Seminar über Algorithmen 05.11.2013

Regret Minimization

Alexander Kauer *Wolfgang Mulzer, Yannik Stein*

1 Allgemeines

1.1 Motivation

Wenn Spiele in unsicheren oder unbekannten Umgebungen gespielt werden, dann kann bei Entscheidungen bzgl. der Strategie nicht auf Informationen der anderen Spieler zurückgegriffen werden, was das Abschätzen der eventuellen Gewinne bzw. Verluste bei der Wahl der eigenen Strategie erschwert. Im Folgenden werden Online-Algorithmen vorgestellt, mit dessen Hilfe trotz dieser Einschränkung möglichst niedrige Verluste bzw. hohe Gewinne erzielt werden können.

1.2 Allgemeine Definitionen

Definition 1. *Sei $X = \{1, 2, \ldots, N\}$ die Menge der verfügbaren Aktionen für den von uns betrachteten Spieler. Zu jedem Zeitpunkt t wählt der gewählte Online-Algorithmus H eine Wahrscheinlichkeitsverteilung p^t über X aus, welche der Spieler zu diesem Zeitpunkt dann als Strategie spielt. Nach diesem Zeitpunkt wird der Verlustvektor $l^t \in [0,1]^N$ ermittelt, wobei H den Verlust $l_H^t = \sum_{i=1}^N p_i^t l_i^t$ erfährt. Weiterhin wird der Gesamtverlust bis Zeitpunkt T der i-ten Aktion mit $L_i^T = \sum_{t=1}^T l_i^t$ bezeichnet, der gesamte Verlust von H bis Zeitpunkt T mit $L_H^T = \sum_{i=1}^T l_H^t$.*

Abhängig vom *Information Model* erhält H verschiedene Informationen über die Verluste:

Definition 2. *Im* Full Information Model *erhält H den kompletten Verlustvektor l^t und erfährt einen Verlust von $l_H^t = \sum_{i=1}^N p_i^t l_i^t$. Im* Partial Information Model *erhält H hingegen ausschließlich l_H^t, d.h. es ist nicht bekannt, wie viel Verlust die Nichtgewählten genau verursacht hätten.*

Im Nachfolgenden wird - sofern nicht anders vermerkt - vom *Full Information Model* ausgegangen.

2 External Regret Minimization

2.1 Allgemeines

Sei $\mathcal{G}$ eine Menge an Algorithmen für das Spiel. Ziel bei der External Regret Minimization ist es, mit dem Online-Algorithmus H möglichst nah an den besten Algorithmus aus $\mathcal{G}$ bzgl. des Verlustes mit $L_{\mathcal{G},\min}^T = \min_{g \in \mathcal{G}} L_g^T$ zu kommen. Die Differenz des Verlustes zwischen dem besten Algorithmus aus $\mathcal{G}$ und H nennt sich hierbei Regret mit $R_{\mathcal{G}} = L_H^T - L_{\mathcal{G},\min}^T$ und wird zu minimieren versucht.

Weiterhin ist festzuhalten, dass die Vergleichsalgorithmen $\mathcal{G}$ vergleichsweise eingeschränkt sein müssen, weil ansonsten eine Regret Minimization wenig Sinn ergibt:

Satz 3. *Sei $\mathcal{G}_{all}$ die Menge von allen möglichen Algorithmen bis Zeitpunkt T. Für jeden beliebigen Online-Algorithmus H gibt es bei der Ausführung T Verlustvektoren mit einem Regret von mindestens $R_{\mathcal{G}_{all}} = T(1 - \frac{1}{N})$.*

Beweis. Solch T Verlustvektoren lassen sich konstruieren, indem zu jedem Zeitpunkt t die Aktion i^t von H mit der niedrigsten Wahrscheinlichkeit p_i^t einen Verlust von 0 zugewiesen bekommt und alle anderen einen Verlust von 1. Die niedrigste Wahrscheinlichkeit ist maximal, wenn alle Aktionen gleich wahrscheinlich sind, folglich beträgt die niedrigste Wahrscheinlichkeit maximal $\frac{1}{N}$, der Verlust pro Zeitpunkt t mindestens $1 - \frac{1}{N}$ und insgesamt somit mindestens $T(1 - \frac{1}{N})$. In $\mathcal{G}_{\text{all}}$ gibt es hingegen einen Algorithmus, welcher einen Verlust von insgesamt 0 erzielt. □

Aus diesem Grund wird im Folgenden von einer Menge $\mathcal{G}$ ausgegangen, dessen Algorithmen ihren Index in $\mathcal{G}$ als Aktion zu jedem Zeitpunkt wählen (mit $|\mathcal{G}| = N$).

2.2 Algorithmen

2.2.1 Greedy-Algorithmus

Beim Greedy-Algorithmus wird zu jedem Zeitpunkt t deterministisch die bis dato beste Aktion x^t ausgewählt:

$$\begin{array}{ll} \text{Start:} & x^1 := 1 \\ \text{Zeitpunkt } t\text{:} & S^{t-1} := \{i : L_i^{t-1} = L_{\min}^{t-1}\} \\ & x^t := \min S^{t-1} \end{array}$$

Satz 4. *Für den Greedy-Algorithmus gilt für jede beliebige Reihenfolge an Verlusten*

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

Beweis. Zu jedem Zeitpunkt, an dem dieser Algorithmus einen Verlust (von bis zu 1) erhält und $L_{\min}$ sich nicht verändert, muss mindestens eine Aktion aus S^t entfernt werden. Dies kann maximal $|N|$ mal geschehen, bevor $L_{\min}^t$ erhöht werden muss. □

Allerdings ist diese lineare obere Schranke nicht sonderlich gut, jedoch für diese Art von Algorithmen nahezu ideal:

Satz 5. *Für jeden beliebigen deterministischen Algorithmus H existiert eine Verlustvektorsequenz, für welche sich $L_H^T = T$ und $L_{\min}^T \leq \lfloor \frac{T}{N} \rfloor$ ergeben.*

Beweis. Sei x^t die Aktion, welche der Algorithmus zu Zeitpunkt t auswählt. Setze l^t folgendermaßen:

$$l_i^t = \begin{cases} 1 & \text{wenn } i = x^t \\ 0 & \text{sonst} \end{cases}$$

Somit ist $L_H^T = T$. Da es N verschiedene Aktionen gibt, muss es mindestens eine Aktion geben, welche von H maximal $\lfloor \frac{T}{N} \rfloor$ ausgewählt wurde. Da ausschließlich die Aktionen von H einen Verlust bedeuten, ergibt sich $L_{\min}^T \leq \lfloor \frac{T}{N} \rfloor$. □

2.2.2 Randomized-Greedy-Algorithmus

Um dieses Problem zu umgehen, kann der Algorithmus randomisiert werden. Es wird nun statt einer einzelnen Aktion x^t eine Wahrscheinlichkeitsverteilung p^t über die bis dato besten Aktionen ausgewählt:

$$\begin{array}{ll} \text{Start:} & p_i^1 := \frac{1}{N} \\ \text{Zeitpunkt } t\text{:} & S^{t-1} := \{i : L_i^{t-1} = L_{\min}^{t-1}\} \\ & p_i^t := \begin{cases} \frac{1}{|S^{t-1}|} & \text{falls } i \in S^{t-1} \\ 0 & \text{sonst} \end{cases} \end{array}$$

Satz 6. *Für den Randomized-Greedy-Algorithmus gilt für jede beliebige Reihenfolge an Verlusten*

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

(ohne Beweis)

2.2.3 Polynomial-Weights-Algorithmus

Es lassen sich noch bessere Ergebnisse erzielen, wenn man den einzelnen Aktionen Gewichte w^t zuordnet und die Wahrscheinlichkeiten abhängig von den Gewichten macht (und somit nicht nur die bis dato besten Aktionen in die Berechnung miteinbezieht):

$$\begin{array}{ll} \text{Start:} & w_i^1 := 1,\ p_i^1 := \frac{1}{N}, i \in X \\ \text{Zeitpunkt } t\text{:} & w_i^t := w_i^{t-1}(1 - \eta l_i^{t-1}) \\ & p_i^t := \frac{w_i^t}{W^t} \text{ mit } W^t := \sum_{i \in X} w_i^t \end{array}$$

Satz 7. *Für den Polynomial Weights Algorithmus gilt mit $\eta \leq 0.5$, jeder Verlustvektorsequenz und jedem $k \in X$*

$$L_{PW}^T \leq L_k^T + \eta Q_k^T + \frac{\ln N}{\eta}$$

mit $Q_k^T = \sum_{t=1}^T (l_k^t)^2$. Setzt man $\eta = \min\{\sqrt{\frac{\ln N}{T}}, 0.5\}$, dann ergibt sich mit dem Wissen von $Q_k^T \leq T$ die Ungleichung $L_{PW}^T \leq L_{\min}^T + 2\sqrt{T \ln N}$.

Beweis. Sei $F^t = \frac{\sum_i w_i^t l_i^t}{W^t}$ der Verlust zum Zeitpunkt t. Mit dieser Definition und den Aktualisierungsregeln ergeben sich $W^{t+1} = W^t - \sum_i \eta w_i^t l_i^t = W^t(1 - \eta F^t)$ und $W^{T+1} = N \prod_{t=1}^T (1 - \eta F^t)$.
Obere Schranke:

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

Untere Schranke (mit $k \in X$):

$$\begin{array}{rcl} \ln W^{T+1} & \geq & \ln w_k^T + 1 = \sum_{t=1}^T \ln(1 - \eta l_k^t) \\ & \overset{\ln(1-z) \geq -z-z^2, 0 \leq z \leq 0.5)}{\geq} & -\sum_{t=1}^T \eta l_k^t - \sum_{t=1}^T (\eta l_k^t)^2 = -\eta L_k^T - \eta^2 Q_k^T \end{array}$$

Kombiniert man die beiden Ungleichungen, ergibt sich die Formel aus dem Satz. □

2.2.4 Untere Grenzen für External Regret Minimization

Der Polynomial-Weights-Algorithmus verhält sich bzgl. unterer Grenze nahezu optimal. Es ist für einen beliebigen Algorithmus für kleine $T < \log_2 N$ nicht möglich, sich besser als linear und für sonstige T nicht besser als $\Omega(\sqrt{T})$ bzgl. des Regrets zu verhalten.

Satz 8. *Sei $T < \log_2 N$. Dann gibt es für jeden Algorithmus A_1 eine Verlustvektorsequenz, wodurch sich zwingend $E[L_{A_1}^T] = \frac{T}{2}$ und $L_{\min} = 0$ ergeben.*

Beweis. Setze zu jedem Zeitpunkt t den Verlustvektor folgendermaßen fest: Setze Position i des Verlustvektors auf 1, falls es einen Vektor bei Zeitpunkt $t-1$ gab und Position i ebenfalls auf 1 gesetzt war. Setze anschließend die restlichen Elemente je zur Hälfte auf 0 und 1.

Diese Konstruktion sorgt dafür, dass jeder Algorithmus zu jedem Zeitpunkt mindestens einen Verlust von 0.5 im Erwartungswert erfährt, obwohl es zum Zeitpunkt T noch mindestens eine Aktion mit einem Gesamtverlust von 0 gibt. □

Satz 9. *Angenommen $N = 2$. Dann gibt es für jeden Algorithmus A_2 eine Verlustvektorsequenz, wodurch sich zwingend $E[L_{A_2}^T - L_{\min}] \in \Omega(\sqrt{T})$ ergibt.*

Beweis. Werfe zu jedem Zeitpunkt t eine Münze und setze $l^t := z_1 = (0,1)$ oder $l^t := z_2 = (1,0)$. Nach T Würfen ergeben sich $\frac{T}{2} + y$ Verluste z_1 und $\frac{T}{2} - y$ Verluste z_2 mit $\frac{T}{2} - L_{\min} = |y|$. Nun wird die untere Grenze für $E[|y|]$ benötigt; diese ergibt sich aus der Wahrscheinlichkeit von y mit $\binom{T}{0.5T+y} \cdot 2^{-T} \in \mathcal{O}(\frac{1}{\sqrt{T}})$ nach Sterling zu

$$E[|y|] \in \Omega(\sqrt{T})$$

□

Weiterhin kann jeder andere Fall mit $N > 2$ auf $N = 2$ zurückgeführt werden, indem an allen Postionen außer den ersten beiden ausschließlich immer ein Verlust von 1 über den Verlustvektor generiert wird und ansonsten wie oben verfahren wird.

2.3 Das Partial Information Model

Die bisherigen Betrachtungen bzgl. des External Regrets haben jedoch eine Einschränkung: Sie nehmen an, dass der Algorithmus H den Verlustvektor $l^t \in [0,1]^N$ erhält. In der Realität hingegen erhält H jedoch oft nur den Gesamtverlust $\sum_i p_i^t l_i^t$ mitgeteilt und die in den vorherigen Abschnitten vorgestellten Algorithmen lassen sich nicht (direkt) anwenden.

Sei H ein Algorithmus mit oben bereits beschriebener Einschränkung und O ein (normaler) Algorithmus zur Regretminimierung mit Regret R. Wir teilen die T Zeitpunkte in K Blöcke B^τ auf, wobei O nun auf diesen Blöcken statt auf allen Zeitpunkten arbeitet; wir lassen O in jedem Schritt eine Verteilung p^τ für den gesamten Block wählen, führen H mit dieser Verteilung mit eingestreuten "Erkundszeiträumen" aus und geben den Durchschnitt des Blockes $c^\tau = \sum_{t \in B^\tau} \frac{p^\tau \cdot l^t}{|B^\tau|}$ zurück zu O für die Berechnung von $p^{\tau+1}$ (und nehmen dabei zunächst an, dass wir das l^t von H haben). Angenommen $C_i^K = \sum_{\tau=1}^K c_i^\tau$ und $C_{\min}^K = \min_i C_i^K$, dann ergibt sich für O bzgl. der c^τ maximal ein Verlust von $C_{\min} + R$ und somit für alle Zeitpunkte

$$L_O^T = \sum_{\tau=1}^{K} \sum_{t \in B^\tau} p^\tau \cdot l^t = \frac{T}{K} \sum_{\tau=1}^{K} p^\tau \cdot c^\tau \leq \frac{T}{K} \left(C_{\min}^K + R \right)$$

Nun muss H nur noch so entworfen werden, dass trotz des Erkundens L_O^T gültig bleibt. Dies kann erreicht werden, indem wir statt c^τ eine Zufallsvariable c'^τ mit $E[c'^\tau] = c^\tau$ generieren. Dies ist möglich, indem wir für jedes $i \in X$ ein $t_i \in B^\tau$ (mit $i \neq j \rightarrow t_i \neq t_j$) wählen und zu Zeitpunkt t_i Aktion i mit Wahrscheinlichkeit 1 spielen und $c_i'^\tau$ auf den Verlust setzen und ansonsten p^τ spielen. Somit ergibt sich im Erwartungswert L_O^T zuzüglich der Verluste beim Erkunden:

$$E[L_H^T] \leq NK + \frac{T}{K}\left(C_{\min}^K + R\right)$$

3 Swap Regret Minimization

3.1 Definitionen

Definition 10. *Eine Modification Rule ist eine Funktion F, welche sämtliche in Vergangenheit als auch die aktuell vom Algorithmus selektierte Strategie erhält und womöglich eine neue Strategie zurückliefert, welche stattdessen gespielt werden soll.*

Definition 11. *Sei für $x_1, b_1, b_2 \in X$ $switch_i(x_1, b_1, b_2)$ für Spieler i die folgende Funktion:*

$$switch_i(x_1, b_1, b_2) = \begin{cases} b_2, & \text{falls } x_1 = b_1 \\ x_1, & \text{sonst} \end{cases}$$

Mit diesen Definitionen lässt sich nun Regret allgemeiner definieren:

Definition 12. *Gegeben eine Modification Rule F (und Verlustfunktion $s_i(x) = s_i(x_i, x_{-i}) = l_H^t$ mit x als Verteilungssvektor aller Spieler, siehe vorherige Vorträge), dann gilt*

$$regret_i(x, F) = s_i(x) - s_i(F(x_i), x_{-i})$$

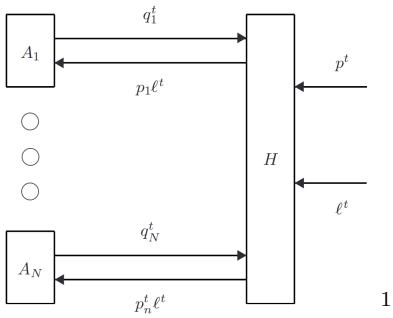
Falls $F(x_1) = switch_i(x_1, b_1, b_2)$, dann nennt sich dies Swap Regret.

Ebenso erlauben diese Definitionen eine andere, äquivalente Definition eines korrellierten Equilibriums:

Definition 13. *Ein Spiel mit den Spielern $M = \{1, \ldots, m\}$ und den gewählten Strategien x der Spieler befindet sich in einem korrelliertem Equilibrium genau dann, wenn gilt*
$\forall i \in M\ \forall F : X_i \rightarrow X_i\ E[regret_i(x, F)] \leq 0$

3.2 Reduktion von External Regret zu Swap Regret

Um einen Algorithmus H mit gutem Swap Regret zu erhalten, kann man einen Algorithmus A mit (gutem) External Regret R wiederverwenden. Die Idee hierbei ist, dass man von N Instanzen $A_1, \ldots, A_N$ die gewählten Wahrscheinlichkeiten q_i zu einem p kombiniert und anschließend den Verlust l abhängig von p wieder an die A_i zurückgibt.



[1]

Wie zuvor bereits angedeutet nehmen wir von einem Online-Algorithmus mit einem External Regret von R insgesamt N Instanzen $A_1, \ldots, A_N$ und führen diese zu jedem Zeitpunkt t aus, womit diese jeweils Verteilungen q_i^t erzeugen. Diese Verteilungen werden dann zu einer Verteilung p^t von H zusammengesetzt mit $\forall j \in \{1, \ldots, N\}\ p_j^t = \sum_i p_i^t q_{i,j}^t$ bzw. alternativ $p^t = p^t Q^t$, wobei Q^t eine Matrix aus den $q_{i,j}$ darstellt.

Zum einen ist diese Art von Gleichung immer und effizient lösbar, zum anderen erlaubt sie zwei verschiedene, aber äquivalente, Betrachtungsweisen: Es kann die Gleichung so betrachetet werden, dass das p^t jede Aktion j mit einer gewissen Wahrscheinlichkeit selektiert, es kann die Gleichung aber auch so betrachtet werden, dass p^t ein A_i mit einer gewissen Wahrscheinlichkeit auswählt und dieses dann die jeweilige Aktion selektiert.

Nachdem mit dem p^t gespielt wurde, wird an die jeweiligen A_i der für sie relevante Teil des Verlustvektors mit $p_i^t l^t$ zurückgegeben, womit jedes A_i einen Verlust von $(p_i^t l^t) \cdot q_i^t = p_i^t (q_i^t \cdot l^t)$ erfährt, wodurch sich für jedes A_i ergibt $\forall i, j \in \{1, \ldots, N\}\ \sum_{t=1}^T p_i^t (q_i^t \cdot l^t) \leq \sum_{t=1}^T p_i^t l_j^t + R$. Beachtet man $\sum_i p_i^t (q_i^t \cdot l^t) = p^t Q^t l^t = p^t l^t$ und dass die obige Formel für alle j und somit auch für alle $F : \{1, \ldots, N\} \to \{1, \ldots, N\}$ gilt , kann man nun über alle A_i die oben genannte Formel summieren und erhält

$$L_H^T \leq \sum_{i=1}^N \sum_{t=1}^T p_i^t l_{F(i)}^t + NR = L_{H,F}^T + NR$$

Quelle: Nisan, Roughgarden, Tardos, Vazirani: Algorithmic Game Theory, Kapitel 4.

[1] aus Nisan, Roughgarden, Tardos, Vazirani: Algorithmic Game Theory, Seite 92