Treffen mit Sebastian am Dienstag, 18.6. an der Uni Augsburg

# Ziel

* Besprechen der bisherigen Ergebnisse

🡺 Sicherstellen, dass ich es korrekt verstanden habe

* Abstecken der Ziele / Gliederung für die Masterarbeit  
  🡺 kann mit Schreiben beginnen
* Absprechen der weiteren Arbeit (Deliverables)  
  Zitationsstil: as stated in Sutton and Barto (2018). Bibliographie wie im Dokument  
  „We“ als Erzähler

# Nächste Schritte

* Recherche / Konzepte
  + Wie Neuronales Netz einbauen
  + Statistik zu Voraussetzungen für t-Test
* Implementierung
  + DP überprüfen (sollte im Mittel das Beste sein, da obere Schranke)
  + exponential smoothing überprüfen (theta\_k beeinflusst das sample v für die Berechnung von theta\_k^update)
  + Darstellung value function für single leg
  + Neuronales Netz über Value Function Approximation einbauen
* Aufschreiben
  + Theorie zu t-Test
  + Erstes Szenario single leg 100 Zeitschritte, 120 Kapazität
* Weiteres Vorgehen
  + Weitere kleine Beispiele
* Offene Fragen
  + Neuronale Netze wie Architektur modellieren, wie Zsgh beschreiben
* Eigenes hinzugefügt (nicht besprochen)
  + Code durchgehen und ausmisten, Ergebnisse rausschreiben

# Dokumente

# Erreicht

# Erkenntnisse

* CDLP Optimierung abhängig von Entscheidungsvariablen (leeres Offerset einschließen führt zu gleichem Zielfuntionswert bei unterschiedlicher Belegung der Entscheidungsvariablen)

# Ideen

* Kapazitätsbedarf von einem Produkt für eine Ressource in {0, 1, …, N} statt nur in {0, 1}
* Preise sind bisher fixiert. Könnte man hier näher an die Realität (auch Preis für First Class ändert sich über Zeit)? Wird teilweise schon abgedeckt mit den versch. Produkten.
* Zeitwert des Geldes einführen. Der Fluggesellschaft sollte lieber sein, den Euro heute zu bekommen als morgen (könnte ihn investieren)  
  => Wertfunktion anpassen

# Große Punkte

* Rote Kästen im 0618 - 0 ADP
* API linear exponential smoothing  
  Dokument: 0618 - 1 Exploring exponential smoothing (hier ohne epsilon-greedy)  
  Dokument 0618 - 2 E\_API\_lin\_single\_leg with epsilon greedy  
  exponential smoothing wie in Koch, 3.3. Update of parameters erwähnt, ist im Wesentlichen eine Durchschnittsbildung
* T-Test für Vergleich verschiedener Policies  
  Code: T\_compare\_strategies\_single\_leg  
  matched pairs t-test mit Null-Hypothese API - DP < 0 (erwarte, dass DP besser ist)
* MLP Ansatz und wie an die Pi’s kommen

# Kleine Punkte

* Sprache: A customer arrives. “He” oder “She” oder “It” buys a product.
* Arrival probability wird bei Bront et al über einen Poisson Prozess modelliert, was zu mehr als einem Customer pro Zeitschritt führen könnte. Im Erwartungswert lambda pro Zeitschritt. Ich habe implementiert mit maximal 1 customer pro Zeitschritt (Summe der Ankunftswkeiten < 1), relevant bei simulate\_sales(offer\_set).
* Algorithmus determine\_offer\_tuple (von Bront et al) anpassen, dass nur mögliche Produkte in s\_prime berücksichtigt werden (Kapazität); passt schon, da cap = 0 => pi = \infty

# Neue Punkte

* API: warum dieselben sample paths in jeder policy iteration?

# Nächstes Treffen

* Notation (zB t Index oben rechts)