

# misalignment

February 27, 2025

## 1 Simulation on the Misaligned Setting

### 1.1 List of Experiments

This set of experiment simulate the cases where the human and algorithm are misaligned:

- **Experiment 1:** Complementarity of human and algorithm with **Equal Accuracy** and **Single best**
- **Experiment 2:** Aligned human and misaligned human with **Equal Accuracy** and **Single best**
- **Experiment 3:** Comparison between type-i and type-(i+1)
- **Experiment 4:** Complementarity of human and algorithm with **Different Accuracy** and **Single best**
- **Experiment 5:** Extension of our framework

Denote the ground truth ranking of the algorithm and the human by  $\pi_a^*$  and  $\pi_h^*$ .

```
[77]: import numpy as np
import matplotlib.pyplot as plt
from model.mallows import Mallows
from model.rum import SuperStar, gRUMBel
from human_ai import HumanAI
```

### 1.2 Experiment 1 (Complementarity under Misaligned and Equal-Acc Setting)

There exist cases where it is possible to achieve complementarity with **misaligned** and **equally accurate** algorithm (potential example?)

#### 1.2.1 1.1. Mallows Model

**Answer:** Yes, the following is an example, where

- $\pi_a^* = [x_1, x_2, x_3, \dots, x_{10}]$ ,
- $\pi_h^* = [x_2, x_1, x_{10}, \dots, x_3]$
- $0 < \phi_a = \phi_h \leq 0.3$

```
[18]: m = 10
k = 2
slight_misaligned_pi_h_star = [2, 1] + [i for i in range(m, 2, -1)]

## Accuracy from 0 to 1
```

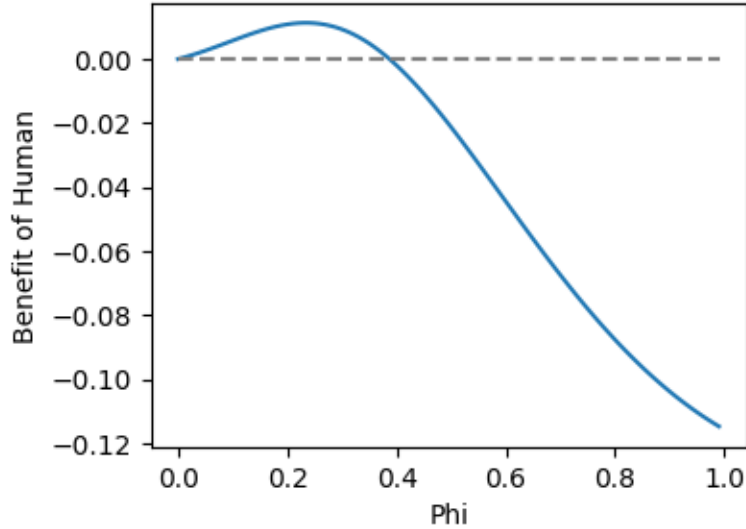
```

phis, benefits = [], []
for phi in np.arange(0, 1, 0.01):
    D_a = Mallows(m, phi, list(range(1, m+1)))
    D_h = Mallows(m, phi, slight_misaligned_pi_h_star)
    joint_system = HumanAI(m, D_a, D_h)

    benefit = joint_system.benefit_of_human_single_best(k)
    phis.append(phi)
    benefits.append(benefit)

plt.figure(figsize=(4, 3))
plt.plot(phis, benefits)
plt.plot(phis, [0 for _ in range(len(phis))], linestyle='--', color='gray')
plt.xlabel('Phi')
plt.ylabel('Benefit of Human')
plt.show()

```



### 1.2.2 1.2. gRUMbel Model (Superstar Setting)

However, it may **not** hold in the gRUMbel model.

Consider the algorithm and the human having different super stars. For example, in the following experiment,

- the superstar of the algorithm is  $x_1$  while the superstar of the human is  $x_2$ .
- $m = 10$  and  $k = 2$ .

The following results show collaboration will always cause the human make worse decision.

```

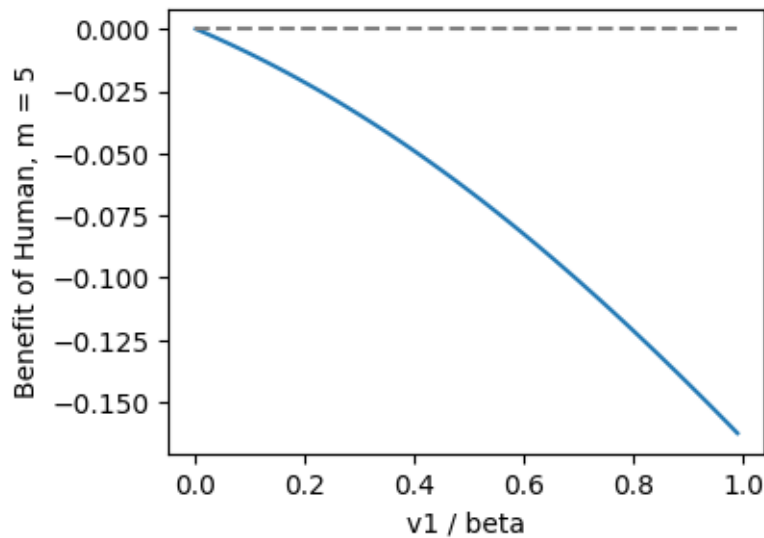
[49]: m = 10
      k = 2
      algorithm_super_star = 1
      human_super_star = 2

      ## v1 / beta, benefits
      v1_betas, benefits = [], []
      for ratio in np.arange(0, 1, 0.01):
          D_a = gRUMBel(m, algorithm_super_star, ratio, 1) #
          D_h = gRUMBel(m, human_super_star, ratio, 1)
          joint_system = HumanAI(m, D_a, D_h)

          benefit = joint_system.benefit_of_human_single_best(k)
          v1_betas.append(ratio)
          benefits.append(benefit)

      plt.figure(figsize=(4, 3))
      plt.plot(v1_betas, benefits)
      plt.plot(v1_betas, [0 for _ in range(len(v1_betas))], linestyle='--',
               color='gray')
      plt.xlabel('v1 / beta')
      plt.ylabel(f'Benefit of Human, m = {m}')
      plt.show()

```



### 1.3 Experiment 2: Aligned Human v.s. Misaligned Human

**Question:** Does there exist cases where misalignment may be helpful for accuracy. \* e.g.: example where a misaligned human achieves complementarity, an aligned human doesn't.

**Answer:** Yes, consider the following case, where

- The algorithm and the human have the same accuracy parameter  $\phi$
- The misaligned human receives better benefit in some cases.

```
[54]: m = 20
k = 2
aligned_pi_h_star = [i for i in range(1, m+1)]
slight_misaligned_pi_h_star = [2, 1] + [i for i in range(m, 2, -1)]

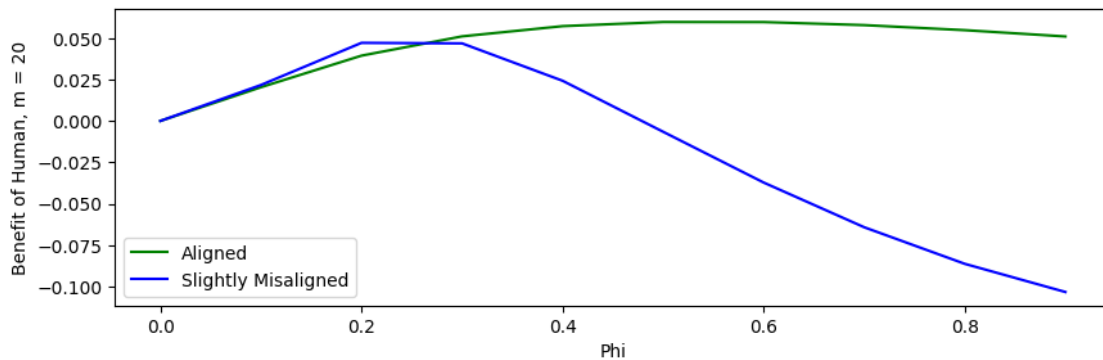
## Accuracy from 0 to 1
phis, aligned_benefits, misaligned_benefits = [], [], []
for phi in np.arange(0, 1, 0.1):
    D_a = Mallows(m, phi, list(range(1, m+1)))
    aligned_D_h = Mallows(m, phi, aligned_pi_h_star)
    misaligned_D_h = Mallows(m, phi, slight_misaligned_pi_h_star)

    aligned_joint_system = HumanAI(m, D_a, aligned_D_h)
    misaligned_joint_system = HumanAI(m, D_a, misaligned_D_h)

    aligned_benefit = aligned_joint_system.benefit_of_human_single_best(k)
    misaligned_benefit = misaligned_joint_system.benefit_of_human_single_best(k)

    phis.append(phi)
    aligned_benefits.append(aligned_benefit)
    misaligned_benefits.append(misaligned_benefit)

plt.figure(figsize=(10, 3))
plt.plot(phis, aligned_benefits, color='green', label='Aligned')
plt.plot(phis, misaligned_benefits, color='blue', label='Slightly Misaligned')
plt.xlabel('Phi')
plt.ylabel(f'Benefit of Human, m = {m}')
plt.legend()
plt.show()
```



## 1.4 Experiment 3: Comparion between Type-i and Type-(i+1)

### 1.4.1 3.1. Type-i Worst v.s Type-(i+1) Best

The following experiment suggests that, a type-i human may not always receives better benefit than a type-(i+1) human.

Consider the benefit of following types of humans, i.e.,

- $\pi_i^{least} : \{x_i, x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_m\}$
- $\pi_{i+1}^{best} : \{x_{i+1}, x_m, \dots, x_{i+2}, x_i, \dots, x_1\}$

```
[76]: m = 20
k = 2
results = []

def compare_type_i_and_i_plus_one(i, m, phi, results):
    pi_a = list(range(1, m+1))
    pi_i_least = [i] + [j for j in range(1, m+1) if j != i]
    pi_plus_one_best = [i+1] + [j for j in range(m, 0, -1) if j != i+1]

    D_a = Mallows(m, phi, pi_a)
    D_h = Mallows(m, phi, pi_i_least)
    D_h_one = Mallows(m, phi, pi_plus_one_best)

    joint_system_i = HumanAI(m, D_a, D_h)
    joint_system_i_plus_one = HumanAI(m, D_a, D_h_one)

    benefit_of_i = joint_system_i.benefit_of_human_single_best(k)
    benefit_of_i_1 = joint_system_i_plus_one.benefit_of_human_single_best(k)

    if benefit_of_i < benefit_of_i_1:
        results.append((i, phi, 'red', 'x'))
    elif benefit_of_i > benefit_of_i_1:
        results.append((i, phi, 'green', 'o'))
    else:
        results.append((i, phi, 'gray', '*'))

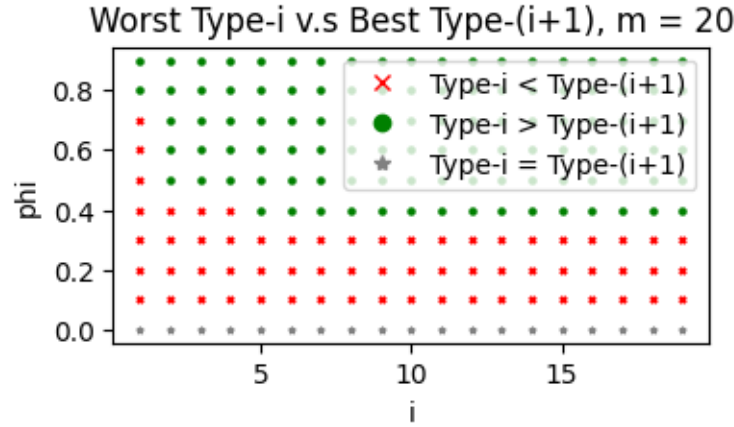
for phi in np.arange(0, 1, 0.1):
    for i in range(1, m):
        compare_type_i_and_i_plus_one(i, m, phi, results)

plt.figure(figsize=(4, 2))
for i, phi, color, marker in results:
    plt.scatter(i, phi, color=color, marker=marker, s=5)
plt.xlabel("i")
plt.ylabel("phi")
plt.title(f"Worst Type-i v.s Best Type-(i+1), m = {m}")
```

```

plt.legend(handles=[
    plt.Line2D([0], [0], color='red', marker='x', linestyle='None',
    ↪label='Type-i < Type-(i+1)'),
    plt.Line2D([0], [0], color='green', marker='o', linestyle='None',
    ↪label='Type-i > Type-(i+1)'),
    plt.Line2D([0], [0], color='gray', marker='*', linestyle='None',
    ↪label='Type-i = Type-(i+1)')
], loc=1)
plt.show()

```



## 1.5 Experiment 4: Misaligned Setting with Different Accuracy

In the above experiments, we focus on the case where the algorithm and the human have the same accuracy parameters.

So what if they can be **differently accurate**? When does complementarity hold?

In the following experiment, we vary  $\phi_a$  and  $\phi_h$  to see the benefits of the human. In particular,

- We fix  $\pi_a^* = [x_1, \dots, x_m]$ , and
- consider the following three misaligned ground-truth:
  - $\pi_h^* = [x_2, x_1, x_3, \dots, x_m]$ ;
  - $\pi_h^* = [x_2, x_1, x_m, \dots, x_3]$ ;
  - $\pi_h^* = [x_m, x_{m-1}, \dots, x_1]$ .

```

[19]: def diff_accurate_misaligned(phi_h_star, k):
    pos_phi_as, neg_phi_as = [], []
    pos_phi_hs, neg_phi_hs = [], []
    pos_colors, neg_colors = [], []

    print("phi_h_star: ", phi_h_star)
    for phi_a in np.arange(0, 1, 0.1):
        for phi_h in np.arange(0, 1, 0.1):

```

```

D_a = Mallows(m, phi_a, list(range(1, m+1)))
D_h = Mallows(m, phi_h, phi_h_star)
joint_system = HumanAI(m, D_a, D_h)

utility = joint_system.benefit_of_human_single_best(k)
intensity = abs(utility)
color = (1 - intensity, 0, 0.5) if utility > 0 else (0, 1-intensity, 0.5)

if utility > 0:
    pos_phi_as.append(phi_a)
    pos_phi_hs.append(phi_h)
    pos_colors.append(color)
else:
    neg_phi_as.append(phi_a)
    neg_phi_hs.append(phi_h)
    neg_colors.append(color)

fig = plt.figure(figsize=(3, 3))
plt.scatter(pos_phi_hs, pos_phi_as, c=pos_colors, s=20, marker="x",
label="Pos Utility")
plt.scatter(neg_phi_hs, neg_phi_as, c=neg_colors, s=20, marker="o",
label="Neg Utility")
plt.xlabel("phi_h")
plt.ylabel("phi_a")
plt.legend(loc=1)
plt.title("Utility of the human in slightly misaligned case")
plt.show()

misaligned_pi_h_star1 = [2, 1] + [i for i in range(3, m+1, 1)]
diff_accurate_misaligned(misaligned_pi_h_star1, 2)

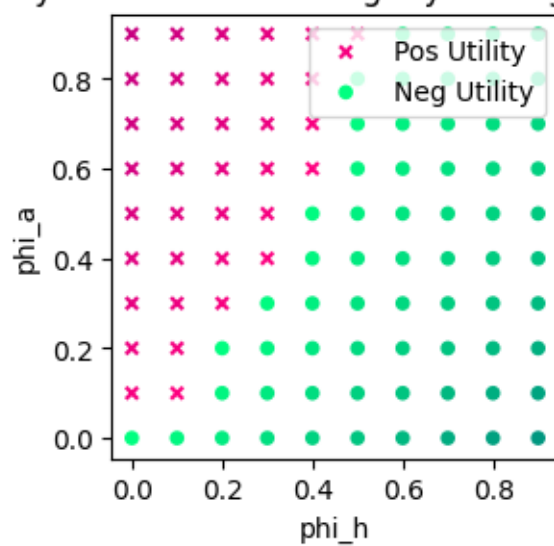
misaligned_pi_h_star2 = [2, 1] + [i for i in range(m, 2, -1)]
diff_accurate_misaligned(misaligned_pi_h_star2, 2)

misaligned_pi_h_star3 = [i for i in range(m, 0, -1)]
diff_accurate_misaligned(misaligned_pi_h_star3, 2)

```

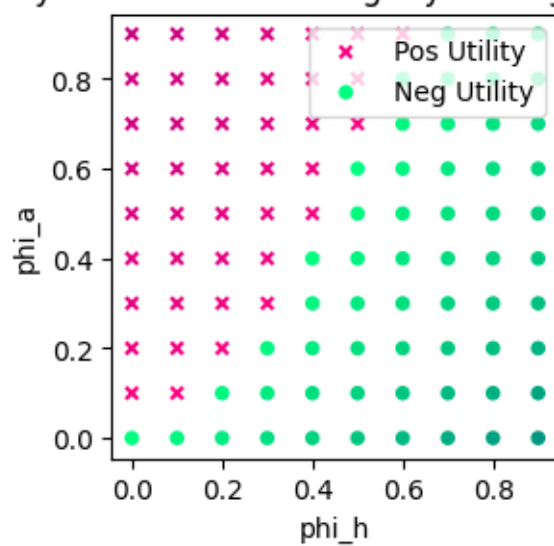
phi\_h\_star: [2, 1, 3, 4, 5, 6, 7, 8, 9, 10]

Utility of the human in slightly misaligned case



`phi_h_star:` [2, 1, 10, 9, 8, 7, 6, 5, 4, 3]

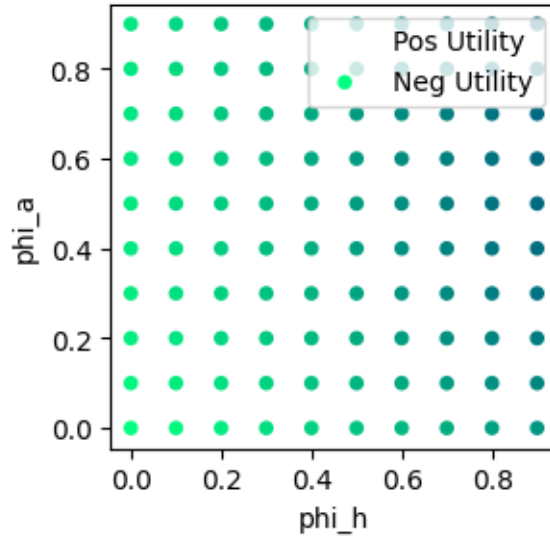
Utility of the human in slightly misaligned case



`phi_h_star:` [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]



Utility of the human in slightly misaligned case



## 1.6 Experiment 5: A Possible Comparison with Other Pluralistic Framework

Most pluralistic alignment work is implicitly studying the case where human has perfect accuracy (e.g. I know my political opinions perfectly).

This experiment aims to extend our setting to the scenario studied in other pluralistic framework.

To be more specific, we investigate the case where the human has **perfect accuracy**. However, as humans' energy and attention are limited, they still suffer from the scalability issue. For example, she can only handle a bounded-size of items or has a probability of missing some items.

Consider the following scenario:

- A professor is reviewing hundreds of PhD applications. Due to the heavy teaching and research workload, the professor has a probability of  $p_i$  of missing an application  $i$ .
- The professor decides to meet with some students that she has noticed.
- The professor has a keen eye for talent. After interviewing with the students, she is able to select the best applicant. However, among the emails she misses, some outstanding applicants may have been overlooked.
- Suppose there is an AI-tool that helps with reviewing the applicants. It can handle all the applicants in seconds. However, the tool is not very accurate.
- So will it help the professor recruit the best PhD student by using AI to screen the resumes to get a shorter list and then reviews the students in the list?

```
[9]: m = 100 # Assume there are 100 students

    ## Expected index of the recruited student
```

```

def expect_index(m, p):
    expect_index = 0
    for i in range(m):
        i_being_picked = 1
        for j in range(i):
            i_being_picked *= p[j]
        i_being_picked *= (1 - p[i])
        expect_index += i_being_picked * (i + 1)

    return expect_index

# Professor A: carefully review the resumes
# The probability of missing an applicant increases when the quality of student
↳decreases
p_prof_A = [min(0.1 + 0.1*(i+1), 1) for i in range(m)]

# Professor B: randomly review the resumes
# The probability of missing an applicant is the same
p_prof_B = [0.9 for _ in range(m)]

E_id_A = expect_index(m, p_prof_A)
E_id_B = expect_index(m, p_prof_B)
print("Expected index of the recruited student for Professor A: {}".
      ↳format(E_id_A))
print("Expected index of the recruited student for Professor B: {}".
      ↳format(E_id_B))

## Estimate the expected benefit with AI assistance
def expect_index_with_AI_assistance(k, phi_a, verbose=False):
    D_AI = Mallows(m, phi_a, list(range(1, m+1)))
    number_of_iter = 100
    expected_index_with_AI_assistance = 0

    for _ in range(number_of_iter):
        sampled_perm = D_AI.sample()
        short_list = sampled_perm[:k]
        if verbose:
            print("short list: ", short_list)
        expected_index_with_AI_assistance += min(short_list)
    expected_index_with_AI_assistance /= number_of_iter

    return expected_index_with_AI_assistance

## An highly accurate AI assistant
phi_a = 1
E_AI = expect_index_with_AI_assistance(10, phi_a)
print("Expected index with highly-accurate AI assistance: {}".format(E_AI))

```

```

## An mediumly accurate AI assistant
phi_a = 0.2
E_AI = expect_index_with_AI_assistance(10, phi_a)
print("Expected index with mediumly-accurate AI assistance: {}".format(E_AI))

## An zero accurate AI assistant
phi_a = 0
E_AI = expect_index_with_AI_assistance(10, phi_a)
print("Expected index with zero-accurate AI assistance: {}".format(E_AI))

```

Expected index of the recruited student for Professor A: 1.28324160000000002  
 Expected index of the recruited student for Professor B: 9.997078246122367  
 Expected index with highly-accurate AI assistance: 1.0  
 Expected index with mediumly-accurate AI assistance: 1.11  
 Expected index with zero-accurate AI assistance: 10.63

## 1.7 Experiment 6: Misaligned Setting Beyond Single Best Item

(Ongoing...)

```

[9]: import itertools
      print(list(itertools.permutations([1, 2, 3])))

      pi_h_stars = list()

```

[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]