Informatik IV - Tutorium XII & XIII (SR -120)

Tut Nr. 5 – Wdh. & Optimierung & Neuronale Netze

David Münch

Universität Karlsruhe (TH)
Fakultät für Informatik
IBDS Prautzsch

29. Mai 2008

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/∼uhbro/

Tutorium 12: Donnerstags 8:00 Uhr - Raum -120
Tutorium 13: Donnerstags 9:45 Uhr - Raum -120

Übungsblattabgabe Donnerstag.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben.
Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Abgabe in Zweiergruppe erlaubt und ausdrücklich erwünscht!

Schein / Übungsblätter

Nur die mit (**K**) gekennzeichneten Aufgaben sind abzugeben. Diese werden korrigiert und bewertet.
66% der Punkte aller mit (**K**) gekennzeichneten Aufgaben aller Übungsblätter sind notwendig, um einen Schein zu erhalten.

Abgabe meistens Donnerstags.

Abgabe in Zweiergruppe erlaubt und ausdrücklich erwünscht!

Um das Übungsteam zu unterstützen bitte folgendes Deckblatt verwenden:
`http://www.stud.uni-karlsruhe.de/~unbdh/deckblatt/index.php?course=5`

Literatur

Boehm, Prautzsch: Numerical Methods. AK Peters 1993. ISBN 3-528-06350-5
`http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=7319953&nd=3204657`

Ash: Information Theory. Dover 1990. ISBN 0-486-66521-6
`http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&nd=9866904`

Goos: Vorlesungen über Informatik. Bd. 4, Springer 1998. ISBN 3-540-60650-5
`http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=9316367&nd=6568301`

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- Evolutionäre Algorithmen verstehen

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- Evolutionäre Algorithmen verstehen
- Stochastische Optimierungsalgorithmen kennen lernen

Was wollen wir heute erreichen?

- Übungsblatt 5 besprechen
- Evolutionäre Algorithmen verstehen
- Stochastische Optimierungsalgorithmen kennen lernen
- Wiederholung für die Hausaufgabenklausur

Aufgabe 25

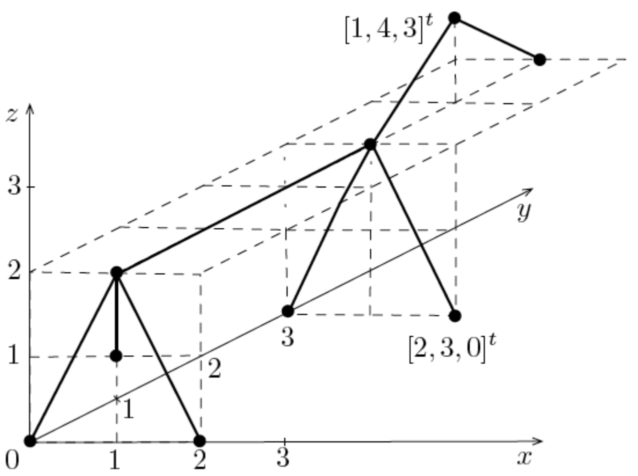
Berechnen Sie die multidimensionale Skalierung der in der Abbildung gegebenen Figur mit der Stressfunktion

$$F(f, \mathbf{p}_1, \ldots, \mathbf{p}_9) = \sum_{i,j=1}^{9} (f(\delta_{ij}^2) - d_{ij}^2)^2$$

und mit Hilfe des Pool-Adjacent-Violator-Algorithmus und des Levenberg-Marquardt-Algorithmus, so dass F minimal ist, wobei δ_{ij} der geodätische Abstand zwischen $\mathbf{p}_i$ und $\mathbf{p}_j$ und d_{ij} der euklidische Abstand zwischen $\mathbf{p}_i$ und $\mathbf{p}_j$ und f eine monoton steigende Funktion ist.

1. Berechnen Sie die geodätischen Abstände δ_{ij}.
2. Führen Sie einmal den Pool-Adjacent-Violator-Algorithmus ausführlich durch.
3. Wie rechnen Sie den Gradienten, der beim Levenberg-Marquardt-Algorithmus benötigt ist?
4. Berechnen Sie die multidimensionale Skalierung mithilfe von Computereinsatz und geben Sie eine optimale Pose der Figur und den Funktionswert von F aus.

Aufgabe 25



Lösung

Algorithmus von MDS:

1. Berechne $\delta_{i,j}$, $\delta_{i,j}^2$ und sortiere $\delta_{i,j}^2$.
2. while (Abbruchbedingung nicht erfüllt)
   2.1. Halte $\mathbf{p}_i$ fest und minimiere F über f mit dem Pool-Adjacent-Violator-Algorithmus.
   2.2. Halte f fest und minimiere F über $\mathbf{p}_i$ mit z. B. dem Levenberg-Marquardt-Algorithmus.

Die Abbruchbedingung ist eine der folgenden Bedingungen:

- Maximale Anzahl von Iterationsschritten erreicht.
- $|F_{\text{alt}} - F_{\text{neu}}| < \delta$, wobei F_{alt} bzw. F_{neu} der Funktionswert von F vor bzw. nach einer Schleife ist und δ eine gegebene oder genügend kleine Toleranz ist.

Übungsblatt 5

k	1	2	3	4	5	6	7	8	9	10
i,j	1, 1	2, 2	3, 3	4, 4	5, 5	6, 6	7, 7	8, 8	9, 9	3, 4
$x_k(=\delta^2_{i,j})$	0	0	0	0	0	0	0	0	0	1
$y_k(=d^2_{i,j})$	0	0	0	0	0	0	0	0	0	1
k	11	12	13	14	15	16	17	18	19	20
i,j	4, 3	7, 8	8, 7	8, 9	9, 8	1, 4	4, 1	2, 4	4, 2	5, 7
$x_k(=\delta^2_{i,j})$	1	2	2	2	2	5	5	5	5	5
$y_k(=d^2_{i,j})$	1	2	2	2	2	5	5	5	5	5
k	21	22	23	24	25	26	27	28	29	30
i,j	7, 5	6, 7	7, 6	7, 9	9, 7	4, 7	7, 4	1, 3	3, 1	2, 3
$x_k(=\delta^2_{i,j})$	5	5	5	8	8	9	9	10.4721	10.4721	10.4721
$y_k(=d^2_{i,j})$	5	5	5	4	4	9	9	2	2	2
k	31	32	33	34	35	36	37	38	39	40
i,j	3, 2	5, 8	8, 5	6, 8	8, 6	3, 7	7, 3	4, 8	8, 4	1, 2
$x_k(=\delta^2_{i,j})$	10.4721	13.3246	13.3246	13.3246	13.3246	16	16	19.4853	19.4853	20
$y_k(=d^2_{i,j})$	2	11	11	11	11	10	10	17	17	4
k	41	42	43	44	45	46	47	48	49	50
i,j	2, 1	5, 6	6, 5	5, 9	9, 5	6, 9	9, 6	4, 5	5, 4	4, 6
$x_k(=\delta^2_{i,j})$	20	20	20	25.6491	25.6491	25.6491	25.6491	27.4164	27.4164	27.4164
$y_k(=d^2_{i,j})$	4	4	4	9	9	9	9	14	14	14
k	51	52	53	54	55	56	57	58	59	60
i,j	6, 4	1, 7	7, 1	2, 7	7, 2	3, 8	8, 3	4, 9	9, 4	3, 5
$x_k(=\delta^2_{i,j})$	27.4164	27.4164	27.4164	27.4164	27.4164	29.3137	29.3137	33.9706	33.9706	38.8885
$y_k(=d^2_{i,j})$	14	14	14	14	14	20	20	25	25	11
k	61	62	63	64	65	66	67	68	69	70
i,j	5, 3	3, 6	6, 3	1, 8	8, 1	2, 8	8, 2	3, 9	9, 3	1, 5
$x_k(=\delta^2_{i,j})$	38.8885	38.8885	38.8885	44.2262	44.2262	44.2262	44.2262	46.6274	46.6274	55.8328
$y_k(=d^2_{i,j})$	11	11	11	26	26	26	26	26	26	9
k	71	72	73	74	75	76	77	78	79	80
i,j	5, 1	2, 5	5, 2	1, 6	6, 1	2, 6	6, 2	1, 9	9, 1	2, 9
$x_k(=\delta^2_{i,j})$	55.8328	55.8328	55.8328	55.8328	55.8328	55.8328	55.8328	65.0361	65.0361	65.0361
$y_k(=d^2_{i,j})$	9	13	13	13	13	9	9	30	30	30
k	81									
i,j	9, 2									
$x_k(=\delta^2_{i,j})$	65.0361									
$y_k(=d^2_{i,j})$	30									

Wir führen den *Pool-Adjacent-Violator-Algorithmus* durch. Für die Liste $\{y_i\} = \{y_{a_1} \geq \ldots \geq y_{b_1} < y_{a_2} \geq \ldots \geq y_{b_2} < \ldots < y_{a_m} \geq \ldots \geq y_{b_m}\}$ fassen wir alle monoton fallenden Gruppen der y_i zusammen und berechnen den Mittelwert jeder Gruppe, also $d_i = b_i - a_i + 1,\ \bar{y}_i = \sum_{j=a_i}^{b_i} y_i / d_i$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d_i	9	2	4	10	6	6	6	4	8	2	6	8	6	4
$\bar{y}_i$	0	1	2	4.8	4.333	10.667	8.333	9	14	20	15.667	21.750	11.667	30

Wir definieren $y_{a_i} := \ldots := y_{b_i} := \bar{y}_i$ und führen das obige Verfahren iterativ durch.

i	1	2	3	4	5	6	7	8	9	10
d_i	9	2	4	16	12	4	8	8	14	4
$\bar{y}_i$	0	1	2	4.625	9.5	9	14	16.75	17.429	30

i	1	2	3	4	5	6	7	8	9
d_i	9	2	4	16	16	8	8	14	4
$\bar{y}_i$	0	1	2	4.625	9.375	14	16.75	17.429	30

3. Lösung: Wir führen den *Levenberg-Marquardt-Algorithmus* durch. Sei f fest. Wir definieren

$$G(\mathbf{p}_1, \ldots, \mathbf{p}_9) := F(f, \mathbf{p}_1, \ldots, \mathbf{p}_9) = \sum_{i,j=1}^{9} \left(f_{\sigma(i,j)} - (\mathbf{p}_i - \mathbf{p}_j)^2\right)^2.$$

Sei $g_{i,j} = \left(f_{k(i,j)} - (\mathbf{p}_i - \mathbf{p}_j)^2\right)^2$. Es gilt

$$\begin{aligned}
\frac{\partial G}{\partial \mathbf{p}_k} &= \frac{\partial G}{\partial \mathbf{p}_k} \sum_{i=k, j\neq k} g_{i,j} + \frac{\partial G}{\partial \mathbf{p}_k} \sum_{i\neq k, j=k} g_{i,j} + \frac{\partial G}{\partial \mathbf{p}_k} \sum_{i=j=k} g_{i,j} + \frac{\partial G}{\partial \mathbf{p}_k} \sum_{i\neq k, j\neq k} g_{i,j} \\
&= \sum_{j\neq k} 2\left(f_{\sigma(k,j)} - (\mathbf{p}_k - \mathbf{p}_j)^2\right)\left(-2(\mathbf{p}_k - \mathbf{p}_j)\right)^t \\
&\quad + \sum_{i\neq k} 2\left(f_{\sigma(i,k)} - (\mathbf{p}_i - \mathbf{p}_k)^2\right)\left(-2(\mathbf{p}_i - \mathbf{p}_k)\right)^t (-1) + \mathbf{0} + \mathbf{0} \\
&= 2\sum_{j\neq k} 2\left(f_{\sigma(k,j)} - (\mathbf{p}_k - \mathbf{p}_j)^2\right)\left(-2(\mathbf{p}_k - \mathbf{p}_j)\right)^t \\
&= -8\sum_{j\neq k} \left(f_{\sigma(k,j)} - (\mathbf{p}_k - \mathbf{p}_j)^2\right)(\mathbf{p}_k - \mathbf{p}_j)^t.
\end{aligned}$$

Daher erhalten wir den Gradienten

$$(J =)\nabla G = \left[\frac{\partial G}{\partial \mathbf{p}_1} \cdots \frac{\partial G}{\partial \mathbf{p}_9}\right] \in \mathbb{R}^{1\times 27}.$$

Mit der Gleichung

$$(J^t J + \mu)\mathbf{v} = J^t(0 - G(\mathbf{p}_1, \ldots, \mathbf{p}_9))$$

können wir jetzt den Differenzvektor $\mathbf{v}$ berechnen.

4. Lösung: Mit einem Programm und mit dem Dämpfungsfaktor μ zwischen 0.1 und 10^{15} sind die Funktionswerte $F(\mathbf{p} + \mathbf{v})$ immer größer als $F(\mathbf{p})(\approx 1404.428571)$, wobei $\mathbf{p} = (\mathbf{p}_1, \ldots, \mathbf{p}_9)$ ist. Daher ist $\mathbf{p}$ schon eine lokale optimale Lösung.
Da F durch den Levenberg-Marquardt-Algorithmus nicht verbessert wird, steht $\mathbf{p}$ bereits für eine lokale optimale Pose. Der optimale Wert von F beträgt 1404.428571.

Aufgabe 26

Die XOR-Funktion („entweder-Oder“, logische Antivalenz) ist nicht linear separierend, d. h. sie kann von keinem einfachen Perzeptron berechnet werden.

Konstruieren Sie ein Perzeptron, das auch Zwischenschichten (die weder Ein- noch Ausgabe-Neuronen enthalten) haben kann,[1] das die n-wertige Funktion

$$\Pi : \{0,1\}^n \to \{0,1\}$$
$$(a_1, \ldots, a_n) \mapsto \Pi(a_1, \ldots, a_n) = \left(\sum_{i=1}^{n} a_i\right) \bmod 2 = \frac{1}{2}\left(1 - \prod_{i=1}^{n} (-1)^{a_i}\right)$$

realisiert und aus $\{0,1\}$-Neuronen aufgebaut ist.

1. Versuchen Sie eine Lösung zu finden, die mit möglichst wenigen Neuronen auskommt.
   Wie viele Neuronen benötigen Sie? Wie viele Zeitschritte benötigt das Netz zum Berechnen der Funktion?
2. Versuchen Sie eine Lösung zu finden, die das Ergebnis möglichst schnell berechnet.
   Wie viele Neuronen benötigen Sie? Wie viele Zeitschritte benötigt das Netz zum Berechnen der Funktion?
3. Beweisen Sie die Identität
   $$\left(\sum a_i\right) \bmod 2 = \frac{1}{2}\left(1 - \prod (-1)^{a_i}\right)$$
   für $a_i \in \mathbb{N}$.

Aufgabe Simplex-Algorithmus

minimiere: $-x_1 - 5x_2$

unter den Nebenbedingungen:

$-x_1 + 3x_2 \leq 10$

$2x_1 - x_2 \leq 6$

$x_1, x_2 \geq 0$

Beginne mit $(0,0)^t$.

Aufgabe Ausgleichsgerade

Berechne die Ausgleichsgerade für die Punkte:
(0,0), (2,1), (2,4), (4,3), (2,2).
Hinweis: Die Gleichung $x^2 - 18x = -44$ hat die beiden Lösungen:
$x_1 = 2,91$ und $x_2 = 15,08$

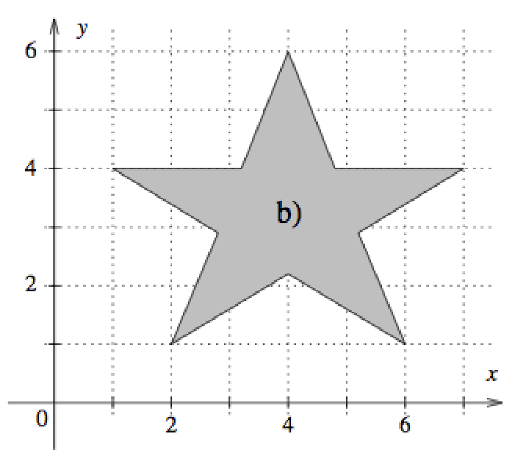
Aufgabe Perzeptron

Ein Perzeptron soll Punkte im Innern geometrischer Figuren, hier: Polygone, als solche korrekt klassifizieren.

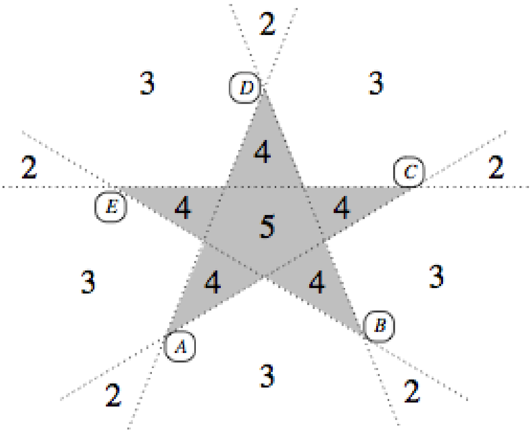
Dazu stehen binäre $\{0, 1\}$-Neuronen zur Verfügung mit der Eigenschaft, daß ihre Ausgabe q *undefiniert*[1] ist, wenn der Erregungszustand $\sum w_i q_i$ genau mit dem Schwellwert b übereinstimmt. Wie üblich sei für $\sum w_i q_i > b$ die Ausgabe $q = 1$, und $q = 0$ für $\sum w_i q_i < b$.

Geben Sie ein neuronales Netz – mit sämtlichen Gewichten und Schwellwerten – an, das genau die Figur b) erkennt. Benötigen Sie mehr als 6 Neuronen?

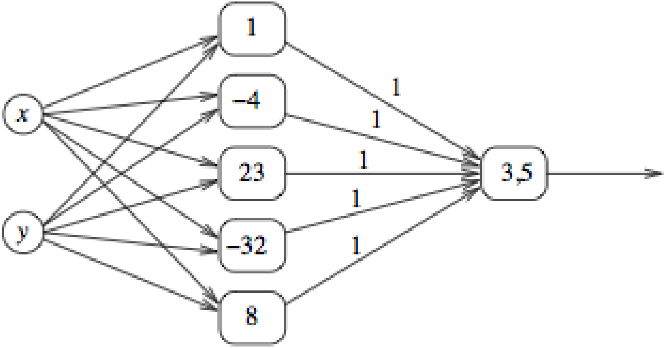
Aufgabe Perzeptron



Anzahl erregter Neuronen

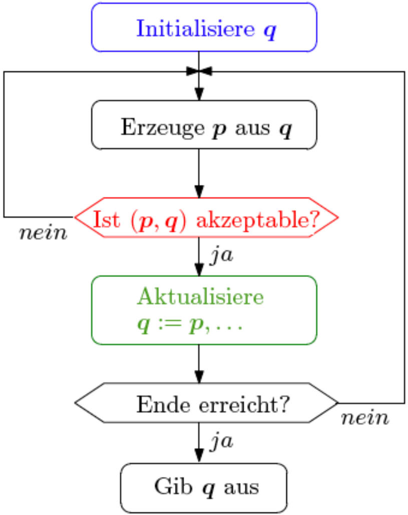


Lösung



Stochastische Verfahren

Zufallsgesteuerte Optimierung

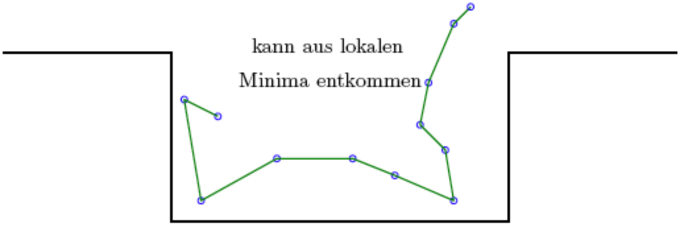


Algorithm 1 SIMULIERTES TEMPERN

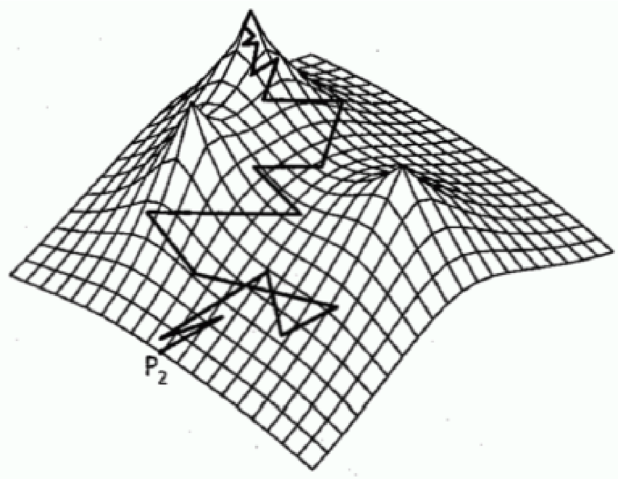
input : initialer Zustand q_0, initiale Temperatur T_0, Kostenfkt. c
output: beste Lösung

$q \leftarrow q_0$, $T \leftarrow T_0$
while *Stopp-Kriterium nicht erfüllt* **do**
    while *noch nicht im Gleichgewicht* **do**
        $q' \leftarrow$ irgendein zufälliger, benachbarter Zustand zu q;
        $\Delta c = c(q') - c(q)$;
        $Probability \leftarrow min(1, e^{\frac{-\Delta c}{T}})$;
        if $random(0,1) \leq Probability$ **then**
            $q \leftarrow q'$;
        end
    end
    Aktualisiere Temperatur T;
end
Gib die beste Lösung aus.

Simuliertes Tempern



Simuliertes Tempern



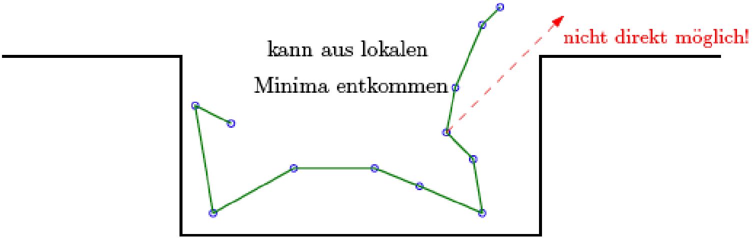
Schwellwertalgorithmus

Nachfolger p von q wird akzeptiert, wenn

$$c(p) \leq c(q) + \sigma$$

wobei σ der Schwellwert ist, welcher regelmässig abgesenkt wird.

Schwellwertalgorithmus



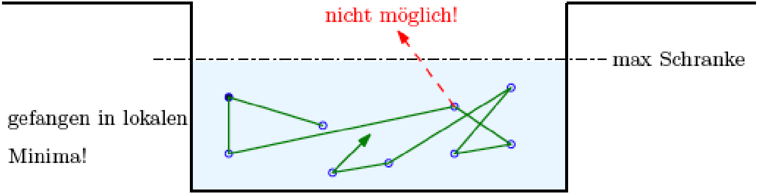
Sintflut-Algorithmus

Nachfolger p von q wird akzeptiert, wenn

$$c(p) \leq H$$

wobei H die Sintflutgrenze ist, die regelmässig abgesenkt wird.

Sintflut-Algorithmus



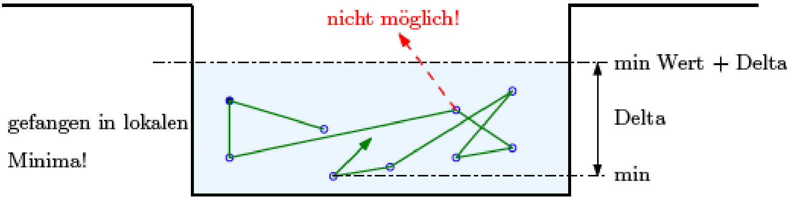
Rekordjagt-Algorithmus

Es wird die beste Bewertung gespeichert ($record$). Diese darf nur einen bestimmten Wert δ überschritten werden.
Nachfolger p von q wird akzeptiert, wenn

$$c(p) \leq record + \delta$$

wobei der Maximalwert δ regelmässig abgesenkt wird.

Rekordjagt-Algorithmus



Zusammenfassung

Zusammenfassung

- Schwellwert-Verfahren und Simuliertes Tempern können aus kleinen, aber tiefen lokalen Minima entkommen, die anderen Verfahren nicht. Aber bei grösser werdenden Problemen werden die lokalen Minima weniger markant und tief.

Zusammenfassung

- Schwellwert-Verfahren und Simuliertes Tempern können aus kleinen, aber tiefen lokalen Minima entkommen, die anderen Verfahren nicht. Aber bei grösser werdenden Problemen werden die lokalen Minima weniger markant und tief.
- Sintflut- und Schwellwert-Verfahren sind dem Simulierten Tempern überlegen.

Zusammenfassung

- Schwellwert-Verfahren und Simuliertes Tempern können aus kleinen, aber tiefen lokalen Minima entkommen, die anderen Verfahren nicht. Aber bei grösser werdenden Problemen werden die lokalen Minima weniger markant und tief.
- Sintflut- und Schwellwert-Verfahren sind dem Simulierten Tempern überlegen.
- Der Sintflut-Algorithmus ist fast genauso gut wie das Schwellwert-Verfahren, nicht ganz so stabil in der Qualität der Lösung, dafür jedoch etwas schneller.

Zusammenfassung

- Schwellwert-Verfahren und Simuliertes Tempern können aus kleinen, aber tiefen lokalen Minima entkommen, die anderen Verfahren nicht. Aber bei grösser werdenden Problemen werden die lokalen Minima weniger markant und tief.
- Sintflut- und Schwellwert-Verfahren sind dem Simulierten Tempern überlegen.
- Der Sintflut-Algorithmus ist fast genauso gut wie das Schwellwert-Verfahren, nicht ganz so stabil in der Qualität der Lösung, dafür jedoch etwas schneller.
- Das Schwellwert-Verfahren ist die Methode der Wahl, also vorzuziehen.

Zusammenfassung

- Schwellwert-Verfahren und Simuliertes Tempern können aus kleinen, aber tiefen lokalen Minima entkommen, die anderen Verfahren nicht. Aber bei grösser werdenden Problemen werden die lokalen Minima weniger markant und tief.
- Sintflut- und Schwellwert-Verfahren sind dem Simulierten Tempern überlegen.
- Der Sintflut-Algorithmus ist fast genauso gut wie das Schwellwert-Verfahren, nicht ganz so stabil in der Qualität der Lösung, dafür jedoch etwas schneller.
- Das Schwellwert-Verfahren ist die Methode der Wahl, also vorzuziehen.
- Wenn man ein reichhaltiges Sortiment von Veränderungsschritten zulässt, wird auch besser optimiert.

Evolutionäre Algorithmen

Es wird versucht das Evolutionsprinzip der Biologie nachzubilden. Die Elemente $q \in Q$ sind die Induviduen einer Population. Dabei besitzt jedes Individuum:

- einen Genotyp (Merkmalsvektor) $k \in (R)^n$ und
- einen Phänotyp (Bewertung). (Fitnessfunktion $fit(q)$.

Individuen können sich fortpflanzen und dabei ihren Genotyp verändern (Mutation). Bei einem Elter: Klon; bei mehreren Eltern: Kreuzung.

Ein Individuum wird durch das Paar (k, σ) repräsentiert.

Ein Individuum wird durch das Paar (k, σ) repräsentiert.
σ ist der Streuungsvektor, der dir maximale Veränderung bei der Mutation angibt. Kann auch geändert werden.

Ein Individuum wird durch das Paar (k, σ) repräsentiert.
σ ist der Streuungsvektor, der dir maximale Veränderung bei der Mutation angibt. Kann auch geändert werden.
Nachkommen mutieren nach der Regel: $k^{t+1} = k^t + d$, wobei die Koordinaten von d normalverteilt sind mit dem Streuungsvektor σ.

Plus-Strategie $(\mu + \lambda)$

Aus den μ Eltern werden λ Nachkommen geklont (Eltern werden gleichverteilt unter den μ Individuen gewählt, jedes Individuum kann auch mehrmals Elter werden)
Die μ Individuen mit der besten Fitness-Funktion bilden die nächste Generation (Eltern können also überleben)
Unter μ Eltern und λ Nachkommen werden die μ stärksten (u.a. mit $fit(q)$) ausgewählt, die anderen λ sterben.

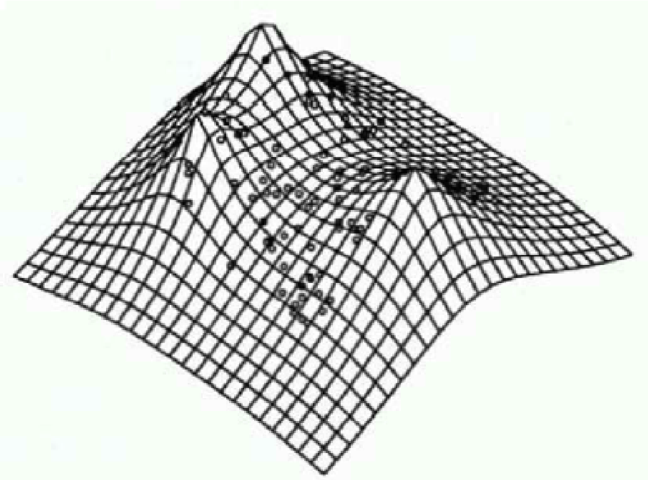
Plus-Strategie $(\mu + \lambda)$

Aus den μ Eltern werden λ Nachkommen geklont (Eltern werden gleichverteilt unter den μ Individuen gewählt, jedes Individuum kann auch mehrmals Elter werden)
Die μ Individuen mit der besten Fitness-Funktion bilden die nächste Generation (Eltern können also überleben)
Unter μ Eltern und λ Nachkommen werden die μ stärksten (u.a. mit $fit(q)$) ausgewählt, die anderen λ sterben.

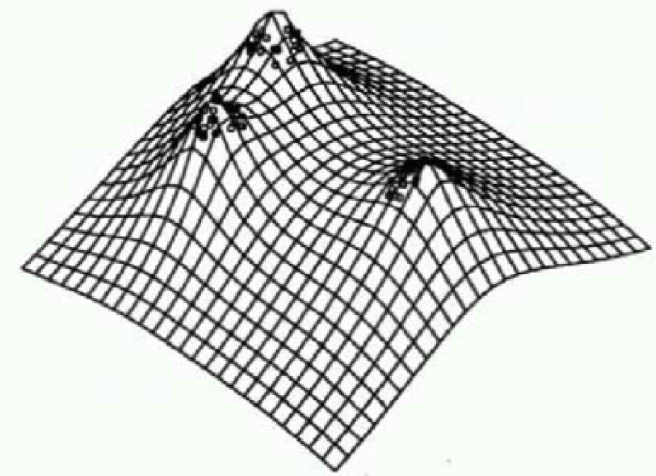
Komma-Strategie (μ, λ)

Aus den μ Eltern werden λ Nachkommen geklont (Eltern werden gleichverteilt gewählt, jedes Individuum kann auch mehrmals Elter werden)
Die Eltern überleben nicht, es werden nur unter den Nachkommen die μ stärksten ausgesucht.

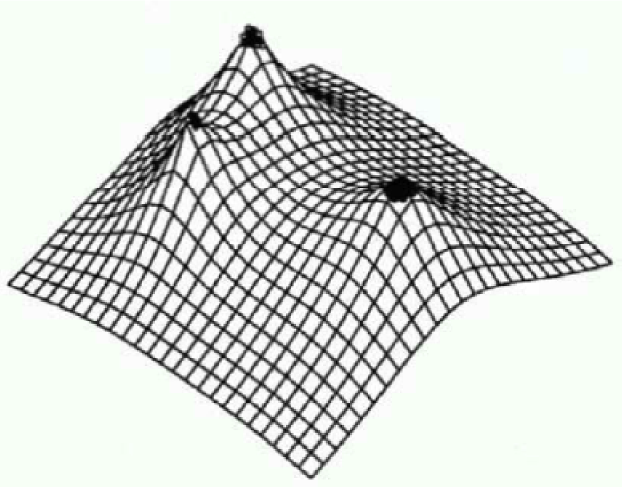
ES(150+1000), Generation 1



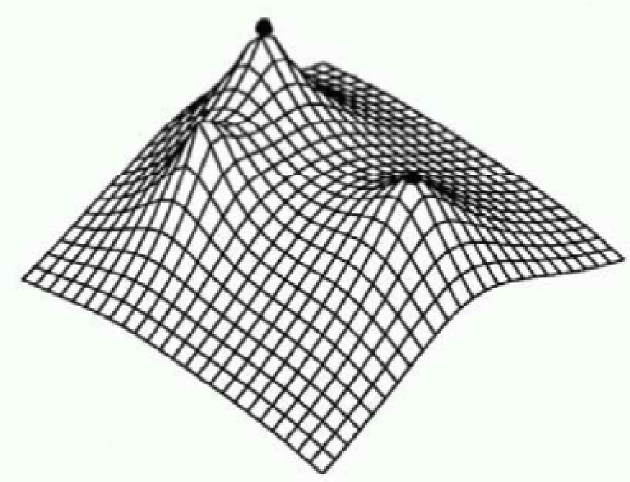
ES(150+1000), Generation 2



ES(150+1000), Generation 3



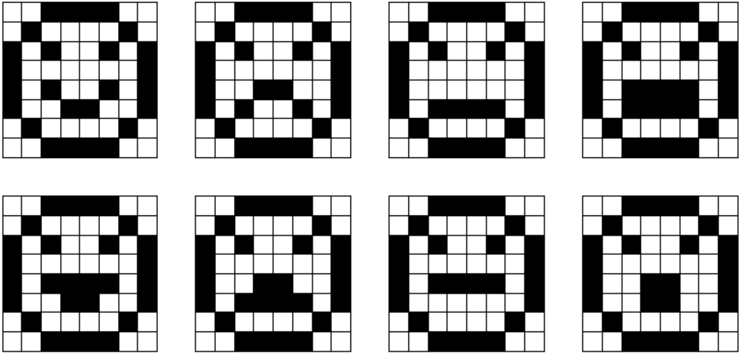
ES(150+1000), Generation 6



Bonusaufgabe

Betrachten Sie die hier aufgeführten Emotionen als 8×8 Pixel Bilder die jeweils die Werte 1 (Schwarz) und 0 (Weiss) annehmen können.
Entwerfen Sie ein neuronales Netz, das in der Lage ist zwischen den vier verschiedenen Emotionen zu unterscheiden.
Beschreiben Sie die Eingabe, Ausgabe des Netzes und geben Sie eine Topologie des Netzes mit möglichst wenig Neuronen an.
Geben Sie ausserdem die Gewichte sowie Schwellwerte der Neuronen an. Gehen Sie dabei von der binären Sprungfunktion als Aktivierungsfunktion für die Neuronen aus.

Zufallsgesteuerte Optimierung



Lachend Traurig Neutral Erstaunt

Noch Fragen?

Vorschau

Vorschau

- Lernen von Neuronalen Netzen

Vorschau

- Lernen von Neuronalen Netzen
- Hopfield-Netze

Vorschau

- Lernen von Neuronalen Netzen
- Hopfield-Netze
- Informationstheorie

Viel Erfolg bei der Hausaufgabenklausur!

Bis zum nächsten Mal