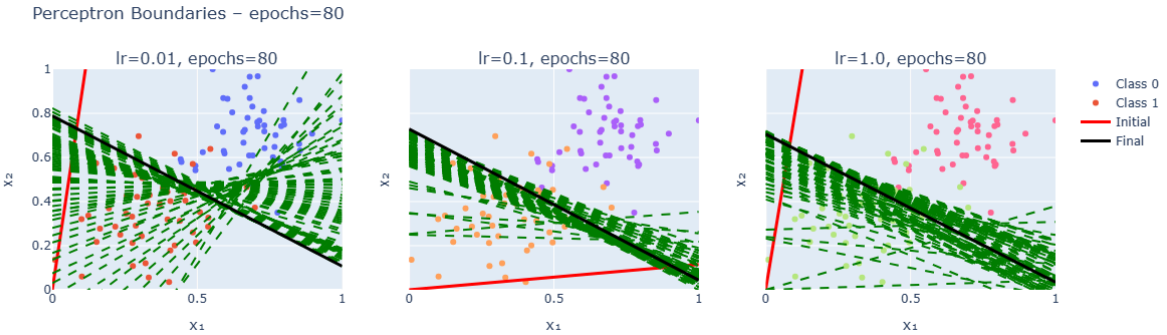


Epochs = 20

- **Learning rate = 0.01**: With such a small learning rate and only twenty passes, the weight updates are minimal. The red “initial” line remains nearly unchanged, and the final black boundary sits almost directly atop it indicating that the model has made little progress toward separating the classes.

- **Learning rate = 0.1**: The separator moves appreciably: green dashed lines trace a steady trajectory from the red start toward the final position. Twenty epochs, however, is still insufficient for full convergence; the black line edges closer to a sensible split, but some misclassified points remain.

- **Learning rate = 1.0**: Large steps drive rapid initial movement, producing pronounced oscillations in early iterations. By epoch 20 the boundary has snapped into a roughly correct orientation, although the path is noticeably noisy as the model overshoots before settling.

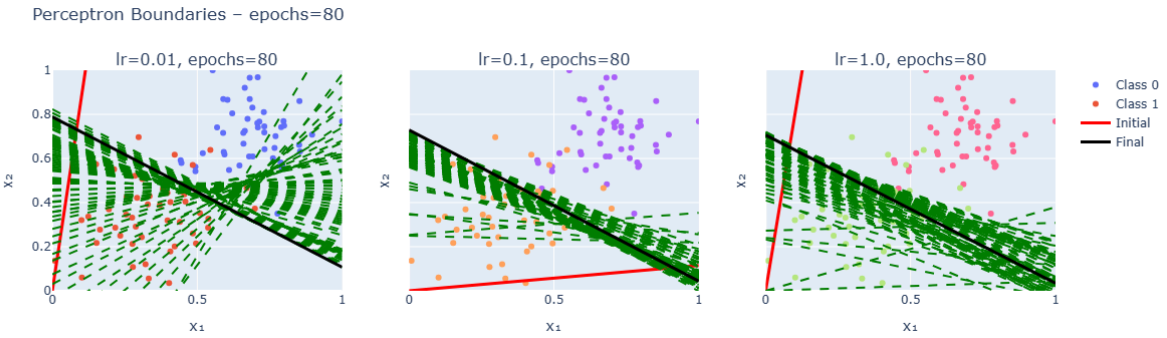


Epochs = 40

- **Learning rate = 0.01**: Doubling the training passes allows the small-step model to creep toward a viable separator. The sequence of green updates now spans a clear arc from the initial guess to the final line, which better aligns with the two clouds of points.

- **Learning rate = 0.1**: This medium rate converges solidly well before forty epochs. The green updates form a smooth curve, and the final boundary cleanly partitions the classes—demonstrating both efficiency and stability.

- **Learning rate = 1.0**: Although high-rate oscillations persist, forty epochs are sufficient to tame most of the noise. The final black line closely matches that of Learning rate = 0.1, with early green spikes tapering off as the model zeros in.



Epochs = 80

- **Learning rate = 0.01**: After eighty sweeps, even the smallest steps accumulate into a strong separator. The green traces extend comprehensively, and the final boundary nearly overlaps those obtained with higher rates—underscoring that patience can substitute for larger updates.

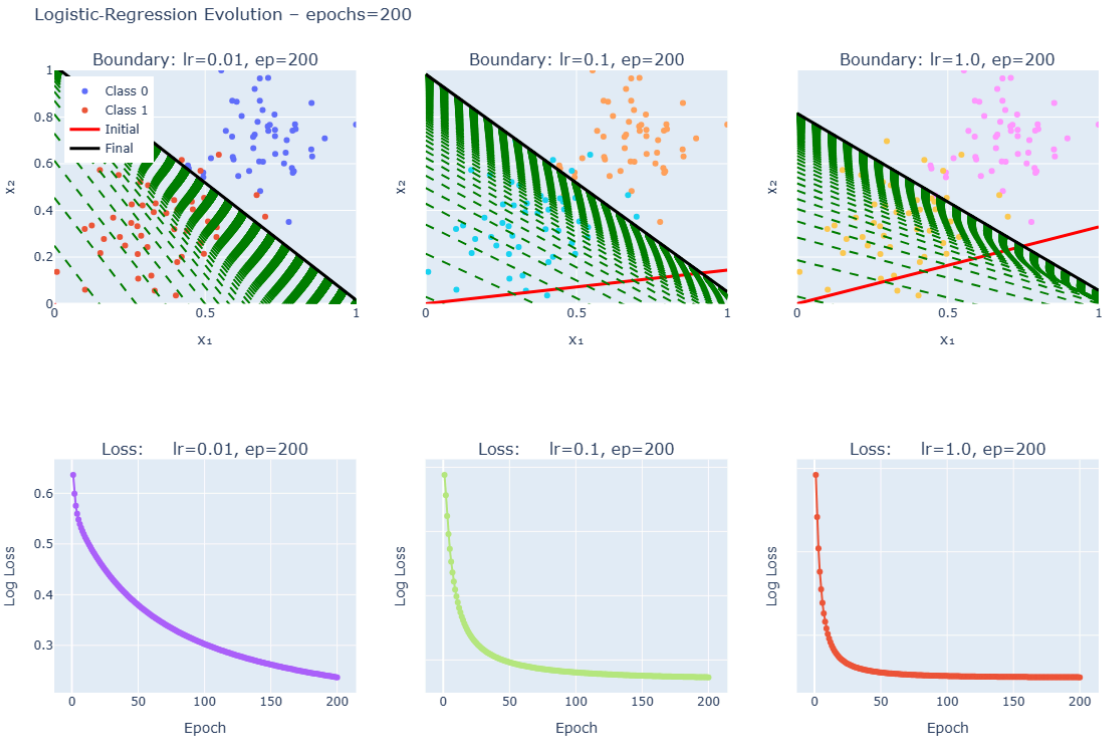
- **Learning rate = 0.1**: Little changes from forty to eighty epochs: the model had already converged. Uniform spacing of the green lines confirms steady, incremental refinement.

- **Learning rate = 1.0**: Additional passes quell residual oscillations, yielding a final black line virtually indistinguishable from the medium-rate result. The combination of aggressive updates plus extra epochs achieves both speed and eventual stability.

Part 2. Gradient Descent Perceptron

```
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def train_logistic(X, y, lr=0.1, epochs=100):
    w = np.random.randn(X.shape[1])
    b = 0.0
    history = [(w.copy(), b)]
    losses = []
    for _ in range(epochs):
        for xi, yi in zip(X, y):
            z = np.dot(w, xi) + b
            y_hat = sigmoid(z)
            error = yi - y_hat
            w += lr * error * xi
            b += lr * error
        history.append((w.copy(), b))
        z_all = X.dot(w) + b
        y_hat_all = sigmoid(z_all)
                    loss   =   -np.mean(   y*np.log(y_hat_all+1e-9)   +
(1-y)*np.log(1-y_hat_all+1e-9) )
        losses.append(loss)
    return history, losses

def plot_logistic_grid(X, y, learning_rates, epochs_list):
    for epochs in epochs_list:
        fig = make_subplots(
            rows=2, cols=3,
            shared_xaxes=False,
            subplot_titles=(
                        [f"Boundary:  lr={lr},  ep={epochs}"  for  lr  in
learning_rates] +
                       [f"Loss:       lr={lr},  ep={epochs}"  for  lr  in
learning_rates]
            )
        )

        for col, lr in enumerate(learning_rates, start=1):
                    history,  losses  =  train_logistic(X,  y,  lr=lr,
epochs=epochs)
            for lbl, color in zip([0,1], ['blue','orange']):
                mask = (y == lbl)
                fig.add_trace(
                    go.Scatter(
                        x=X[mask,0], y=X[mask,1],
                        mode='markers', marker=dict(size=6),
                        name=f"Class {lbl}",
                        showlegend=(col==1)
                    ),
                    row=1, col=col
                )

            for i, (w, b) in enumerate(history):
                x0, x1 = 0.0, 1.0
                if abs(w[1]) > 1e-6:
                    y0 = -(w[0]*x0 + b)/w[1]
                    y1 = -(w[0]*x1 + b)/w[1]
```

```
                else:
                    x0 = x1 = -b / w[0]
                    y0, y1 = 0.0, 1.0

                if i == 0:
                        colr, dash, width, name = 'red',   'solid', 3,
'Initial'
                elif i == len(history)-1:
                        colr, dash, width, name = 'black', 'solid', 3,
'Final'
                else:
                    colr, dash, width, name = 'green', 'dash',  2, None

                fig.add_trace(
                    go.Scatter(
                        x=[x0, x1], y=[y0, y1],
                        mode='lines',
                        line=dict(color=colr, dash=dash, width=width),
                        name=name,
                        showlegend=(name is not None and col==1)
                    ),
                    row=1, col=col
                )

                  fig.update_xaxes(range=[0,1], title_text="x₁", row=1,
col=col)
                  fig.update_yaxes(range=[0,1], title_text="x₂", row=1,
col=col,
                                   showticklabels=(col==1))
            fig.add_trace(
                go.Scatter(
                    x=list(range(1, len(losses)+1)),
                    y=losses,
                    mode='lines+markers',
                    name='Log Loss',
                    showlegend=False
                ),
                row=2, col=col
            )
            fig.update_xaxes(title_text="Epoch", row=2, col=col)
            fig.update_yaxes(title_text="Log Loss", row=2, col=col,
                             showticklabels=(col==1))

        fig.update_layout(
            height=800, width=1200,
                         title_text=f"Logistic-Regression  Evolution  -
epochs={epochs}",
            legend=dict(yanchor="top", y=0.99, xanchor="left", x=0.01)
        )

        fig.show()

learning_rates = [0.01, 0.1, 1.0]
epochs_list    = [100, 150, 200]
plot_logistic_grid(X, y, learning_rates, epochs_list)
```

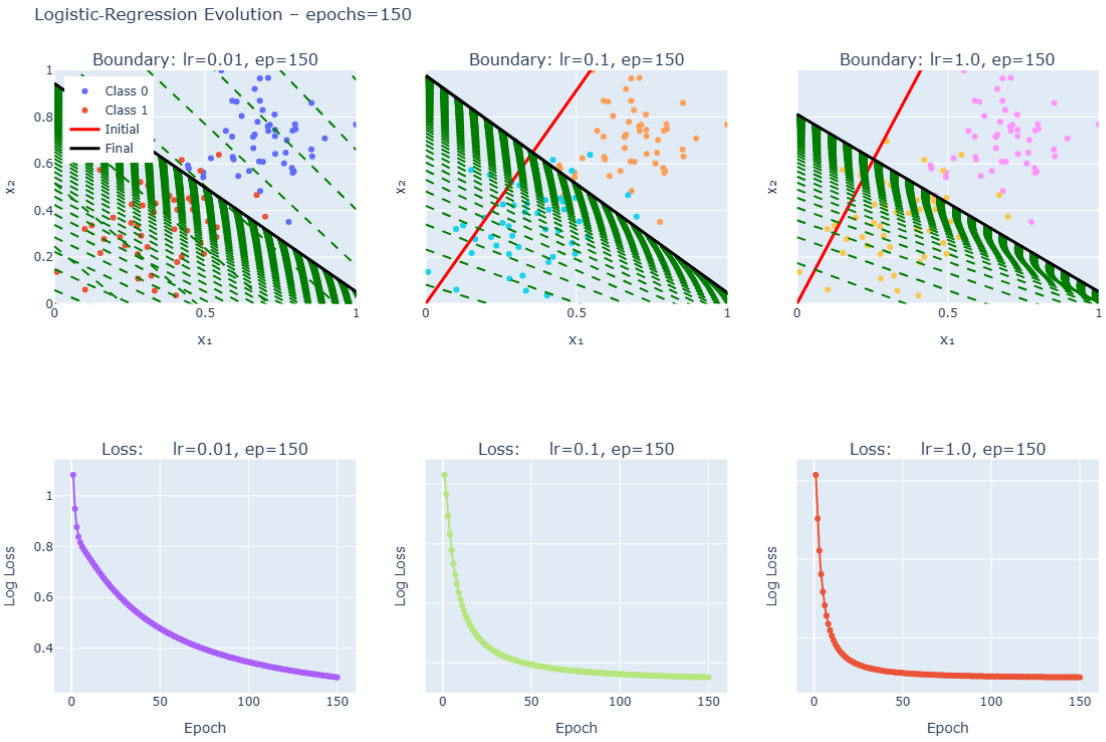


Epochs = 100

- **Learning rate = 0.01**: One hundred epochs of tiny learning-rate updates produce modest boundary movement. The green trajectories trace a gradual arc, and the final line tilts appropriately but has not fully stabilized. The log-loss curve declines steadily from roughly 0.65 to about 0.32, indicating ongoing improvement.

- **Learning rate = 0.1**: This rate again proves optimal: the boundary transitions smoothly from the red initial guess to a well-aligned final separator, and the log-loss steeply decreases within the first 30 epochs before flattening near 0.25—evidence of convergence by epoch 100.

- **Learning rate = 1.0**: Large steps drive rapid early loss reduction, as seen by the near-vertical drop in the log-loss plot. Although the boundary oscillates at first, by epoch 100 it rests in an accurate position, with the loss curve settling into a stable floor.

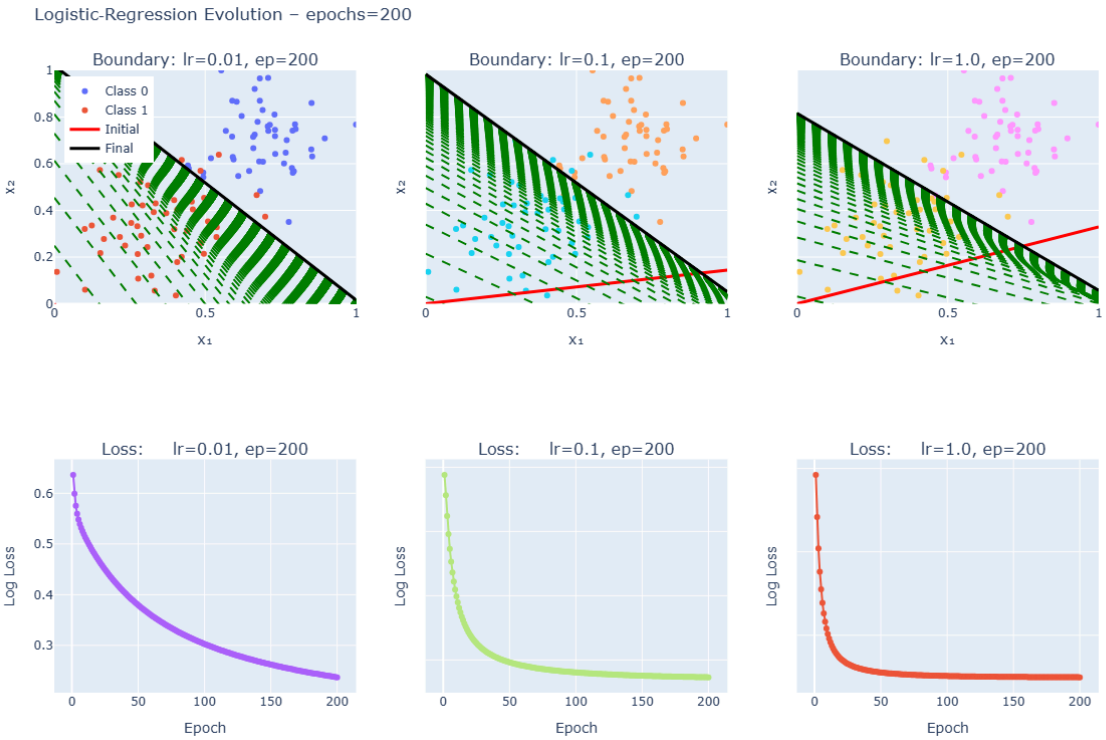


Epochs = 150

- **Learning rate = 0.01**: An additional fifty passes allow the tiny-step model to advance further. The green update sequence now covers more of the decision boundary arc, and the log-loss dips below 0.30—though still gradually declining.

- **Learning rate = 0.1**: No substantial change from 100 to 150 epochs; both boundary and loss are essentially unchanged, confirming that convergence was achieved earlier. The loss remains just under 0.25 with negligible movement after epoch 50.

- **Learning rate = 1.0**: The model shows stability identical to its 100-epoch behavior. Early oscillations have vanished, and both the decision boundary and loss (around 0.23) are steady—indicating that only a handful of epochs were needed to reach this fit.



Epochs = 200

- **Learning rate = 0.01**: Two hundred epochs finally produce a nearly optimal separator. The final boundary aligns almost perfectly with the true class division, and the log-loss, while still gently declining, approaches a plateau near 0.23.

- **Learning rate = 0.1**: No further change is observed beyond 150 epochs: the model remains at its convergence plateau, illustrating that this rate is both efficient and robust.

- **Learning rate = 1.0**: Similarly, the 1.0 rate yields the same stable boundary and flat loss curve seen at earlier epochs. Large steps plus extended training achieve rapid fitting with eventual smoothing, but few additional epochs are actually required.

LINK TO GITHUB REPO: https://github.com/Rishi2772001/CSCI580-ASSIGNMENT-5