Online Lernen

Algorithmische Spieltheorie

Sommer 2017

Einführendes Beispiel

In vielen Fällen trifft man Entscheidungen nicht einmal sondern wiederholt, z.B. jeden Tag, ohne zu wissen was andere Teilnehmer/Spieler oder die "Natur" an diesem Tag tun werden. Dazu betrachten wir Lernalgorithmen, die mit dieser Art von Unsicherheit umgehen können.

Beispiel:

- Du mußt jeden Tag von zu Hause zum Campus Bockenheim pendeln.
- Die Fahrzeit liegt zwischen 30 und 60 Minuten, je nachdem welches Verkehrsmittel und welche Route Du wählst und wie der Verkehr ist oder wieviel Verspätung die Busse und Züge haben.
- Stell Dir vor, Du kennst drei *Experten*, die auch aus Deiner Straße zum Unicampus pendeln und dafür unterschiedliche Wege benutzen.

Wir werden zeigen, dass Du auf die Dauer im Durchschnitt fast so schnell sein kannst wie der beste Experte, nur indem Du die Wahlen der Experten imitierst.

Das Experten-Problem

Wir betrachten ein Gegnermodel mit diskreten Zeitschritten $1, \ldots, T$.
Sei $[T] = \{1, \ldots, T\}$.

Die Experten und ihre Kosten

- Es gibt N *Experten*, nummeriert von 1 bis N.
- Im Schritt $t \in [T]$ erfährt Experte $i \in [N]$ die *Kosten* $\ell_i^t \in [0, 1]$, die von einem unbekannten Gegner gewählt werden (z.B. der "Natur")
- Sei $L_i^t = \sum_{k=1}^{t} \ell_i^k$.

Kombination von Experten

- In Schritt t wählt ein *Online-Algorithmus* H den Experten $i \in [N]$ mit Wahrscheinlichkeit p_i^t.
- Der Vektor p^t kann von den vorherigen Kosten $\ell^1, \ldots, \ell^{t-1}$ abhängen.
- Der (erwartete) Verlust von H in Schritt t ist $\ell_H^t = \sum_{i \in [N]} p_i^t \ell_i^t$.
- Sei $L_H^t = \sum_{k=1}^{t} \ell_H^k$.

Ein Greedy-Algorithmus

Im Folgenden sei $L_{min}^{t-1} = \min_{i \in [N]} L_i^{t-1}$ für $1 \leq t \leq T$.

Greedy Algorithmus

In jedem Schritt t,

- sei $S^{t-1} = \{i : L_i^{t-1} = L_{min}^{t-1}\}$;
- sei $j = \min\{S^{t-1}\}$;
- setze $p_j^t = 1$, und $p_i^t = 0$, für $i \neq j$.

Für die Analyse des Greedy-Algorithmus nehmen wir der Einfachheit halber an, dass alle Verluste entweder 0 oder 1 sind (anstatt reelle Zahlen in $[0, 1]$).

Greedy Algorithmus

Beispiel:

ℓ_1	1	0	0	1	0	0	1	0	0	1	0	0	1	0
L_1	1	1	1	2	2	2	3	3	3	4	4	4	5	5
ℓ_2	0	1	0	0	1	0	0	1	0	0	1	0	0	1
L_2	0	1	1	1	2	2	2	3	3	3	4	4	4	5
ℓ_3	0	0	1	0	0	1	0	0	1	0	0	1	0	0
L_3	0	0	1	1	1	2	2	2	3	3	3	4	4	4
j	1	2	3	1	2	3	1	2	3	1	2	3	1	2
ℓ_{Greedy}	1	1	1	1	1	1	1	1	1	1	1	1	1	1
L_G	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Greedy Algorithmus

Satz

Der Greedy Algorithmus garantiert für jede Folge von Verlusten aus $\{0,1\}$

$$L_G^T \leq N \cdot L_{min}^T + (N-1).$$

Beweis:

- Teile die Zeit in Phasen $0, \ldots, L_{min}^T$ ein, so dass gilt:

  Jeder Schritt t mit $L_{min}^{t-1} = i$ gehört zu Phase i.

- In jeder Phase $i < L_{min}^T$ erfährt der Greedy-Algorithmus einen Verlust von höchstens N.
- In Phase L_{min}^T ist der Verlust von Greedy höchstens $N-1$.

□

Untere Schranke für deterministische Algorithmen

Satz

Für jeden deterministischen Online-Algorithmus D und jedes $T \geq 1$ gibt es eine Folge von T Verlusten, so dass

$$L_D^T = T \quad \textit{und} \quad L_{min}^T \leq \lfloor T/N \rfloor.$$

Diese untere Schranke kann man leicht zeigen, indem man das obige Beispiel für Greedy verallgemeinert. (Wie?)

Die untere Schranke zeigt, dass man ohne Randomisierung die Garantie des Greedy Algorithmus nicht schlagen kann.

Random-Greedy Algorithmus

Randomisierter Greedy Algorithmus (RG)

In jedem Schritt t,

- sei $S^{t-1} = \{i : L_i^{t-1} = L_{min}^{t-1}\}$;
- setze $p_i^t = 1/|S^{t-1}|$ für jedes $i \in S^{t-1}$, und sonst $p_i^t = 0$.

In Worten: In jedem Schritt t wählt RG einen Experten aus der Menge S^{t-1} unabhängig und uniform zufällig.

Der Random-Greedy Algorithmus

Satz

Der RG Algorithmus garantiert für jede Folge von Verlusten aus $\{0,1\}$

$$L_{RG}^T \leq (1 + \ln N) L_{min}^T + \ln N.$$

Beweis:

- Betrachte Phase $i < L_{min}^T$ (Phase definiert wie oben).
- Sei u der erste Schritt und v der letzte Schritt dieser Phase.
- In Schritt $t \in \{u, \ldots, v\}$ wählt RG einen Experten aus S^{t-1} (unabhängig und uniform zufällig)
- Es gilt $[N] \supseteq S^{u-1} \supseteq S^u \supseteq \ldots \supseteq S^{v-1} \supsetneq \emptyset$.
- Für eine einfachere Notation setzen wir $S^v = \emptyset$.

Der Random-Greedy Algorithmus

Der Verlust von RG in der betrachteten Phase beträgt

$$\sum_{t=u}^{v} \frac{|S^{t-1}| - |S^t|}{|S^{t-1}|} \ ,$$

da $S^{t-1} \setminus S^t$ die Experten aus S^{t-1} mit Verlust 1 in Schritt t sind, und die Experten in S^t einen Verlust 0 erfahren.

Dieser Ausdruck wird maximiert wenn $|S^{t-1} \setminus S^t| = 1$ für $u \leq t \leq v$. Dann ist die Summe

$$\sum_{t=1}^{|S^{u-1}|} \frac{1}{t} \ \leq \ 1 + \ln |S^{u-1}| \ \leq \ 1 + \ln N \ .$$

Der Verlust von RG in Phase $i < L_{min}^T$ ist höchstens $1 + \ln N$.

Genauso ist der Verlust von RG in Phase L_{min}^T höchstens $\sum_{t=2}^{N} \frac{1}{t} \leq \ln N$.
Daraus folgt der Satz. □

Randomized Weighted Majority Algorithmus (RWM)

Sei $\eta \in (0, \frac{1}{2}]$ ein passend gewählter Parameter.

Randomized Weighted Majority (RWM) Algorithmus

Am Anfang sei $w_i^1 = 1$ für jedes $i \in [N]$.
In jedem Schritt t,

- sei $W^t = \sum_{i=1}^{N} w_i^t$;
- wähle Experten i mit Wahrscheinlichkeit $p_i^t = w_i^t / W^t$;
- setze $w_i^{t+1} = w_i^t \cdot (1 - \eta)^{\ell_i^t}$.

Randomized Weighted Majority (RWM) Algorithmus

Satz (Littlestone, Warmuth, 1994)

Der RWM Algorithmus garantiert für jede Folge von Verlusten aus $[0,1]$

$$L_{RWM}^T \leq (1+\eta) L_{min}^T + \frac{\ln N}{\eta} \ .$$

Wenn wir $\eta = \sqrt{\frac{\ln N}{T}}$ wählen, dann ergibt sich

$$L_{RWM}^T \leq L_{min}^T + 2\sqrt{T \ln N} \ .$$

Randomized Weighted Majority (RWM) Algorithmus

Der *Regret* eines Lernalgoithmus H wird definiert als $R(T) = L_H^T - L_{min}^T$.

Korollar

Der RWM Algorithmus mit $\eta = \sqrt{\frac{\ln N}{T}}$ *hat einen Regret von höchstens* $2\sqrt{T \ln N}$.

Der *durchschnittliche Regret pro Zeitschritt* ist daher nur $2\sqrt{\frac{\ln N}{T}}$.

Beachte: Dieser Term geht gegen 0 für wachsendes T.

Algorithmen mit dieser Eigenschaft heißen

No-Regret Lernalgorithmen.

Im Gegensatz zum einfachen Greedy ist RWM ein No-Regret Algorithmus.

Randomized Weighted Majority (RWM) Algorithmus

Beweis des Satzes:

- Wir analysieren, wie sich die Summe der Gewichte W^t über die Zeit verringert. Es gilt

$$W^{t+1} = \sum_{i=1}^{N} w_i^{t+1} = \sum_{i=1}^{N} w_i^t (1-\eta)^{\ell_i^t} .$$

- Beachte: $(1-\eta)^\ell = (1-\ell\eta)$ in beiden Fällen $\ell = 0$ und $\ell = 1$.
- Daneben ist $(1-\eta)^\ell$ eine konvexe Funktion in ℓ.
- Für $\ell \in [0,1]$ ergibt dies $(1-\eta)^\ell \leq (1-\ell\eta)$.
- Daraus folgt:

$$W^{t+1} \leq \sum_{i=1}^{N} w_i^t (1-\ell_i^t \eta) .$$

Randomized Weighted Majority (RWM) Algorithmus

- Sei F^t der erwartete Verlust von RWM in Zeitschritt t.
- Es gilt $F^t = \sum_{i=1}^{N} \ell_i^t w_i^t / W^t$.
- Einsetzen in die Schranke für W^{t+1} ergibt

$$W^{t+1} \leq W^t - \eta F^t W^t = W^t(1 - \eta F^t) .$$

- Daraus folgt

$$W^{T+1} \leq W^1 \prod_{t=1}^{T}(1 - \eta F^t) = N \prod_{t=1}^{T}(1 - \eta F^t) .$$

- Die Summe der Gewichte nach Schritt T kann also nach oben beschränkt werden durch den erwarteten Verlust von RWM.

Randomized Weighted Majority (RWM) Algorithmus

- Andererseits kann die Summe der Gewichte nach Schritt T auch nach unten beschränkt werden durch den Verlust des besten Experten:

$$W^{T+1} \geq \max_{1 \leq i \leq N} (w_i^{T+1}) = \max_{1 \leq i \leq N} \left((1-\eta)^{\sum_{t=1}^{T} \ell_i^t} \right) = (1-\eta)^{L_{min}^T} .$$

- Kombination der beiden Schranken und Logarithmieren auf beiden Seiten führt zu

$$L_{min}^T \ln(1-\eta) \leq (\ln N) + \sum_{t=1}^{T} \ln(1-\eta F^T) .$$

- Wir vereinfachen den Ausdruck mit Hilfe der folgenden Abschätzung

$$-z - z^2 \leq \ln(1-z) \leq -z ,$$

die für jedes $z \in [0, \frac{1}{2}]$ gilt.

Randomized Weighted Majority (RWM) Algorithmus

- Diese Vereinfachung ergibt

$$\begin{aligned} L_{min}^T(-\eta - \eta^2) &\leq (\ln N) + \sum_{t=1}^{T}(-\eta F^t) \\ &= (\ln N) - \eta L_{RWM}^T \ . \end{aligned}$$

- Zum Schluß lösen wir die Ungleichung nun nur noch nach L_{RWM}^T auf:

$$L_{RWM}^T \leq (1+\eta)L_{min}^T + \frac{\ln N}{\eta}$$

□

Erlernen von Gleichgewichten in Spielen

Regret-Minimierung ist ein natürlicher Ansatz für Verhalten bei wiederholten Entscheidungen mit unvollständiger Information.

Wir betrachten Regret-Lernen in einem Spiel $\Gamma = (\mathcal{N}, (\Sigma_i)_{i \in \mathcal{N}}, (c_i)_{i \in \mathcal{N}})$, das T Runden lang wiederholt hintereinander gespielt wird. (sog. *wiederholtes Spiel*).

Anfangs kennt keiner der Spieler $i \in \mathcal{N}$ das Spiel. In jeder Runde t wählt jeder Spieler i eine reine Strategie $s_i^t \in \Sigma_i$ mit seinem eigenen No-Regret Algorithmus. Der Algorithmus von i benutzt nur die *beobachteten Kosten von i in den vorherigen Runden*.

Wird dieses System zu einem (approx.) Nash-Gleichgewicht konvergieren?

Dies wäre eine tolle und sehr plausible Erklärung, wie Nash-Gleichgewichte in der Praxis entstehen können. Leider ist im Allgemeinen die Antwort: Nein.

In einigen interessanten Fällen können wir aber mit No-Regret Algorithmen (schnell) in die Nähe eines Nash-Gleichgewichts gelangen. Mit diesen Algorithmen können die Spieler hier also Nash-Gleichgewichte "erlernen".

Erinnerung: Nullsummenspiele mit 2 Spielern

- In einem Nullsummenspiel mit 2 Spielern gilt für die beiden Spieler $c_{\mathtt{I}}(s) + c_{\mathtt{II}}(s) = 0$ in jedem Zustand s des Spiels.
- Matrix A mit $|\Sigma_{\mathtt{I}}|$ Zeilen und $|\Sigma_{\mathtt{II}}|$ Spalten.
  Spieler I ist Zeilenspieler, Spieler II ist Spaltenspieler.
- a_{ij} ist Nutzenwert von Spieler I in Zustand (i, j),
  a_{ij} ist Kostenwert oder Verlust von Spieler II in Zustand (i, j).
- Wir normalisieren A so dass $a_{ij} \in [0, 1]$:

  A wird nicht-negativ wenn wir $\max |a_{ij}|$ zu jedem Eintrag hinzuaddieren. Dann teilen wir durch den entstandenen höchsten Eintrag. Damit erhalten wir alle Einträge a_{ij} aus dem Intervall $[0, 1]$.

  Beachte: Diese Anpassung hat keine Auswirkung auf die optimalen Stragegien (und damit die Nash-Gleichgewichte) des Spiels.

Beispiele

Matching Pennies (normalisiert)

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Stein-Papier-Schere (normalisiert)

$$\begin{pmatrix} 1/2 & 0 & 1 \\ 1 & 1/2 & 0 \\ 0 & 1 & 1/2 \end{pmatrix}$$

Ein Spiel mit $|\Sigma_I| \neq |\Sigma_{II}|$:

$$\begin{pmatrix} 0 & 1/2 & 1 \\ 1/4 & 1/2 & 3/4 \end{pmatrix}$$

Maximin-Strategien

- Gain-Floor für Spieler I: $v_{\mathtt{I}}^* = \max_x \min_y x^T A y$.

  Optimale Strategie x^* garantiert Gain-Floor für I (Maximin-Strategie).

- Loss-Ceiling für Spieler II: $v_{\mathtt{II}}^* = \min_y \max_x x^T A y$.

  Optimale Strategie y^* garantiert Loss-Ceiling für II (Minimax-Strategie).

Lemma

Es gilt: $v_{\mathtt{I}}^* \leq v_{\mathtt{II}}^*$.

Satz (Minimax Theorem)

In jedem Nullsummenspiel mit 2 Spielern gilt $v = v_{\mathtt{I}}^* = v_{\mathtt{II}}^*$.

Gemischte Nash-Gleichgewichte

Korollar

Zustand (x, y) im Nullsummenspiel mit 2 Spielern ist ein gemischtes Nash-Gleichgewicht

$\Leftrightarrow$

x und y sind optimale Strategien.

Korollar

Jedes Nullsummenspiel mit 2 Spielern hat mindestens ein gemischtes Nash-Gleichgewicht, und alle gemischten Nash-Gleichgewichte ergeben den gleichen erwarteten Nutzenwert für Spieler `I`.

Satz

In Nullsummenspielen mit 2 Spielern kann ein gemischtes Nash-Gleichgewicht in polynomieller Zeit berechnet werden.

Lernen in Nullsummenspielen

- Spieler kennen das Spiel nicht, in dem sie spielen. Sie verwenden No-Regret Algorithmen für die Wahl der Strategie.

  Können die Spieler optimale Strategien und somit ein gemischtes Nash-Gleichgewicht erlernen?

Lernen in Nullsummenspielen

- Spieler kennen das Spiel nicht, in dem sie spielen. Sie verwenden No-Regret Algorithmen für die Wahl der Strategie.

  Können die Spieler optimale Strategien und somit ein gemischtes Nash-Gleichgewicht erlernen?

- Betrachte Spieler `II`. Experten sind reine Strategien, Gegner ist Spieler `I`.
- In jedem Schritt t wählt der Lernalgorithmus H von Spieler `II` eine gemischte Strategie y^t gegen die unbekannte Strategie x^t des Gegenspielers `I`.

Lernen in Nullsummenspielen

- Spieler kennen das Spiel nicht, in dem sie spielen. Sie verwenden No-Regret Algorithmen für die Wahl der Strategie.

  Können die Spieler optimale Strategien und somit ein gemischtes Nash-Gleichgewicht erlernen?

- Betrachte Spieler `II`. Experten sind reine Strategien, Gegner ist Spieler `I`.
- In jedem Schritt t wählt der Lernalgorithmus H von Spieler `II` eine gemischte Strategie y^t gegen die unbekannte Strategie x^t des Gegenspielers `I`.
- Verlust in Schritt t für Strategie (Experte) i beträgt

$$\ell_i^t = \sum_{j \in \Sigma_{\mathtt{I}}} x_j^t a_{ji} \ .$$

- Gesamter Verlust des Lernalgorithmus H in Schritt t beträgt

$$\ell_H^t = c_{\mathtt{II}}(x^t, y^t) = \sum_{i \in \Sigma_{\mathtt{II}}} \sum_{j \in \Sigma_{\mathtt{I}}} x_j^t a_{ji} y_i^t \ .$$

No-Regret und optimale Strategien

- No-Regret Lernalgorithmus H:

$$\frac{L_H^T - L_{min}^T}{T} \longrightarrow 0 \qquad \text{für } T \to \infty \ .$$

No-Regret und optimale Strategien

- No-Regret Lernalgorithmus H:

$$\frac{L_H^T - L_{min}^T}{T} \longrightarrow 0 \qquad \text{für } T \to \infty \ .$$

- Nach Definition: Der durchschnittliche Verlust pro Schritt eines No-Regret Algorithmus wird so klein wie der durchschnittliche Verlust der (im Nachhinein nach t Schritten) besten reinen Strategie.
- Wird der durchschnittliche Verlust L_H^T/T so klein wie der Wert des Spiels?

No-Regret und optimale Strategien

- No-Regret Lernalgorithmus H:

$$\frac{L_H^T - L_{min}^T}{T} \longrightarrow 0 \qquad \text{für } T \to \infty \ .$$

- Nach Definition: Der durchschnittliche Verlust pro Schritt eines No-Regret Algorithmus wird so klein wie der durchschnittliche Verlust der (im Nachhinein nach t Schritten) besten reinen Strategie.
- Wird der durchschnittliche Verlust L_H^T/T so klein wie der Wert des Spiels?

Satz

Betrachte ein wiederholtes Nullsummenspiel mit 2 Spielern. Wenn Spieler `II` *über die T Zeitschritte einen Algorithmus H mit Regret R für seine Strategiewahl nutzt, dann beträgt sein durchschnittlicher Verlust*

$$\frac{L_H^T}{T} \leq v_{\mathtt{I}}^* + \frac{R}{T} \ .$$

Dies Resultat gilt analog für Spieler `I` *und die Loss-Ceiling.*

Erlernen des Gain-Floors

Beweis:

- Wir zeigen, dass die beste Strategie am Ende der T Schritte die Gesamtkosten von höchstens $L_{min}^T \leq T \cdot v_{\mathtt{I}}^*$ hat.
- Betrachte den Ablauf der Strategiewahlen des Gegenspielers `I`, also Strategien $x^1, x^2, \ldots, x^T$. Addiere sie zu einer "durchschnittlichen Strategie"

$$\hat{x}_j = \frac{1}{T} \sum_{t=1}^{T} x_j^t \qquad \text{für jedes } j \in \Sigma_{\mathtt{I}}$$

- Gesamter Verlust L_i^T einer reinen Strategie $i \in \Sigma_{\mathtt{II}}$ bleibt gleich, wenn Spieler `I` in allen Zeitschritten immer nur $\hat{x}$ gespielt hätte:

$$L_i^t = \sum_{t=1}^{T} \sum_{j \in \Sigma_{\mathtt{I}}} x_j^t \cdot a_{ji} = \sum_{j \in \Sigma_{\mathtt{I}}} \left(\sum_{t=1}^{T} x_j^t \right) \cdot a_{ji} = T \cdot \sum_{j \in \Sigma_{\mathtt{I}}} \hat{x}_j \cdot a_{ji} \ .$$

Erlernen des Gain-Floors

- Nehmen wir also an, I spielt immer nur $\hat{x}$. Betrachte die im Nachhinein beste reine Strategie für II. Damit ergibt sich nun ein Spiel mit nur noch einem Zeitschritt.

- In diesem Einschrittspiel muss Spieler I zuerst die durchschnittliche Strategie $\hat{x}$ wählen. Dann wählt Spieler II eine (reine) beste Antwort $\hat{x}$ – also muss I zuerst wählen, dann antwortet II.
- Per Definition des Gain-Floors gibt es immer $i \in \Sigma_{\mathtt{II}}$ so dass der Nutzen von I (und der Verlust von II) höchstens $v_{\mathtt{I}}^*$ beträgt, d.h., $c_{\mathtt{II}}(\hat{x}, i) \leq v_{\mathtt{I}}^*$.

- Also gibt es eine reine Strategie $i \in \Sigma_{\mathtt{II}}$ mit

$$L_{min}^T \leq L_i^T \leq T \cdot v_{\mathtt{I}}^* \ .$$

- Zusammen ergeben die beiden Einsichten:

$$L_H^T \ \leq \ L_{min}^T + R \ \leq \ T \cdot v_{\mathtt{I}}^* + R \ .$$

Ein einfacher Beweis des Minimax-Theorems

Satz (Minimax Theorem)

In jedem Nullsummenspiel mit 2 Spielern gilt $v = v_{\mathtt{I}}^* = v_{\mathtt{II}}^*$.

Beweis:

- Zum Widerspruch nehmen wir an $v_{\mathtt{I}}^* + \gamma = v_{\mathtt{II}}^*$ für ein $\gamma > 0$.
- Beide Spieler spielen nun das Spiel wiederholt mit einem Lernalgorithmus. Sei T groß genug, so dass der Regret der Algorithmen $R/T < \gamma/3$.
- Über die Gesamtzeit der T Runden können wir wie oben zeigen $L_{min}^T \leq v_{\mathtt{I}}^*$ für Spieler II, und $L_{min}^T \leq -v_{\mathtt{II}}^*$ für Spieler I ("$-$" da Verlust).
- Dies bedeutet, dass die Algorithmen höchstens $v_{\mathtt{I}} + \gamma/3$ durchschnittliche Kosten für Spieler II und mindestens $v_{\mathtt{II}} - \gamma/3$ durchschnittlichen Nutzen für Spieler I garantieren.
- Durchschnittliche Kosten für II sind durchschnittlicher Nutzen für I $\rightarrow$ Widerspruch. □

Konvergenz

Korollar

Wenn beide Spieler einen No-Regret Lernalgorithmus nutzen, konvergiert der durchschnittliche Ablauf des Spiels $(\hat{x}, \hat{y})$ zu optimalen Strategien und damit zu einem gemischten Nash-Gleichgewicht des Spiels.

Der Satz garantiert Konvergenz nur für den durchschnittlichen Ablauf des Spiels, aber nicht für das eigentliche Verhalten in den Strategien x^t und y^t!

Satz

Es gibt No-Regret Lernalgorithmen für Spieler `I` *und* `II` *mit denen die Strategien x^t und y^t nicht zu optimalen Strategien konvergieren.*

Konvergenz für allgemeine No-Regret Algorithmen

Matching Pennies (normalisiert)

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Beweis: Ein "seltsamer" No-Regret Algorithmus H:

- Regel: `I` und `II` wählen reine Strategien, `I` wechselt zur anderen reinen Strategie in Runde 1, 3, 5, ..., `II` wechselt zur anderen reinen Strategie in Runde 2, 4, 6,...
- Wenn ein Spieler von der Regel abweicht, ruft der andere Spieler den RWM-Algorithmus auf. (Dieser Trick sichert die No-Regret-Eigenschaft für jeden möglichen Ablauf).

- $L_i^T/T \to 0.5$ für beide Strategien $i = 1, 2$ von `II`, durchschnittlicher Verlust des Algorithmus $L_H^T/T \to 0.5$. No-Regret Algorithmus für `II`! (gleiches Argument für `I`).
- Keine der Verteilungen y^t ist nahe an der optimalen Strategie $(0.5, 0.5)$, kein Verlust in einer einzelnen Runde ℓ_H^t ist nahe am Wert $v = 0.5$. □

Ein adaptiver RWM Algorithmus (Freund, Schapire, 1999)

Sei $\eta_0 \in (0, \frac{1}{2}]$ und u eine obere Schranke auf den Wert des Spiels.

Variabler Randomized Weighted Majority (vRWM) Algorithmus

Initialisiere $w_i^1 = 1$ für alle $i \in [N]$.
In jedem Schritt t,

- sei $W^t = \sum_{i=1}^{N} w_i^t$;
- wähle Experte i mit W.keit $p_i^t = w_i^t / W^t$;
- **if** $\ell_{vRWM}^t \leq u$ **then** setze $w_i^{t+1} = w_i^t$;
- **else** setze

$$\eta_t = 1 - \frac{u(1 - \ell_{vRWM}^t)}{(1-u)\ell_{vRWM}^t}$$

und $w_i^{t+1} = w_i^t \cdot (1 - \eta_t)^{\ell_i^t}$.

Vergleich von Verteilungen

Definition

Die Kullback-Leibler Divergenz oder relative Entropie von zwei Verteilungen y und y' ist gegeben durch

$$RE(y \mid\mid y') = \sum_{i=1}^{n} y_i \cdot \ln\left(\frac{y_i}{y'_i}\right) \quad .$$

Zum Vergleich von $a, b \in [0,1]$ nutzen wir die Verteilungen $(a, 1-a)$ und $(b, 1-b)$:

$$\begin{aligned} RE(a \mid\mid b) &= RE((a,1-a) \mid\mid (b,1-b)) \\ &= a \ln\left(\frac{a}{b}\right) + (1-a)\ln\left(\frac{1-a}{1-b}\right) \quad . \end{aligned}$$

Die Kullback-Leibler Divergenz für Verteilungen ist immer nicht-negativ und $RE(y \mid\mid y') = 0$ genau dann wenn $y = y'$.

Konvergenz von vRWM

Satz

Sei y' eine beliebige gemischte Strategie für `II`*, die einen Verlust von höchstens u garantiert gegen jede beste Antwort von* `I`*. In jeder Iteration t von vRWM, in der $\ell^t_{vRWM} \geq u$, sinkt die relative Entropie zwischen y' und y^{t+1} um*

$$RE(y' \mid\mid y^{t+1}) \quad \leq \quad RE(y' \mid\mid y^t) - RE(u \mid\mid \ell^t_{vRWM}) \ .$$

In jedem Schritt, in dem der Verlust von vRWM zu hoch ist, bewegt die Anpassung die neue Strategie näher an jede gute Strategie.

Beweis des Satzes

Beweis:
Zur Vollständigkeit beweisen wir den Satz. Betrachte einen Schritt t mit $\ell^t_{vRWM} > u$. Dann gilt

$$\begin{aligned} & RE(y' \mid\mid y^{t+1}) - RE(y' \mid\mid y^t) \\ = & \sum_{i \in \Sigma_{II}} y'_i \ln \frac{y'_i}{y_i^{t+1}} - \sum_{i \in \Sigma_{II}} y'_i \ln \frac{y'_i}{y_i^t} \\ = & \sum_{i \in \Sigma_{II}} y'_i \ln \frac{y_i^t}{y_i^{t+1}} \\ \leq & \sum_{i \in \Sigma_{II}} y'_i \ln \frac{1 - \eta_t \ell^t_{vRWM}}{(1-\eta_t)^{\ell_i^t}} \quad . \end{aligned}$$

Dabei nutzen wir, dass $y_i^{t+1} = w_i^t (1-\eta_t)^{\ell_i^t} / W^{t+1}$ und $W^{t+1} \leq W^t(1 - \eta_t F^t) = W^t(1 - \eta_t \ell^t_{vRWM})$ wie oben gesehen.

Beweis des Satzes

$$
\begin{aligned}
& RE(y' \mid\mid y^{t+1}) - RE(y' \mid\mid y^t) \\
\leq & \sum_{i \in \Sigma_{II}} y'_i \ln \frac{1 - \eta_t \ell^t_{vRWM}}{(1-\eta_t)^{\ell^t_i}} \\
= & \sum_{i \in \Sigma_{II}} y'_i \ln \left(\frac{1}{1-\eta_t} \right)^{\ell^t_i} + \ln(1 - \eta_t \ell^t_{vRWM}) \\
= & \left(\ln \frac{1}{1-\eta_t} \right) \cdot \sum_{i \in \Sigma_{II}} y'_i \ell^t_i + \ln(1 - \eta_t \ell^t_{vRWM}) \\
\leq & \left(\ln \frac{1}{1-\eta_t} \right) u + \ln(1 - \eta_t \ell^t_{vRWM}) \ ,
\end{aligned}
$$

weil Strategie y' nie mehr als u an Verlust erzeugt.

Beweis des Satzes

Die Ableitung von

$$\left(\ln \frac{1}{1-\eta_t}\right) u + \ln(1-\eta_t \ell^t_{vRWM})$$

nach η_t setzen wir gleich 0. Damit erhalten wir das Minimum

$$\eta_t = 1 - \frac{u(1-\ell^t_{vRWM})}{(1-u)\ell^t_{vRWM}}$$

wie gewünscht. Einsetzen in der Formel ergibt

$$\begin{aligned} & - u \ln \left(\frac{u}{\ell^t_{vRWM}} \cdot \frac{1-\ell^t_{vRWM}}{1-u} \right) + \ln \frac{1-\ell^t_{vRWM}}{1-u} \\ = & - RE(u \mid\mid \ell^t_{vRWM}) \ . \end{aligned}$$

□

Konvergenz von vRWM

Korollar

Für jede Folge von Strategien $x^1, x^2, \ldots$ beträgt die Anzahl der Runden mit Verlust $\ell^t_{vRWM} \geq u + \varepsilon$ höchstens

$$\frac{\ln |\Sigma_{II}|}{RE(u \mid\mid u + \varepsilon)} .$$

Für ein festes ε ist diese Zeit unabhängig von T. Für sp tere Zeitschritte t muss daher der Verlust immer näher an u herankommen. Dies Verhalten ist viel besser als z.B. beim seltsamen No-Regret Algorithmus, der einen Verlust von 1 in jeder zweiten Runde generiert, sogar für beliebig späte Zeitschritte t.

Bandbreitenspiel

Eine Menge $\mathcal{N}$ von n Nutzern möchten Videos über eine gemeinsamen Internetzugang herunterladen.

- Zugang hat *Kapazität* C, sei oBdA $C = 1$.
- Als Strategie platziert Spieler i ein Gebot $s_i \in \Sigma_i$, wobei $\Sigma_i = [b_{\min}, 1]$ mit $b_{\min} > 0$ als Minimalgebot.
- Der Service-Provider M sammelt den Vektor s aller Gebote und richtet eine proportionale Durchsatzrate für jeden Spieler ein:

$$M_i(s) = \frac{s_i}{\sum_{j \in \mathcal{N}} s_j}.$$

  Beachte: $M_i(s) > 0$ und $\sum_i M_i(s) = 1 = C$.
- Spieler i erhält seine Rate und zahlt sein Gebot an den Provider. Nutzen:

$$u_i(s) = \alpha_i \cdot M_i(s) - s_i \quad = \quad \frac{\alpha_i \cdot s_i}{\sum_{j \in \mathcal{N}} s_j} - s_i$$

  (Spieler i tauscht Geld gegen Rate mit Faktor $\alpha_i > 0$).

Beispiel: (sei, z.B., $b_{\min} = 0.01$)

Spieler	α_i	Gebot s_i	Rate $M_i(s)$	Nutzen $u_i(s)$
1	2	0.9	0.45	0.90 - 0.9 = 0
2	3	0.7	0.35	1.05 - 0.7 = 0.35
3	4	0.4	0.25	1.00 - 0.4 = 0.60

Funktionen $u_i(s_i, s_{-i})$ für $s = (0.9, 0.7, 0.4)$:

- $u_1(x, 0.7, 0.4) = \frac{2x}{x+1.1} - x$
- $u_2(0.9, x, 0.4) = \frac{3x}{x+1.3} - x$
- $u_1(0.9, 0.7, x) = \frac{4x}{x+1.6} - x$

Beachte: Nutzenfunktionen sind konkav.

Konkavität/Konvexität

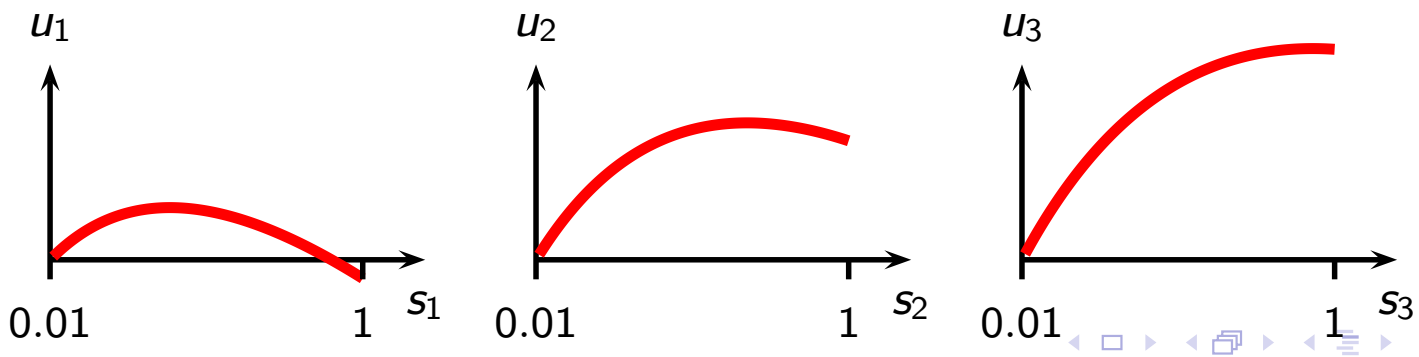
Definition (Konkav/Konvex)

Eine Funktion $f : X \to \mathbb{R}$ ist konvex (konkav) auf $X \subset \mathbb{R}^k$ wenn die direkte Verbindung zwischen $f(x)$ und $f(y)$ immer über (unter) f liegt, $\forall x, y \in X$.
Formal sei für alle $x, y \in \mathbb{R}$

$$\begin{array}{lrcll} \text{Konvex:} & \lambda f(x) + (1-\lambda) f(y) & \geq & f(\lambda x + (1-\lambda) y), & \text{für jedes } \lambda \in (0,1) \\ \text{Konkave:} & \lambda f(x) + (1-\lambda) f(y) & \leq & f(\lambda x + (1-\lambda) y), & \text{für jedes } \lambda \in (0,1) \end{array}$$

Streng konkav/konvex: $\geq$ und $\leq$ ersetzt durch $>$ und $<$.
Klar: Wenn f (streng) konkav, dann $-f$ (streng) konvex und umgekehrt.

Nutzenfunktionen im Bandbreitenspiel sind streng konkav. Im obigen Beispiel:



Existenz eines Gleichgewichts

Ein Bandbreitenspiel ist kein endliches Spiel, es gibt unendlich viele Strategien.

Gibt es immer ein Nash-Gleichgewicht?

Mit der Konkavität der Nutzenfunktionen und dem Satz von Brouwer kann man zeigen:

Lemma
Jedes Bandbreitenspiel hat mindestens ein (reines) Nash-Gleichgewicht.

Wie kann sich in diesen Spielen ein Nash-Gleichgewicht einstellen?

Konvergieren die Spieler mit No-Regret Lernalgorithmen zum Nash-Gleichgewicht?

Dafür benötigen wir zuerst No-Regret-Algorithmen für unendlich viele Experten...

Online Konvexe Minimierung

Ein Experten-Problem mit unendlich vielen Experten:

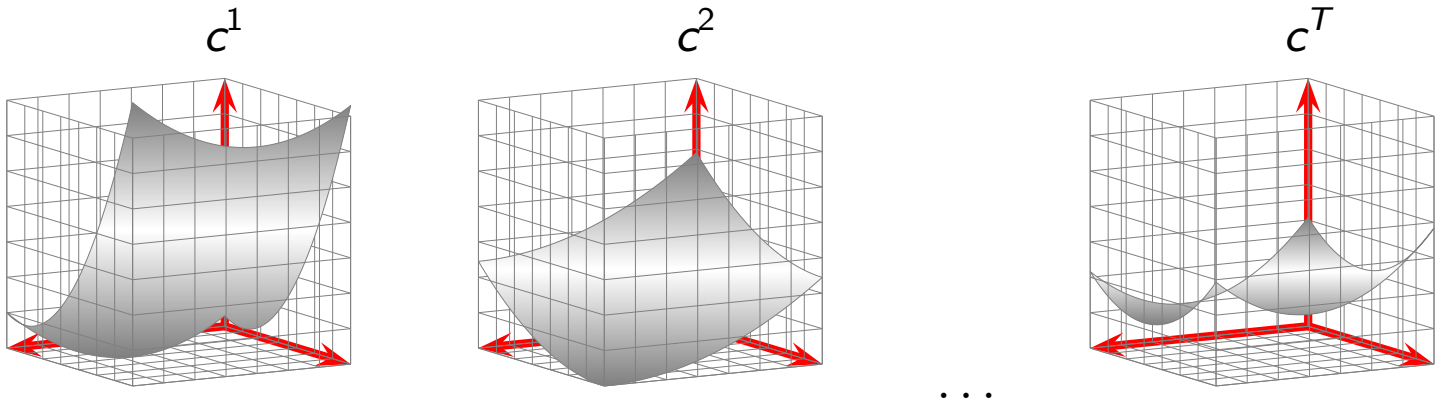
- "Experten": Alle Punkte einer konvexen und kompakten Menge $D \subset \mathbb{R}^k$
- In jedem Schritt $t = 1, \ldots, T$ wählen wir einen Punkt $x^t \in D$
- Danach wird eine differenzierbare und konvexe Kostenfunktion $c^t : D \to \mathbb{R}$ aufgedeckt.

Ziel: Wähle x^t's um die Gesamtkosten $\sum_{t=1}^{T} c^t(x^t)$ zu minimieren.

Beachte: Anstatt Minimierung von konvexen Kosten könnten wir genauso die Maximierung von konkaven Nutzenfunktionen $-c^t(x^t)$ betrachten.

Beispiele

Beispiel: Hier ist z.B. $D = [0,1]^2$, und wir sehen diverse konvexe Kostenfunktionen:



Beispiel: Im wiederholten Bandbreitenspiel

- hat Spieler i eine kompakte und konvexe Strategiemenge $D = [b_{\min}, 1]$
- Die Nutzenfunktion $u_i(s_i^t, s_{-i}^t)$ ist konkav und differenzierbar. Dazu hängt sie von unbekannten Geboten s_{-i}^t ab, die nur bekannt werden, nachdem Strategie s_i^t gewählt wurde.

No-Regret Eigenschaft

- Kosten von Spieler i: $\sum_{t=1}^{T} c^t(x^t)$
- Bester Experte $x^* \in \arg\min_{x \in D} \sum_{t=1}^{T} c^t(x)$
- Durchschnittlicher Regret pro Zeitschritt:

$$\frac{R(T)}{T} = \frac{1}{T}\left(\sum_{t=1}^{T} c^t(x^t) - \sum_{t=1}^{T} c^t(x^*)\right)$$

- No-Regret Algorithmus wenn $\frac{R(T)}{T} \to 0$ für $T \to \infty$.

Wir entwerfen einen No-Regret-Algorithmus mit projeziertem Gradientenabstieg.

Gradientenabstieg

Definition (Gradient)

Der Gradient einer differenzierbaren Function $f : \mathbb{R}^k \to \mathbb{R}$ am Punkt $x \in \mathbb{R}^k$ ist ein Vektor $\nabla f(x) \in \mathbb{R}^k$, der in die Gegenrichtung des stärksten Abstiegs zeigt.

Im Bandbreitenspiel sind die Kosten $c^t(x) = -u_i(x, s^t_{-i})$ nur eine Funktion von $x \in \mathbb{R}$, da s^t_{-i} nicht vom Spieler i beeinflusst werden kann. Dann ist ∇c^t die Ableitung von c^t nach x:

$$\nabla c^t(x) = 1 - \alpha_i \cdot \frac{\sum_{j \neq i} s^t_j}{\left(x + \sum_{j \neq i} s^t_j\right)^2} \quad .$$

Gradientenabstieg

Um eine *einzige* konvexe Funktion $f : \mathbb{R}^k \to \mathbb{R}$ zu minimieren, reicht es aus von einem beliebigen Startpunkt $x^1 \in \mathbb{R}^k$ mit Hilfe des Gradienten nacheinander kleine Schritte in die Richtung des stärksten Abstiegs zu machen

$$x^{t+1} = x^t - \eta \cdot \nabla f(x^t) \ .$$

Mit hinreichend kleiner Schrittlänge $\eta \in \mathbb{R}$ "sinken wir in das Tal" der konvexen Funktion und erreichen die Nähe (je nach η) des globalen Minimums.

Der Gradientenabstieg mit einer einzigen konvexen Funktion *f über einem konvexen Raum* $D \subset \mathbb{R}^k$ kann uns aus D herausführen. Wir nutzen einen Trick, um das zu verhindern: Wir schieben das Ergebnis zurück nach D wie folgt.

Projezierter Gradientenabstieg

Definition (Projektion)

Wir definieren eine Projektion $P : \mathbb{R}^k \to D$ mit $P(x) = x'$, wobei $x \in \mathbb{R}^k$ beliebig und $x' \in D$ der Punkt aus D am mit kürzester Distanz zu x.

Im Bandbreitenspiel gilt $P(x) = x$ für $x \in [b_{\min}, 1]$, $P(x) = 1$ für $x > 1$, und $P(x) = b_{\min}$ für $x < b_{\min}$.

Die Projektion wirft einen Punkt x zurück auf den nächstgelegenen Punkt in D.

Wir minimieren eine einzige konvexe Funktion f über einen konvexen und kompakten Raum D mit projeziertem Gradientenabstieg

$$x^{t+1} = P(x^t - \eta \nabla f(x^t)) \ .$$

Durch die Konvexität "laufen wir am Rand von D entlang" und sinken so zum Minimum von f in D.

GIGA

Im Online-Fall mit wechselnden Funktionen über T Runden nutzen wir den Generalized Infinitesimal Gradient Ascent oder GIGA Algorithmus:

$$\text{Wähle } x^1 \in D \text{ beliebig} \quad \text{und} \quad x^{t+1} = P(x^t - \eta \cdot \nabla c^t(x^t)) \ .$$

GIGA nutzt den Gradienten von c^t zur Optimierung von c^{t+1}. Das wirkt wie eine ziemlich schlechte Idee, denn c^{t+1} kann sich stark von c^t unterscheiden. Dennoch gilt ...

Satz (Zinkevich, 2003)

Sei $G \geq \|\nabla c^t(x)\|$ und $\Delta \geq \|x - y\|$ für jedes $x, y \in D$ und $t = 1, \ldots, T$. Wenn $\eta = \frac{\Delta}{G\sqrt{T}}$, dann erzeugt GIGA einen Regret von

$$R(T) \leq \Delta \cdot G \cdot \sqrt{T}.$$

Wenn G und Δ unabhängig von T sind, dann ist GIGA ein No-Regret Algorithmus.

Beweis des Satzes von Zinkevich

Intuition warum GIGA "funktioniert":

- Projezierter Gradientenabstieg klappt, wenn alle Kostenfunktionen c^t ähnlich sind.
- Wenn die Funktionen sehr unterschiedlich sind, erzeugt GIGA hohe Kosten, aber dann muss auch das Optimum $x^* \in D$ hohe Kosten haben.

Wir erfassen diese Intuition mit einem Potenzialargument:

- OBdA. sei das Optimum $x^* = 0$, der Ursprung eines Koordinatensystems.
- Betrachte "Potenzial" als $\Phi_t = \frac{1}{2\eta} \|x^t\|^2$.
- Φ_t liefert die Distanz von x^t zu $x^* = 0$

Lemma (Kosten-gegen-Distanz)

$$c^t(x^t) - c^t(0) + \Phi_{t+1} - \Phi_t \leq \eta \cdot G^2/2$$

Entweder Kosten sind fast optimal, oder mit x^{t+1} kommen wir näher an x^*.

Beweis des Kosten-gegen-Distanz Lemmas

Wir stellen fest, dass $\|P(x)\| \leq \|x\|$, da D konvex und die Projektion x immer in Richtung von D verschiebt (und damit näher zu $x^* = 0 \in D$).

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \frac{1}{2\eta}(\|x^{t+1}\|^2 - \|x^t\|^2) \\
&\leq \frac{1}{2\eta}(\|x^t - \eta \nabla c^t(x^t)\|^2 - \|x^t\|^2) \qquad (\text{da } \|P(x)\| \leq \|x\|) \\
&= \frac{1}{2\eta}(\|x^t\|^2 + \eta^2 \|\nabla c^t(x^t)\|^2 - 2\eta \nabla c^t(x^t) x^t - \|x^t\|^2)
\end{aligned}$$

Der letzte Schritt nutzt den Kosinussatz für Vektoren,
$\|u + v\|^2 = \|u\|^2 + \|v\|^2 + 2u^T v$.

Beweis des Kosten-gegen-Distanz Lemmas

Vereinfachen und Anwendung der Definition von G ergibt

$$\begin{aligned}\Phi_{t+1} - \Phi_t &\leq \frac{1}{2\eta}(\|x^t\|^2 + \eta^2\|\nabla c^t(x^t)\|^2 - 2\eta\nabla c^t(x^t)x^t - \|x^t\|^2) \\ &\leq \frac{1}{2}\eta G^2 - \nabla c^t(x^t)\cdot x^t \ .\end{aligned}$$

Konvexität bedeutet, dass eine Funktion "superlinear" wächst – formal:

$$c^t(0) - c^t(x^t) \geq \nabla c^t(x^t)\cdot(0 - x^t) \ .$$

Damit ergibt sich

$$\begin{aligned}\Phi_{t+1} - \Phi_t &\leq \eta G^2/2 - \nabla c^t(x^t)\cdot x^t \\ &\leq \eta G^2/2 + c^t(0) - c^t(x^t) \ ,\end{aligned}$$

und das Lemma ist bewiesen. □ (Lemma)

Beweis des Satzes von Zinkevich

Durch Aufsummieren von $t = 1, \ldots, T$ erhalten wir eine Teleskopsumme, und das Lemma ergibt

$$\begin{aligned}\sum_{t=1}^{T}(c^t(x^t) - c^t(0) + \Phi_{t+1} - \Phi_t) &= \Phi_{T+1} - \Phi_1 + \sum_{t=1}^{T}(c^t(x^t) - c^t(0)) \\ &\leq T \cdot \eta G^2/2 \ .\end{aligned}$$

Wir erinnern uns an $x^* = 0$ und nutzen $\frac{\Delta^2}{2\eta} \geq \Phi_t \geq 0$ und $\eta = \frac{\Delta}{G\sqrt{T}}$ für die folgende Herleitung:

$$\begin{aligned}R(T) &= \sum_{t=1}^{T}(c^t(x^t) - c^t(0)) \\ &\leq \Phi_1 - \Phi_{T+1} + \frac{T\eta G^2}{2} \leq \frac{\Delta^2}{2\eta} + \frac{T\eta G^2}{2} \\ &= \frac{\Delta G\sqrt{T}}{2} + \frac{\Delta G\sqrt{T}}{2} \ .\end{aligned}$$

Der Satz ist gezeigt. □ (Satz)

Konvergenz zum Gleichgewicht

Mit GIGA existiert ein No-Regret Algorithmus für unendliche Strategieräume. Damit können wir nun die Frage betrachten, wie sich ein Nash-Gleichgewicht im Bandbreitenspiel (und in allgemeineren Spielen) einstellen kann.

Konvergieren Spieler mit No-Regret Algorithmen zum Nash-Gleichgewicht?

Ein schnelles Experiment im Beispiel-Spiel von oben ergibt Folgendes...

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000
2	(21,22222)	1,00000	(43,44444)	1,00000	(65,66667)	1,00000

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000
2	(21,22222)	1,00000	(43,44444)	1,00000	(65,66667)	1,00000
3	(-0,77778)	0,45003	(-0,55556)	0,60716	(-0,33333)	0,76430

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000
2	(21,22222)	1,00000	(43,44444)	1,00000	(65,66667)	1,00000
3	(-0,77778)	0,45003	(-0,55556)	0,60716	(-0,33333)	0,76430
4	(-0,58664)	0,11133	(-0,26800)	0,45243	(-0,04408)	0,73885

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000
2	(21,22222)	1,00000	(43,44444)	1,00000	(65,66667)	1,00000
3	(-0,77778)	0,45003	(-0,55556)	0,60716	(-0,33333)	0,76430
4	(-0,58664)	0,11133	(-0,26800)	0,45243	(-0,04408)	0,73885
5	(-0,29793)	0,01000	(0,00210)	0,45348	(-0,00324)	0,73723

t	Spieler 1		Spieler 2		Spieler 3	
	$\nabla u_1(s^{t-1})$	s_1^t	$\nabla u_2(s^{t-1})$	s_2^t	$\nabla u_3(s^{t-1})$	s_3^t
1		0,01000		0,01000		0,01000
2	(21,22222)	1,00000	(43,44444)	1,00000	(65,66667)	1,00000
3	(-0,77778)	0,45003	(-0,55556)	0,60716	(-0,33333)	0,76430
4	(-0,58664)	0,11133	(-0,26800)	0,45243	(-0,04408)	0,73885
5	(-0,29793)	0,01000	(0,00210)	0,45348	(-0,00324)	0,73723
6	(-0,17409)	0,01000	(0,03659)	0,46985	(-0,03555)	0,72133
7	(-0,17441)	0,01000	(0,01375)	0,47546	(-0,00227)	0,72040
8	(-0,17759)	0,01000	(0,00461)	0,47720	(0,00157)	0,72099
9	(-0,17917)	0,01000	(0,00154)	0,47775	(0,00128)	0,72145
10	(-0,17984)	0,01000	(0,00051)	0,47792	(0,00075)	0,72170
11	(-0,18012)	0,01000	(0,00016)	0,47797	(0,00040)	0,72182
12	(-0,18024)	0,01000	(0,00004)	0,47798	(0,00021)	0,72189
13	(-0,18029)	0,01000	(0,00000)	0,47798	(0,00011)	0,72192
14	(-0,18031)	0,01000	(-0,00001)	0,47798	(0,00006)	0,72193
15	(-0,18032)	0,01000	(-0,00001)	0,47797	(0,00003)	0,72194
16	(-0,18033)	0,01000	(-0,00001)	0,47797	(0,00002)	0,72195
17	(-0,18033)	0,01000	(-0,00001)	0,47797	(0,00001)	0,72195
18	(-0,18033)	0,01000	(-0,00000)	0,47797	(0,00000)	0,72195

Konkave Spiele

Das Bandbreitenspiel ist ein Spezialfall einer großen Klasse von Spielen, die *sozial konkave Spiele* genannt werden. Jedes sozial konkave Spiel hat ein reines Nash-Gleichgewicht.

Definition

Ein sozial konkaves Spiel ist ein strategisches Spiel $\Gamma = (\mathcal{N}, (\Sigma_i)_{i\in\mathcal{N}}, (u_i)_{i\in\mathcal{N}})$.

1. $\mathcal{N}$ ist eine endliche Menge von n Spielern.
2. Jede Strategiemenge Σ_i ist kompakt und konvex.
3. Nutzenfunktion $u_i(s_i, s_{-i})$ ist konkav in s_i für jedes gegebene s_{-i}.
4. Nutzenfunktion $u_i(s_i, s_{-i})$ ist konvex in s_{-i} für jedes gegebene $s_i \in \Sigma_i$.
5. Es gibt $(\lambda_i)_{i\in\mathcal{N}}$ mit $\lambda_i > 0$, $\sum_i \lambda_i = 1$, so dass $g(s) = \sum_{i\in\mathcal{N}} \lambda_i u_i(s)$ eine konkave Funktion in s ist.

No-Regret Algorithmen in sozial konkaven Spielen

Satz

Wenn in einem wiederholten, sozial konkaven Spiel über T Zeitschritte jeder Spieler einen No-Regret Algorithmus für die Strategiewahl nutzt, dann gilt für $T \to \infty$:

1. *Der durchschnittliche Ablauf des Spiels $\hat{s}$ konvergiert zu einem (gemischten) Nash-Gleichgewicht.*
2. *Der durchschnittliche Nutzen jedes Spielers konvergiert zum Nutzen im gemischten Nash-Gleichgewicht.*

No-Regret Algorithmen in sozial konkaven Spielen

Wenn alle Spieler im wiederholten Bandbreitenspiel den GIGA Algorithmus für die Strategiewahl nutzen, dann konvergiert der durchschnittliche Ablauf des Spiels zu einem Nash-Gleichgewicht.

Daneben gibt es bei streng konkaven Nutzenfunktionen u_i ein eindeutiges gemischtes Gleichgewicht – und dies ist auch ein reines Gleichgewicht.

Daher beweist der Satz unsere Intuition aus dem Experiment: Im Bandbreitenspiel können die Spieler mit dem GIGA Algorithmus ein reines Nash-Gleichgewicht erlernen.

Gleichgewichte und Lernen

Wenn Spieler mit No-Regret Lernalgorithmen spielen, konvergiert ihr Verhalten nicht immer zum gemischten Gleichgewicht. Allerdings verschwindet für jeden Spieler der durchschnittliche Regret über die Zeit. Aus dieser Eigenschaft kann man ein (allgemeineres) Gleichgewichtskonzept ableiten.

Definition

Sei $\mathcal{V}$ eine Wahrscheinlichkeitsverteilung über die Zustände eines endlichen Spiels. $\mathcal{V}$ heißt grob-korreliertes Gleichgewicht wenn für jeden Spieler $i \in \mathcal{N}$ und jede Strategie $s_i' \in S_i$ gilt

$$\mathbb{E}_{s\sim\mathcal{V}}[c_i(s)] \quad \leq \quad \mathbb{E}_{s\sim\mathcal{V}}[c_i(s_i', s_{-i})] \ .$$

$\mathcal{V}$ heißt (additives) ε-approximatives grob-korreliertes Gleichgewicht wenn

$$\mathbb{E}_{s\sim\mathcal{V}}[c_i(s)] \quad \leq \quad \mathbb{E}_{s\sim\mathcal{V}}[c_i(s_i', s_{-i})] + \varepsilon \ .$$

No-Regret und Grob-Korrelierte Gleichgewichte

Betrachte einen Ablauf des Spiels $s^1, s^2, \ldots, s^T$ über T Runden. Wir interpretieren ihn als Verteilung über Zustände, in dem wir $k \in [T]$ uniform zufällig wählen.

Wenn Spieler i einen Regret von $R_i(T)$ hat, dann gilt für jede Strategie $s_i' \in S_i$

$$\begin{aligned} \mathbb{E}_{k\in[T]}[c_i(s^k)] &= \sum_{t=1}^{T} \frac{1}{T} \cdot c_i(s^t) \\ &\leq \sum_{t=1}^{T} \frac{1}{T} \cdot c_i(s_i', s_{-i}^t) + \frac{R_i(T)}{T} \\ &= \mathbb{E}_{k\in[T]}[c_i(s_i', s_{-i}^k)] + \frac{R_i(T)}{T} \quad . \end{aligned}$$

Proposition

Wenn jeder Spieler nach T Runden Regret höchstens R hat, dann stellt der Ablauf des Spiels ein $\frac{R}{T}$-approximatives grob-korreliertes Gleichgewicht dar.

Korrelierte Gleichgewichte

Wenn alle Spieler RWM nutzen, dann erreichen wir schnell ein ε-approximatives grob-korreliertes Gleichgewicht, d.h. schon nach $T = \frac{4}{\varepsilon^2} \cdot \log(\max_i |S_i|)$ Runden.

Allerdings liefern grob-korrelierte Gleichgewichte eine deutlich schwächere Garantie als z.B. gemischte Nash-Gleichgewichte. Sind grob-korrelierte Gleichgewichte das stärkste Gleichgewichtskonzept, das schnell approximiert werden kann?

Definition

Sei $\mathcal{V}$ eine Wahrscheinlichkeitsverteilung über die Zustände eines endlichen Spiels. $\mathcal{V}$ heißt korreliertes Gleichgewicht wenn für jeden Spieler $i \in \mathcal{N}$ und jede Austauschfunktion $\sigma : S_i \to S_i$ gilt

$$\mathbb{E}_{s\sim\mathcal{V}}[c_i(s)] \quad \leq \quad \mathbb{E}_{s\sim\mathcal{V}}[c_i(\sigma(s_i), s_{-i})] \ .$$

$\mathcal{V}$ heißt (additives) ε-approximatives korreliertes Gleichgewicht wenn

$$\mathbb{E}_{s\sim\mathcal{V}}[c_i(s)] \quad \leq \quad \mathbb{E}_{s\sim\mathcal{V}}[c_i(\sigma(s_i), s_{-i})] + \varepsilon \ .$$

Korrelierte Gleichgewichte

Im gemischten Gleichgewicht wählt jeder Spieler eine eigene Verteilung x_i über seine Strategien. Die Verteilung über Zustände ergibt sich erst aus der unabhängigen Kombination der x_i. Keiner der Spieler möchte zu einer anderen (reinen) Strategie abweichen.

In korrelierten und grob-korrelierten Gleichgewichten wird direkt eine Verteilung über Zustände angegeben, die die Strategiewahlen einzelner Spieler miteinander korrelieren kann (z.B. als Resultat von gemeinsamem Lernen über die Zeit).

Keiner der Spieler möchte zu einer anderen (reinen) Strategie abweichen:

- Korreliert: Unter der Bedingung, dass für Spieler i Strategie s_i auswürfelt wurde, möchte i nicht zu einer anderen reinen Strategie abweichen.
- Grob-korreliert: Keiner der Spieler möchte immer zu einer reinen Strategie abweichen (hier wird nicht auf eine auswürfelte Strategie s_i bedingt).

Beispiel

Zwei Autofahrer rasen auf eine Kreuzung zu.

	(G)as	(B)remsen
(G)as	101 101	2 0
(B)remsen	0 2	1 1

Beispiele für Gleichgewichte:

- Rein: Zustände (G,B) und (B,G)
- Gemischt (aber nicht rein): Beide Spieler spielen $x_G = 0.01$ und $x_B = 0.99$.
- Korreliert (aber nicht gemischt):
  $\Pr_{s\sim\nu}[s = (B, G)] = 0.5$ und
  $\Pr_{s\sim\nu}[s = (G, B)] = 0.5$

Beispiel

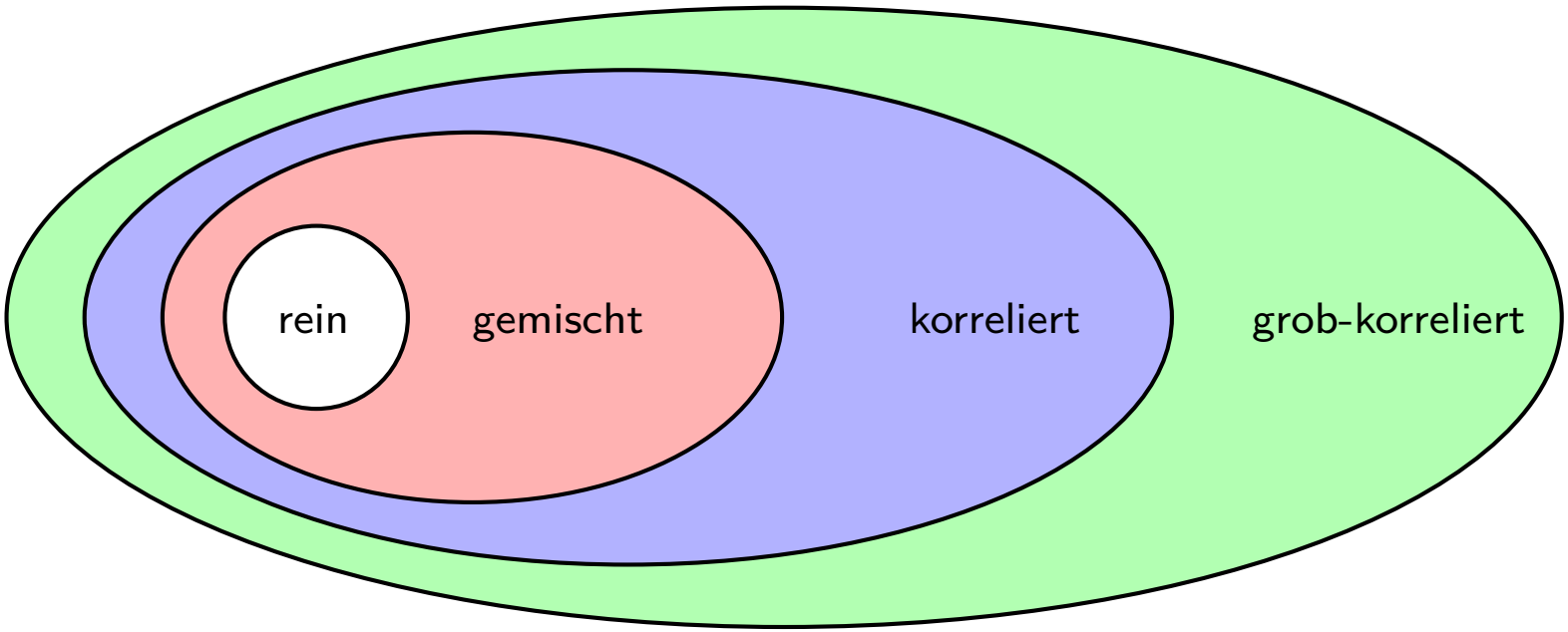
Zwei Autofahrer rasen auf eine Kreuzung zu.

	(G)as	(B)remsen
(G)as	101 101	2 0
(B)remsen	0 2	1 1

Beispiele für Gleichgewichte:

- Man kann sich das korrelierte Gleichgewicht vorstellen wie eine Ampel: Jedem Spieler wird ein Vorschlag gemacht. Kein Spieler will unilateral die empfohlene Strategie ändern, wenn der andere die Empfehlung befolgt.
- Grob-Korreliert (aber nicht korreliert): In symmetrischen 2×2-Spielen nicht möglich (jedes grob-korrelierte Gleichgewicht ist korreliert), siehe Übung.

Eine Hierarchie von Gleichgewichtskonzepten



Erlernen von Korrelierten Gleichgewichten

Können wir korrelierte Gleichgewichte erlernen?

Dafür betrachten wir Algorithmen mit einer stärkeren Garantie im Experten Problem. Für Algorithmus H sei $H(t) \in [N]$ der in Schritt t gewählte Experte. Bisher haben wir den Verlust des besten einzelnen Experten als Vergleichswert für Regret betrachtet. Jetzt betrachten wir die beste Funktion von Experten

$$\sigma^* \in \arg \max_{\sigma:[N]\to[N]} \sum_{t=1}^{T} \ell^t_{\sigma(H(t))} \ ,$$

mit $L^T_{s\min} = \sum_{t=1}^{T} \ell^t_{\sigma^*(H(t))}$ dem Verlust der besten Funktion von Experten.

Swap-Regret

Definition

Der Swap-Regret von H ist gegeben durch

$$SR(T) \quad = \quad L_H^T - L_{s\min}^T \quad = \quad \sum_{t=1}^{t} \ell_H^t - \sum_{t=1}^{T} \ell_{\sigma^* H(t)}^t \ .$$

Swap-Regret beschreibt den Verlust gegenüber Austauschen von Experten.
Wenn $\sigma^*(i) = j$, dann bedeutet dies intuitiv:

Wann immer H Experte i gewählt hat, hätte er lieber Experte j wählen sollen.

Für einen No-Swap-Regret Algorithmus gilt $SR(T)/T \to 0$ für $T \to \infty$.

Es gilt $SR(T) \geq R(T)$ und No-Swap-Regret $\Rightarrow$ No-Regret. (Warum?)

Swap-Regret und korreliertes Gleichgewicht

Betrachte einen Ablauf des Spiels $s^1, s^2, \ldots, s^T$ über T Runden und interpretiere ihn als Verteilung über Zustände.

Wenn Spieler i einen Swap-Regret von $SR_i(T)$ hat, dann gilt für jede Strategie s_i und jede Funktion $\sigma : S_i \to S_i$:

$$\begin{aligned}
\mathbb{E}_{k\in[T]}[c_i(s^k)] &= \sum_{t=1}^{T} \frac{1}{T} \cdot c_i(s_i^t, s_{-i}^t) \\
&\leq \sum_{t=1}^{T} \frac{1}{T} \cdot c_i(\sigma(s_i^t), s_{-i}^t) + \frac{SR_i(T)}{T} \\
&= \mathbb{E}_{k\in[T]}[c_i(\sigma(s_i^k), s_{-i}^k)] + \frac{SR_i(T)}{T} \quad .
\end{aligned}$$

Proposition

Wenn jeder Spieler nach T Runden Swap-Regret höchstens R hat, dann stellt der Ablauf des Spiels ein $\frac{R}{T}$-approximatives korreliertes Gleichgewicht dar.

No-Swap-Regret Algorithmen

Wir können No-Regret Algorithmen in No-Swap-Regret Algorithmen verwandeln!

Satz

Sei H ein Algorithmus mit Regret $R(T)$. Dann existiert ein Algorithmus M mit Swap-Regret $SR(T) = N \cdot R(T)$.

Beweis:

Wir nutzen N Kopien $H_1, \ldots, H_N$ von Algorithmus H. In gewisser Weise sichert uns H_j zu, dass keine Abbildung σ von Experte j auf $\sigma(j)$ allzu großen Swap-Regret erzeugt.

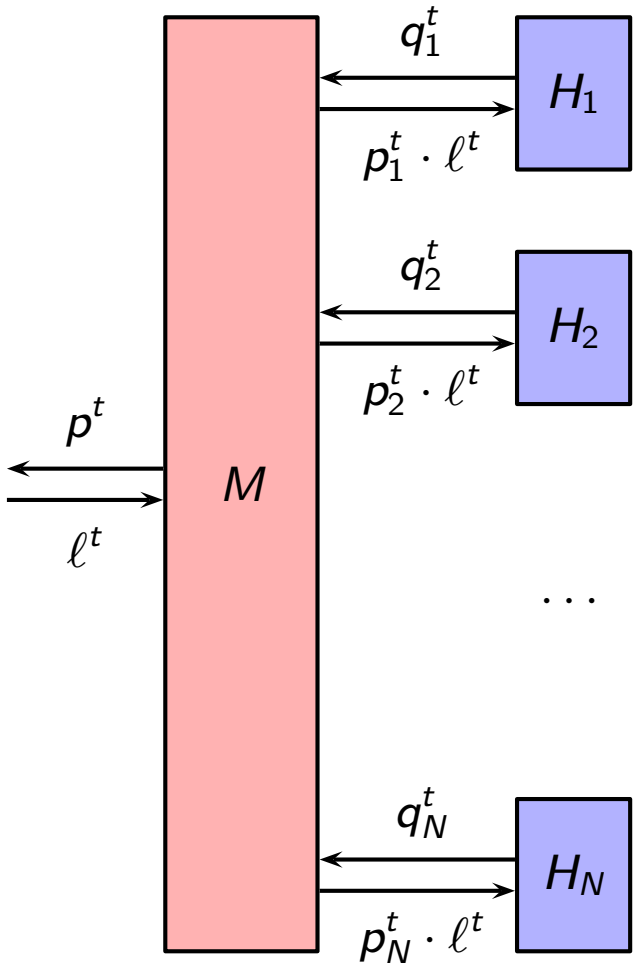
Ein Master-Algorithmus M koordiniert die Kopien von H.

Master-Algorithmus

Master-Algorithmus

In jedem Schritt t,

1. Erhalte Verteilungen $q_1^t, \ldots, q_n^t$ von $H_1, \ldots, H_N$;
2. Errechne Konsens-Verteilung p^t;
3. Wähle Experte i mit W.keit p_i^t;
4. Melde an H_j einen Verlust von $p_j^t \cdot \ell_i^t$, für jeden Experten $i \in N$;



Master-Algorithmus: No-Swap-Regret

Wir betrachten die Errechnung von p^t in Zeile 2. am Ende des Beweises.

Für Swap-Regret betrachten wir die folgenden Terme:

- Verlust des Master-Algorithmus: $L_M^T = \sum_{t=1}^{T} \sum_{i \in [N]} p_i^t \ell_i^t$
- Verlust mit bester Abbildung σ^*: $L_{s\min}^T = \sum_{t=1}^{T} \sum_{i \in [N]} p_i^t \ell_{\sigma^*(i)}^t$

Algorithmus H_j denkt, wir wählen Experte i mit Wahrscheinlichkeit $q_{j,i}^t$ und die Kosten dafür wären $p_j^t \ell_i^t$.

- Interner Verlust von H_j: $L_{H_j}^T = \sum_{t=1}^{T} \sum_{i \in [N]} q_{j,i}^t \cdot (p_j^t \ell_i^t)$
- Bester Experte k_j für H_j: $L_{\min,j}^T = \sum_{t=1}^{T} p_{k_j}^t \ell_{k_j}^t$

H_j hat Regret höchstens $R(T)$, also $L_{H_j}^T \leq L_{\min,j}^T + R(T)$.

Master-Algorithmus: No-Swap-Regret

Sei nun σ^* die beste Abbildung der Experten. Dann gilt mit Aufsummieren:

$$\begin{aligned}\sum_{j\in[N]} L_{H_j}^T &\leq \sum_{j\in[N]} L_{\min,j}^T + R(T) \\ &\leq \sum_{j\in[N]}\sum_{t=1}^{T} p_j^t \ell_{\sigma^*(j)}^t + \sum_{j\in[N]} R(T) \\ &= L_{s\min}^T + N \cdot R(T)\end{aligned}$$

Für die Summe der internen Verluste haben wir die No-Swap-Regret Garantie.

Wie hängt der tatsächliche Verlust L_M^T mit der Summe der internen Verluste $\sum_j L_{H_j}^T$ zusammen?

Dafür betrachten wir nun die Konstruktion der Konsens-Verteilung p^t.

Master-Algorithmus: Konsens-Verteilung

Wähle die Konsens-Verteilung p^t als

$$p_i^t = \sum_{j \in [M]} q_{j,i}^t p_j^t \ .$$

Dann gilt:

$$\begin{aligned} L_M^T &= \sum_{t=1}^{T} \sum_{i \in [M]} p_i^t \ell_i^t \\ &= \sum_{t=1}^{T} \sum_{i \in [M]} \sum_{j \in [M]} q_{j,i}^t p_j^t \ell_i^t \\ &= \sum_{j \in [M]} L_{H_j}^T \\ &\leq L_{s\min}^T + N \cdot R(T) \ , \end{aligned}$$

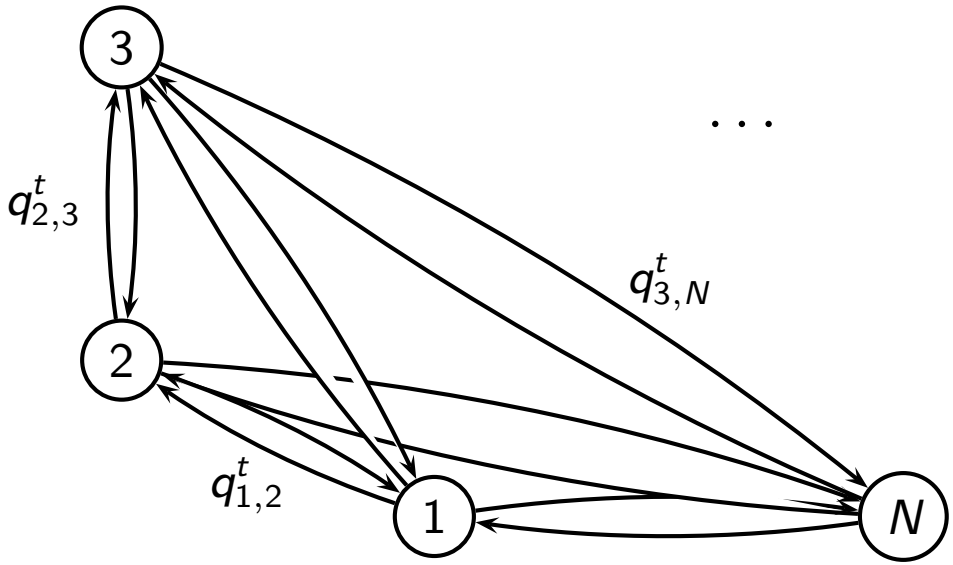
und der Satz ist gezeigt. □

Master-Algorithmus: Markov-Kette

p^t ist die stationäre Verteilung der folgenden Markov-Kette:

Die Zustände sind alle Experten. Wenn wir bei Zustand/Experte i sind, dann ist die Wahrscheinlichkeit zu Zustand/Experte j zu wechseln gegeben durch $q^t_{i,j}$. Matrix Q^t mit $Q^t(i,j) = q^t_{i,j}$ gibt die Übergangswahrscheinlichkeiten von Zustand/Experte i an.

Wähle p^t als die stationäre Verteilung der Markov-Kette, den dominanten Eigenvektor der Matrix Q^t. p^t kann effizient berechnet werden.



Übersicht: Gleichgewichte, Berechnung und Konvergenz

Reines Nash-Gleichgewicht

- Existiert nicht immer
- Berechnung polynomiell in der Kostenmatrix des Spiels (die ist riesig)
- PLS-schwer in kompakt repräsentierten Auslastungsspielen
- Konvergenzzeit von Beste-Antwort-Dynamik exponentiell
- Polynomielle Konvergenzzeit für spezielle Auslastungsspiele

Gemischtes Nash-Gleichgewicht

- Existiert in endlichen Spielen (Satz von Nash)
- Berechnung in PPAD (Sperners Lemma) und PPAD-schwer
- 2-Spieler-Nullsumme: Berechnung in polynomieller Zeit mit LP
- 2-Spieler-Nullsumme: Konvergenz in polynomieller Zeit mit No-Regret
- 2-Spieler-Nullsumme: Konvergenz im Ablauf mit vRWM-Algorithmus
- Sozial-Konkav: Konvergenz in polynomieller Zeit mit No-Regret

Übersicht: Gleichgewichte, Berechnung und Konvergenz

Korreliertes Gleichgewicht

- Existiert in endlichen Spielen (verallgemeinert gemischtes Ggw.)
- Verteilung über Zustände ohne profitable "Swap"-Abweichungen
- Konvergenz in polynomieller Zeit mit No-Swap-Regret

Grob-korreliertes Gleichgewicht

- Existiert in endlichen Spielen (verallgemeinert korreliertes Ggw.)
- Verteilung über Zustände ohne profitable "bester-Experte"-Abweichungen
- Konvergenz in polynomieller Zeit mit No-Regret

Beachte: Konvergenz für No-Regret oder No-Swap-Regret bezieht sich auf ε-approximative Varianten der Gleichgewichte. ε sinkt als Funktion von T.

Literatur

- Nisan, Roughgarden, Tardos, Vazirani. Algorithmic Game Theory, 2007. (Kapitel 4).
- Littlestone, Warmuth. The Weighted Majority Algorithm. Information & Computation 108(2):212–261, 1994.
- Freund, Schapire. Adaptive Game Playing using Multiplicative Weights. Games and Economic Behavior, 29:79–103, 1999.
- Roughgarden. Twenty Lectures on Algorithmic Game Theory, 2016. (Kapitel 17+18).
- Rosen. Existence and Uniqueness of Equilibrium Points for Concave n-Person Games. Econometrica, 33(3):520–534, 1965.

Literatur

- Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. ICML 2003.
- Even-Dar, Mansour, Nadav. On the Convergence of Regret Minimization Dynamics in Concave Games. STOC 2009.
- Blum, Mansour. From External to Internat Regret. Journal of Machine Learning Research 8:1307–1324, 2007.