# 引き分けに対する報酬の追加

　本章では、Q-learningを用いた三目並べAIのより良いモデルの構築を「引き分け」状態に対する報酬を追加することで検証する。

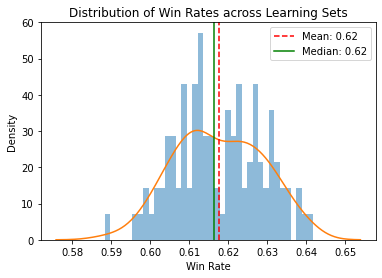
　プログラムは下記の通りであり、引き分け状態の報酬は「reward==○○」の○○に報酬値を入力することで決定する。報酬無し、負の報酬、正の報酬それぞれを与えることでどのようにランダムAIとの勝率が変化するかを検証する。具体的には10,000回の学習を100セット行いモデルを構築し、その後ランダムAIと10,000回戦わせた際の数値的な変化を検証する。

引き分けの学習無し

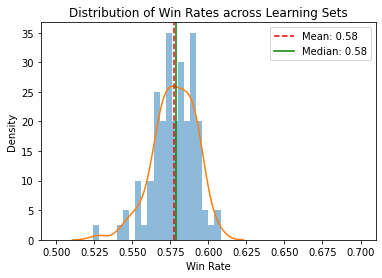
|  |
| --- |
| # ランダムAIとQ学習AIのゲームを実行する関数  def randomAI\_vs\_QLAI(first\_inputter, q\_table, epsilon=0.5):  inputter1 = 'Random AI'  inputter2 = 'Q-Learning AI'  ql\_input\_list = []  play\_area\_list = []  play\_area = list(range(1, 10))  inputter\_count = first\_inputter  end\_flg = 0  ql\_flg = 0  reward = 0  game\_count = 0 # ゲームの回数をカウントするための変数を追加します  while True:  # Q学習退避用  play\_area\_tmp = play\_area.copy()  play\_area\_list.append(play\_area\_tmp)  # Q学習実行フラグ  ql\_flg = 0  # Q学習AIの手番  if (inputter\_count % 2) == 0:  # Q学習AIの入力  play\_area, ql\_ai\_input = get\_ai\_input(play\_area, first\_inputter, mode=1, q\_table=q\_table, epsilon=epsilon)  winner, end\_flg = judge(play\_area, inputter2)  # Q学習退避用  ql\_input\_list.append(ql\_ai\_input)  # Q学習AIが勝利した場合  if winner == inputter2:  reward = 1  ql\_flg = 1  play\_area\_before = play\_area\_list[-1]  ql\_ai\_input\_before = ql\_input\_list[-1]  # ランダムAIの手番  elif (inputter\_count % 2) == 1:  play\_area, random\_ai\_input = get\_ai\_input(play\_area, first\_inputter+1, mode=0)  winner, end\_flg = judge(play\_area, inputter1)  # ランダムAIが勝利した場合  if winner == inputter1:  reward = -1  #ランダムAIが先手の場合の初手以外は学習  elif winner == 'Nobody':  reward = ○○ # 引き分けの場合の報酬  if inputter\_count != 1:  ql\_flg = 1  # Q学習実行  if ql\_flg == 1:  ql\_ai\_input\_before = ql\_input\_list[-1]  q\_table = q\_learning(play\_area\_before, ql\_ai\_input\_before, reward, play\_area, q\_table, end\_flg)  if end\_flg:  break  inputter\_count += 1  game\_count += 1 # ゲームの回数をインクリメントします  print('{} win!!!'.format(winner))  return winner, q\_table |

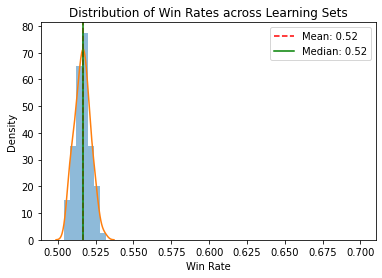
学習統計プログラム

|  |
| --- |
| import matplotlib.pyplot as plt  import seaborn as sns  import numpy as np  # 学習パラメータ  eta = 0.2 # 学習率  gamma = 0.9 # 時間割引率  initial\_epsilon = 0.5 # ε  episode = 10000 # エピソード数  # 学習セット数  learning\_sets = 100  # 各学習セットでの勝率を保存するリスト  win\_rates = []  for set\_index in range(learning\_sets):  # ランダムAI vs Q学習AI  winner\_list = []  # Q学習テーブルを初期化  q\_table = make\_q\_table()  for i in range(episode):  epsilon = initial\_epsilon \* (1 - i / episode)  winner, \_ = randomAI\_vs\_QLAI(1, q\_table, epsilon)  winner\_list.append(winner)  # 各セットでのQ-Learning AIの勝率を計算し、リストに追加  ql\_win\_rate = winner\_list.count('Q-Learning AI') / len(winner\_list)  win\_rates.append(ql\_win\_rate)  print(f"Learning Set {set\_index+1}, Q-Learning AI Win Rate: {ql\_win\_rate}")  # 勝率の平均と中央値を計算  mean\_win\_rate = np.mean(win\_rates)  median\_win\_rate = np.median(win\_rates)  # 勝率の分布をヒストグラムとして表示  plt.hist(win\_rates, bins=50, range=(0.5, 0.7), density=True, alpha=0.5)  # 勝率の分布をカーネル密度推定で表示  sns.kdeplot(win\_rates)  # 平均と中央値をプロット  plt.axvline(mean\_win\_rate, color='r', linestyle='--', label=f'Mean: {mean\_win\_rate:.2f}')  plt.axvline(median\_win\_rate, color='g', linestyle='-', label=f'Median: {median\_win\_rate:.2f}')  plt.title('Distribution of Win Rates across Learning Sets')  plt.xlabel('Win Rate')  plt.ylabel('Density')  plt.legend()  plt.show()  plt.title('Distribution of Win Rates across Learning Sets')  plt.xlabel('Win Rate')  plt.ylabel('Density')  plt.legend()  plt.show() |



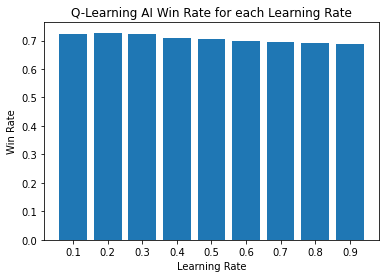
引き分け報酬あり





# 最適な学習率の探求

|  |
| --- |
| # 学習率のリスト  etas = np.arange(0.1, 1, 0.1)  win\_rate\_for\_each\_eta = []  # それぞれの学習率で学習を行います  for eta in etas:  q\_table = make\_q\_table()  gamma = 0.9  initial\_epsilon = 0.5  episode = 100000  winner\_list = []  for i in range(episode):  epsilon = initial\_epsilon \* (1 - i / episode)  winner, q\_table = randomAI\_vs\_QLAI(1, q\_table, epsilon)  winner\_list.append(winner)  # 勝率を計算し、リストに追加  win\_rate = winner\_list.count('Q-Learning AI') / len(winner\_list)  win\_rate\_for\_each\_eta.append(win\_rate)  print(f"eta={eta}, Q-Learning AI Win Rate={win\_rate}")  # 学習率ごとの勝率を棒グラフで表示します  plt.bar(etas, win\_rate\_for\_each\_eta, width=0.08)  plt.title('Q-Learning AI Win Rate for each Learning Rate')  plt.xlabel('Learning Rate')  plt.ylabel('Win Rate')  plt.xticks(etas)  plt.show() |



# 学習回数による勝率の分布の検証