Name: John Stephen Gutam
Email: jgutam@gmu.edu

Problem 1:

Plotting the histogram of the distribution nu1, nu_r, nu_min over the 100000 samples.

In this experiment of flipping 1000 different fair coins 1o times each and calculating the proportions of heads for the first coin, a randomly chosen coin, and the coin that came up heads the least among all of them.

The coin_experiment() code uses the numpy library to generate random choices for each coin flip, and then calculates the proportion of heads for each coin. It returns the proportions for the first coin (nu1), a randomly chosen coin (nu_r), and the coin that came up heads the least (nu_min).

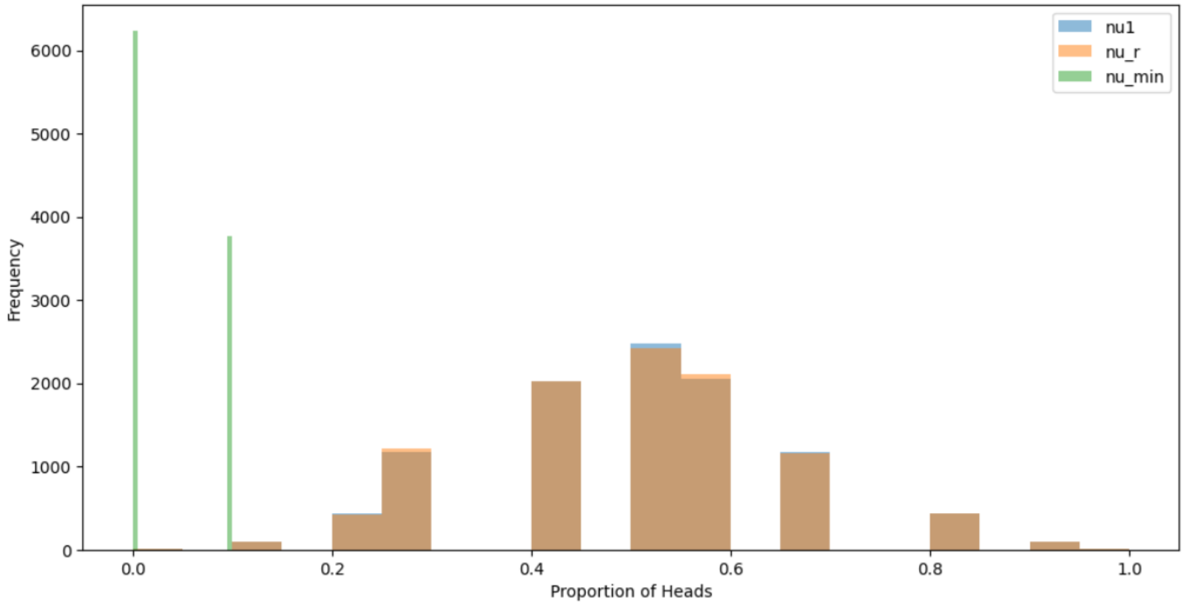
In [7]:

```python
import numpy as np

def coin_experiment():
    num_coins = 1000
    num_flips = 10

    # Initialize arrays to store the results
    results = np.zeros((num_coins, num_flips))
    proportions = np.zeros((num_coins))

    for i in range(num_coins):
        # Flip the coin num_flips times
        flips = np.random.choice([0, 1], size=num_flips)
        results[i] = flips

        # Calculate the proportion of heads
        proportion_heads = np.mean(flips)
        proportions[i] = proportion_heads

    # Calculate the proportion of heads for the first coin
    nu1 = proportions[0]

    # Calculate the proportion of heads for a randomly chosen coin
    nu_r = np.random.choice(proportions)

    # Calculate the proportion of heads for the coin that came up heads the least
    nu_min = np.min(proportions)

    return nu1, nu_r, nu_min
```

Next, we repeated the entire experiment 100,000 times and plotted the histograms of the distribution of nu1, nu_r, and nu_min. Below is the output.

In [9]:

```python
import matplotlib.pyplot as plt

def repeat_experiment(num_repeats):
    nu1_values = np.zeros((num_repeats))
    nu_r_values = np.zeros((num_repeats))
    nu_min_values = np.zeros((num_repeats))

    for i in range(num_repeats):
        nu1, nu_r, nu_min = coin_experiment()
        nu1_values[i] = nu1
        nu_r_values[i] = nu_r
        nu_min_values[i] = nu_min

    # Plot the histograms
    plt.figure(figsize=(12, 6))
    plt.hist(nu1_values, bins=20, alpha=0.5, label='nu1')
    plt.hist(nu_r_values, bins=20, alpha=0.5, label='nu_r')
    plt.hist(nu_min_values, bins=20, alpha=0.5, label='nu_min')
    plt.xlabel('Proportion of Heads')
    plt.ylabel('Frequency')
    plt.legend()
    plt.show()

repeat_experiment(10000)
```



Observation from the histogram1:

The range of possible proportions of heads is divided into intervals, called bins. As we are flipping a coin 10 times, the proportion of heads ranges from 0 to 1 in increments of 0.1. Each interval represents a possible outcome of the experiment.

For each interval, we counted how many times the proportion of heads occurred in our experiment. This count represents the frequency of occurrence for that proportion.

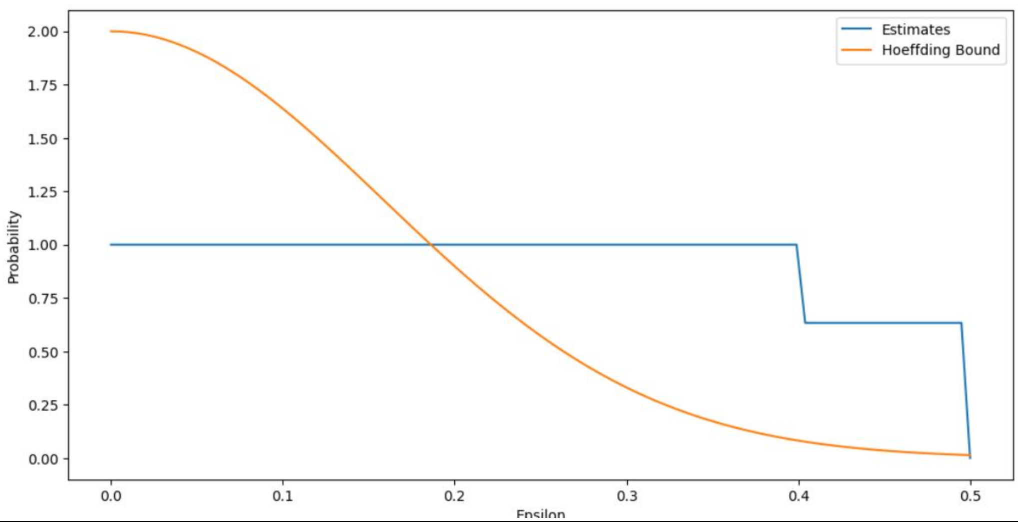
The histogram plot consists of bars, where the height of each bar represents the frequency of occurrence of the corresponding proportion of heads. The x-axis represents the range of possible proportions, while the y-axis represents the frequency of occurrence.

By looking at the histogram we can assess the distribution of the observed proportion of heads(observed empirical risk) for random nu_r is more concentrated at 0.5 and even for the first sample it is showing at 0.5. The proportions are more common at 0.5. Hence the distribution of proportions of heads aligns with our expectations compared to the true population proportion (nu = 0.5)

Plotting the histogram for estimates of Pr(|v-u| > epsilon) as a function of epsilon.

We calculated the estimates of Pr(|v-u| > epsilon) as a function of epsilon and plotted them together with the Hoeffding bound. It uses the coin_experiment function to generate the nu_min values and checks if they exceed epsilon to update the estimates	(code below)

```python
def calculate_hoeffding_bound(epsilon, num_flips):
    return 2 * np.exp(-2 * epsilon**2 * num_flips)

def plot_hoeffding_bound(num_repeats):
    epsilons = np.linspace(0, 0.5, 100)
    hoeffding_bounds = calculate_hoeffding_bound(epsilons, num_flips=10)

    nu_values = np.zeros((num_repeats))
    hoeffding_estimates = np.zeros_like(epsilons)

    for i in range(num_repeats):
        nu1, nu_r, nu_min = coin_experiment()
        nu_values[i] = nu_min

        for j, epsilon in enumerate(epsilons):
            if abs(nu_min - 0.5) > epsilon:
                hoeffding_estimates[j] += 1

    # Normalize the estimates
    hoeffding_estimates /= num_repeats

    # Plot the estimates and the Hoeffding bound
    plt.figure(figsize=(12, 6))
    plt.plot(epsilons, hoeffding_estimates, label='Estimates')
    plt.plot(epsilons, hoeffding_bounds, label='Hoeffding Bound')
    plt.xlabel('Epsilon')
    plt.ylabel('Probability')
    plt.legend()
    plt.show()

plot_hoeffding_bound(10000)
```



Observations from the histogram2:

The observed proportion of heads (v) and the true proportion of heads (u = 0.5), the histogram is plotted for Pr(|v-u| >e) as a function of e.

The range of possible deviations (e) is divided into intervals that range from 0 to 0.5 into smaller intervals, each representing a different magnitude of deviation.

The observed proportion of heads for a coin (v) falling within the Hoeffding bound (as most concentration in the histogram is on 0.5), it means that deviation from the true population (u) is within the acceptable range. In other words, these coins obey the Hoeffding bound, and the observed proportions of heads are consistent with what would be expected (u = 0.5) based on random chance.

Also, for more samples, the whole epsilon term decreases therefore the probability also decreases.

Problem 3:

G. John Stephen
jgutam@gmu.edu

Problem 3

(a) If t is a non-negative random variable, for any $\alpha > 0$, $P_r(t \geq \alpha) \leq E[t]/\alpha$.

Proof:

$$P_r(t \geq \alpha) = \int_{\alpha}^{\infty} f(t)\, dt \quad \text{(where } f(t) \text{ is the probability density function of } t) \longrightarrow eq\ 1$$

Now, condition the expectation on the event $t \geq \alpha$:

$$E[t] = \int_0^{\infty} t \cdot f(t)\, dt = \int_{\alpha}^{\infty} t \cdot f(t)\, dt + \int_0^{\alpha} t \cdot f(t)\, dt.$$

since t is non negative:

$$E[t] \geq \int_{\alpha}^{\infty} t \cdot f(t)\, dt \geq \alpha \int_0^{\alpha} f(t)\, dt \quad (\text{From } eq\ 1)$$

$$E[t] \geq \alpha\, P_r(t \geq \alpha)$$

Divide by α.

$$\boxed{P_r(t \geq \alpha) \leq \frac{E[t]}{\alpha}} \longrightarrow eq\ 2$$

(b) If u is any random variable with mean μ and variance σ^2, and $\alpha > 0$, then $P_r((u-\mu)^2 \geq \alpha) \leq \frac{\sigma^2}{\alpha}$

Proof:

Apply the non-negative random variables $(u-\mu)^2$ with α

$$P_r((u-\mu)^2 \geq \alpha) \leq \frac{E[(u-\mu)^2]}{\alpha} \quad (\because eq\ 2,\ t = (u-\mu)^2)$$

Now, note that $E[(u-\mu)^2] = Var(u) = \sigma^2$

$$\boxed{\therefore P_r((u-\mu)^2 \geq \alpha) \leq \frac{\sigma^2}{\alpha}}$$

Part (c):

If $u_1, \ldots, u_n$ are independent and identically distributed, each with mean μ and variance σ^2, and u is the sample mean of $u_1, \ldots, u_n$, then for $\alpha > 0$: $Pr((u-\mu)^2 \geq \alpha) \leq \frac{\sigma^2}{n\alpha}$

Proof:

Apply part (b) to the random variable u with α and note that $Var(u) = \frac{\sigma^2}{n}$ for n iid means μ and variance σ^2.

$$Pr[(u-\mu)^2 \geq \alpha) \leq \frac{Var(u)}{\alpha} = \frac{\sigma^2}{n\alpha}$$

Problem 4:

4) Problem 4:

Part 1

The perception prediction is given by $h(x) = \text{Sign}(w \cdot x)$ where $w = (w_0, w_1, w_2)$ and $x = \underbrace{(1, x_1, x_2)}_{\text{homogeneous}}$. The regions on the plane where $h(x) = +1$ and $h(x) = -1$ are separated by line.

If we set $h(x) = 0$, we can find the equation of the decision boundary.

$$h(x) = \text{Sign}(w \cdot x)$$

$$0 = w \cdot x$$

$$0 = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2$$

solving for $x_2 = -\frac{w_0}{w_2} - \frac{w_1}{w_2} x_1$, (It is in the form of a line, $ax + b = y$)

(we compare the coefficients.)

$$a = -\frac{w_1}{w_2}, \quad b = \frac{-w_0}{w_2}$$

So, the values of a and b in terms of the elements of w are $a = \frac{-w_1}{w_2}$ and $b = \frac{-w_0}{w_2}$

Part (b)

The plot will show two lines, one for each weight vector. The lines separate the regions where h(x) = +1 and h(x) = -1. The lines have opposite slopes and intercepts, illustrating the difference between the two weight vectors of their signs and magnitude.

jupyter jgutam_hw1_problem2_problem4 Last Checkpoint: an hour ago (unsaved changes)

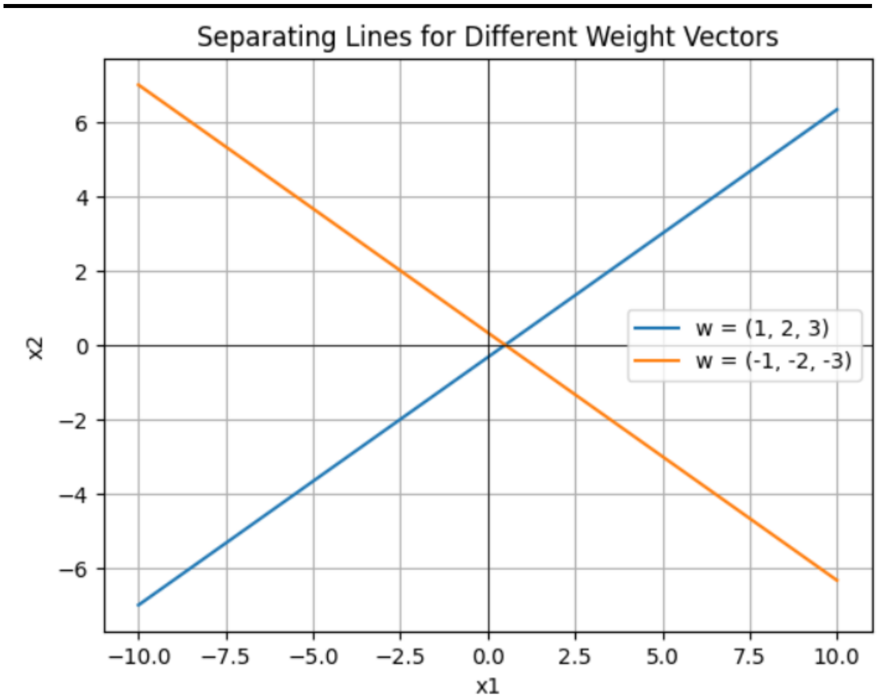
File Edit View Insert Cell Kernel Help

Log10(Difference between bound and iterations)

Problem 4

In [2]:

```python
import matplotlib.pyplot as plt
import numpy as np

# Weight vector w = (1, 2, 3)
w1 = 2/3
w2 = -1/3
x1_vals = np.linspace(-10, 10, 100)
x2_vals = w1*x1_vals + w2

plt.plot(x1_vals, x2_vals, label='w = (1, 2, 3)')

# Weight vector w = (-1, -2, -3)
w1 = -2/3
w2 = 1/3
x2_vals = w1*x1_vals + w2

plt.plot(x1_vals, x2_vals, label='w = (-1, -2, -3)')

plt.xlabel('x1')
plt.ylabel('x2')
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()
plt.grid(True)
plt.title('Separating Lines for Different Weight Vectors')
plt.show()
```



Problem 5:

G. John Stephen.

(5) Problem 5

To derive the estimators μ and σ^2 that maximize the log-likelihood function for a given distribution, we will find the maximum by taking partial derivatives with respect to μ and σ^2, setting them to zero, and solving for the parameters.

Estimating μ:

$$\ln p(x|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 - \frac{n}{2}\ln\sigma^2 - \frac{n}{2}\ln(2\pi)$$

partial derivative w.r.t μ

$$\frac{\partial}{\partial\mu}\ln p(x|x,\sigma^2) = \frac{-1}{\cancel{2}\sigma^2}\times\cancel{2}\sum_{i=1}^{n}(x_i-\mu) - 0 - 0 = 0$$

$$= \frac{-1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu) = 0$$

$$\Rightarrow \sum_{i=1}^{n}(x_i-\mu) = 0$$

$$\sum_{i=1}^{n}x_i - n\mu = 0 \Rightarrow \mu = \frac{1}{n}\sum_{i=1}^{n}x_i$$

So the maximum likelihood estimator for μ is the sample men

$$\boxed{\mu = \frac{1}{n}\sum_{i=1}^{n}x_i.}$$

Estimating σ^2:

$$\ln p(x|\mu,\sigma^2) = \frac{-1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 - \frac{n}{2}\ln\sigma^2 - \frac{n}{2}\ln(2\pi)$$

partial derivative w.r.t σ^2

$$\frac{\partial}{\partial\sigma^2}\ln p(x|\mu,\sigma^2) = -\frac{1}{2}(-2)\sigma^{-3}\sum_{i=1}^{n}(x_i-\mu)^2 - \frac{n}{2}\frac{1}{\sigma^2}(2\sigma) - 0$$

$$-\frac{1}{\cancel{2}}(-\cancel{2})\,\sigma^{-3}\sum_{i=1}^{n}(x_i-\mu)^2-\frac{n}{\cancel{2}}\,\frac{1}{\sigma^{\cancel{2}}}(\cancel{2}\cancel{\sigma})=0$$

$$\frac{1}{\sigma^{\cancel{3}\,2}}\sum_{i=1}^{n}(x_i-\mu)^2=\frac{n}{\cancel{\sigma}}$$

$$\frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2=n$$

$$\boxed{\therefore\quad \sigma^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2}$$

Problem 6:

6. Problem 6

The likelihood function L measures the likelihood of observing the given data $\{(x_i, y_i)\}_{i=1}^{n}$ given the parameters θ of the model.

we model the probability as $P(y=+1 \mid x)$ using a function $h(x)$

The likelihood function for a single data point (x_i, y_i) is given by:

$$L(\theta) = \begin{cases} h(x_i) & \text{if } y_i = +1 \\ 1-h(x_i) & \text{if } y_i = -1 \end{cases}$$

$$\ell(\theta) = \sum_{i=1}^{n} \left[1[y_i = +1] \log(h(x_i)) + 1[y_i = -1] \log(1-h(x_i)) \right]$$

where $1[A]$ is the indicator function. $\sigma(z)$ is the sigmoid function.

given $\quad h(x) = \sigma(w \cdot x)$, where

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

substitute

$$\ell(\theta) = \sum_{i=1}^{n} \left[1[y_i = +1] \log(\sigma(w \cdot x_i)) + 1[y_i = -1] \log(1-\sigma(w \cdot x_i)) \right]$$

$$= \sum_{i=1}^{n} \left[1[y_i = +1] \log\left(\frac{1}{1+e^{-w \cdot x_i}}\right) + 1[y_i = -1] \log\left(1 - \frac{1}{1+e^{-w \cdot x_i}}\right) \right]$$

$$= \sum_{i=1}^{n} \left[1[y_i = +1] \log\left(\frac{1}{1+e^{-w \cdot x_i}}\right) + 1[y_i = -1] \log\left(\frac{e^{-w \cdot x_i}}{1+e^{-w \cdot x_i}}\right) \right]$$

$$= \sum_{i=1}^{n} \left[1[y_i = +1] \log\left(\frac{1}{1+e^{-w \cdot x_i}}\right) + 1[y_i = -1] \log\left(\frac{1}{1+e^{w \cdot x_i}}\right) \right]$$

$$= \sum_{i=1}^{n} \left[1[y_i = +1] \log\left(\frac{1}{1+e^{-w \cdot x_i}}\right) - 1[y_i = -1] \log\left(1+e^{w \cdot x_i}\right) \right)$$

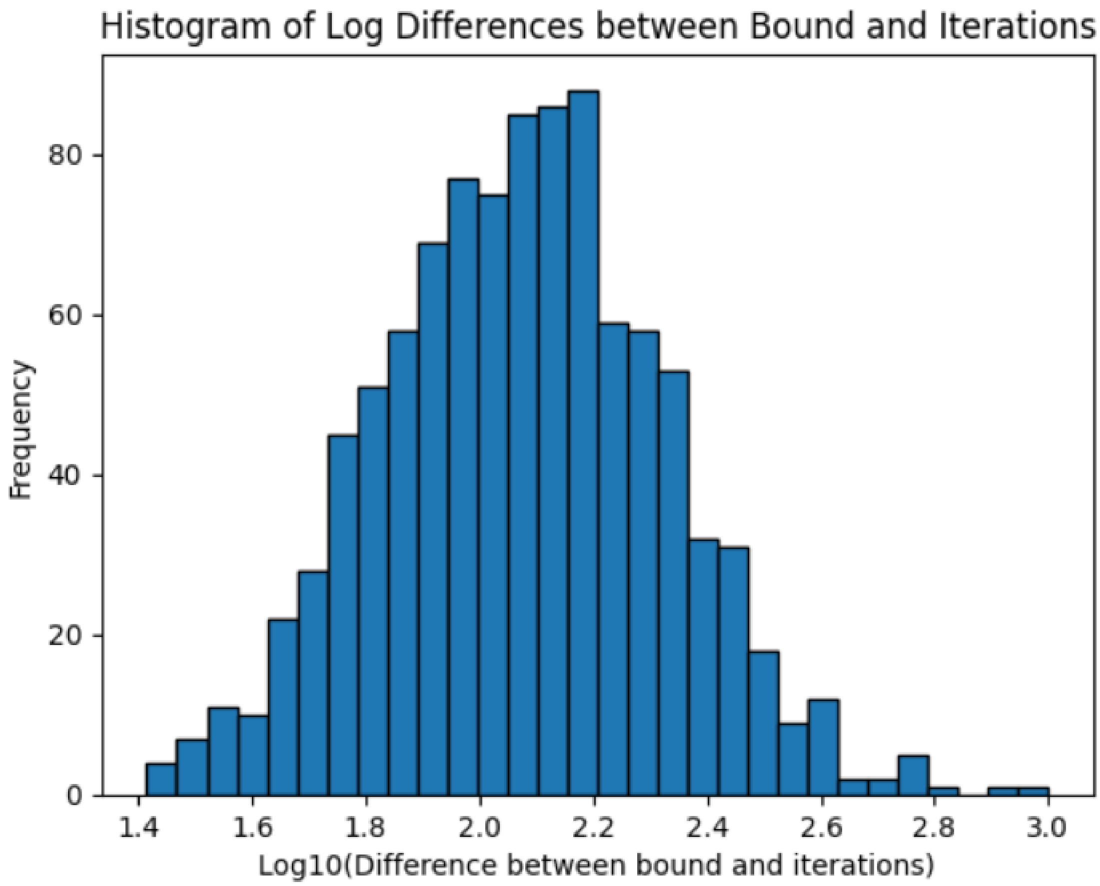
This expression represents the minimized when $h(x) = \sigma(w \cdot x)$.

Problem 2:

1. Generated an 11-dimensional weight vector w∗, where the first dimension is 0 and the other 10 dimensions are sampled independently at random from the uniform (0, 1) distribution.
2. Repeated the steps 1000 times:

   a. Generated a random training set with 100 examples, where each dimension of each training example is sampled independently at random from the uniform (-1, 1) distribution, and the examples are all classified by w∗.
   b. By running the perceptron learning algorithm on the training set, starting with the zero-weight vector, and keeping track of the number of iterations it takes to learn a hypothesis that correctly separates the training data.

3. Plotted a histogram of the number of iterations the algorithm takes to learn a linear separator.
4. Calculate the difference between the number of iterations and the bound on the number of errors derived in class for each experiment.
5. Plot a histogram of the logarithm of this difference.



The code generated a new weight vector and training set for each experiment. By running the perceptron learning algorithm on the training set and keeping track of the number of iterations required to learn a hypothesis that correctly separates the training data. It then plots a histogram of the number of iterations and calculates the difference between the number of iterations and the bound on the number of errors derived in class. Finally, plotted a histogram of the logarithm of this difference.
By analyzing the histograms, we examined the distribution of the number of iterations and the difference between the iterations and the bound. This analysis shows how the perceptron learning algorithm performs in different scenarios where the algorithm takes more iterations to correctly learn a hypothesis that separates the training data. The algorithm took more iterations between 2.0 and 2.2 to correctly separate the training data.

Note that the results may vary between different runs of the experiment due to the randomness involved in generating the weight vector and training set. Therefore, to run the experiment multiple times and analyzed the distribution of the results to draw meaningful conclusions.