Programmieren in R am Beispiel des Maximum Entropy Sampling

Albrecht Gebhardt[1] Claudia Gebhardt[2]

[1] Institut für Statistik
Universität Klagenfurt
Universitätsstr. 65-67, A 9020 Klagenfurt
albrecht.gebhardt@uni-klu.ac.at

[2] Universitätsbibliothek
Universität Klagenfurt
Universitätsstr. 65-67, A 9020 Klagenfurt
claudia.gebhardt@uni-klu.ac.at

Bad Doberan, Oktober 2009

Übersicht

1 Maximum Entropy Sampling

2 R Library edesign

3 Implementationsdetails

4 Anwendungsbeispiel

5 Literatur

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

Voraussetzungen

Ausgangssituation

Gegeben sei eine stationäre räumliche Zufallsvariable mit isotroper Kovarianzfunktion die folgendem Modell genüge:
$Z(\underline{x}) = \theta^\top \underline{f}(\underline{x}) + \varepsilon$, $\mathbb{E}\varepsilon = 0$, $Cov(Z(\underline{x})), Z(\underline{x} + \underline{h}) = C(|\underline{h}|)$
Weiters gegeben:

- Messpunktnetz mit n_F festen Stationen $\underline{x}_F \in X_F$ (fixed points),
- n_E auswählbare Kanditaten $\underline{x_E} \in X_E$ für neue Messpunkte (eligible points),
- Kovarianzfunktion C der zu beobachtenden Zielgröße Z.

UNIVERSITÄT KLAGENFURT
Institut für Statistik

Messpunktplanung

Aufgabe

Auswahl von n_s Punkten aus der Kandidatenmenge, so dass das Entropiekriterium

$$\log \det \mathbf{C}$$

für das Messnetz der $n_F + n_S$ festen und ausgewählten Messpunkte maximal wird.
C - Kovarianzmatrix des resultierenden Messpunktnetzes

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

Theorie

Entropiekriterium

Maximierung der Entropie einer Stichprobe von einer Verteilung mit Dichte f:

$$\text{Ent}(f) = -\int f(x) \log f(x)\, dx \rightarrow max$$

Anwendung auf endliche Menge X_E der eligible points bedeutet Entropiezerlegung (X_S – selected points):

$$const = \text{Ent}(X_E) = \text{Ent}(X_S) + \mathbb{E}_{X_S}\text{Ent}(\overline{X_S} \mid X_S)$$

Möglichst große Entropie für die Auswahl X_S:

$$\text{Ent}(X_S) \rightarrow max$$

Theorie

Entropiekriterium

Struktur der Kovarianzmatrix:

$$\mathbf{C}_{X_E} = \begin{bmatrix} \mathbf{C}_{X_S} & \mathbf{C}_{X_S,\overline{X_S}} \\ \mathbf{C}_{\overline{X_S},X_S} & \mathbf{C}_{\overline{X_S}} \end{bmatrix}$$

für normalverteilte Zufallsgröße Z ergibt sich mit

$$\log\det \mathbf{C}_{X_E} = \log\det \mathbf{C}_{X_S} + \log\det\left(\mathbf{C}_{\overline{C_{X_S}}} - \mathbf{C}_{\overline{X_S},X_S}\mathbf{C}_{X_S}^{-1}\mathbf{C}_{X_S,\overline{X_S}}\right)$$

das eingangs vorgestellte Kriterium $\log\det \mathbf{C}_{\mathbf{X_S}} \rightarrow max$

Institut für Statistik

Theorie

Exakte Lösung der Optimierungsaufgabe

- Ko. et.al gibt einen Branch-and-Bound Algorithmus zu Lösung der Optimierungsaufgabe an
- Grundlage: Obere Schranken (interlacing property der Eigenwerte) für die Entropie des Messnetzes, Obere Schranken werden zur Konstruktion eines B&B Algoritmus benötigt
- ... liefern zusätzlich relative Fehlerschranken

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

R Implementation

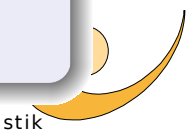
library(edesign) enthält:

- Drei Varianten eine Startlösung zu bestimmen:
  - greedy Algorithmus (Beginne mit n_F festen Punkten, füge den jeweils optimalen nächsten auswählbaren Punkt hinzu bis das Messnetz den Umfang $n_F + n_S$ erreicht), in R: `greedy(...)`
  - dualer greedy Algorithmus (Beginne mit allen $n_F + n_E$ Punkten, entferne schrittweise den jeweils schlechtesten der auswählbaren Punkte bis das Messnetz den Umfang $n_F + n_S$ erreicht), in R: `dualgreedy(...)`
  - interchange Algorithmus (verbessere eine zulässige Lösung durch Entfernen des schlechtesten und Hinzufügen des besten auswählbaren Punkts, in R: `interchange(...)`
- Exakte Lösung des Optimierungsproblems durch Anwendung eines Branch-and-Bound Verfahrens (siehe Ko et.al.), Abbruch auch durch schrittweise adaptierte relative Fehlerschranken möglich, in R: `maxentropy(...)`

Funktion `maxentropy()`

Parameter für `maxentropy()` (B&B)

- Kovarianzmatrix C (pos. definit) für die Menge aller Punkte: $C \in \mathbb{R}_{>}^{(n_F+n_E)\times(n_F+n_E)}$
- n_F - Anzahl der festen Punkte, (entprechen den ersten n_F Zeilen/Spalten von C, d.h. die letzten n_E Zeilen/Spalten von C entsprechen den auswählbaren Punkten)
- n_S - Anzahl der auszuwählenden Punkte
- optional: Startlösung (sonst automatisch bestimmt)
- optional: Methode (greedy/dual greedy) zur Bestimmung der Startlösung



Funktion `maxentropy()`

Rückgabewerte von `maxentropy()`

- optimaler Wert des Determinantenkriteriums.
- Indexvektor der ausgewählten Punkte.
- Zusatzinformation (Anz. der Iterationen, Speicherkonsum, ...)

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

Beispielaufruf maxentropy()

maxentropy()

```
> x <- c(0.97900601, 0.82658702, 0.53105628, 0.9142019,
 .... [TRUNCATED, 20 Werte]
> y <- c(0.36545512, 0.72144122, 0.95688671, 0.25422154,
  .... [TRUNCATED, 20 Werte]
# Konstruktion einer pos. definiten Matrix C (siehe Ko e
> C <- outer(x, x, "-")^2 + outer(y, y, "-")^2
> C <- (2 - C)/10
> diag(C) <- 0
> diag(C) <- 1/20 + apply(C, 2, sum)

# frei gewaehlte Startloesung
# Punkte 1-5 fest, Punkte 6-20 auswaehlbar,
# 5 ausgewaehlt, 0-en markieren auswaehlbare Punkte:
> S.entrp <- c(0, 7, 0, 9, 0, 11, 0, 13, 14, 0, 0, 0,
    0, 0, 0)
```

Maximum entropy sampling with R

```
# B-and-B Lsg: 20x20 Matrix, 5 Punkte fest,
# 5 der restlichen 15 sind auszuwaehlen:
maxentropy(C, 5, 5, S.start = S.entrp)
...

$S.start
 [1]  0  7  0  9  0 11  0 13 14  0  0  0  0  0  0
$det.start
[1] 76351.18
$S
 [1]  0  0  8  9  0  0  0  0  0  0 16  0 18 19  0
$opt
[1] 134444.8
$maxcount
[1] 85
$iter
[1] 194
```

ALPEN – ADRIA UNIVERSITÄT KLAGENFURT
Institut für Statistik

Funktionen für (dual) greedy Algorihmus

Parameter für `greedy()` und `dualgreedy()`

analog zu `maxentropy()`:

- Kovarianzmatrix C (pos. definit) für die Menge aller Punkte: $C \in \mathbb{R}_{>}^{(n_F+n_E)\times(n_F+n_E)}$
- n_F - Anzahl der festen Punkte, (entprechen den ersten n_F Zeilen/Spalten von C, d.h. die letzten n_E Zeilen/Spalten von C entsprechen den auswählbaren Punkten)
- n_S - Anzahl der auszuwählenden Punkte

Rückgabewert von `greedy()` und `dualgreedy()`

- erreichter Wert des Determinantenkriteriums,
- Indexvektor der ausgewählten Punkte.

Funktionen für (dual) greedy Algorihmus

Beispielaufruf `greedy()` und `dualgreedy()`

```
#  C: 20x20 Matrix, 5 feste Punkte, 5 aus 15 wählen:

> greedy(C,5,5)

$S
 [1]  0  0  8  9  0  0  0  0  0  0 16  0 18 19  0

...
> dualgreedy(C,5,5)
$S
 [1]  0  0  8  9  0  0  0  0  0  0 16  0 18 19  0

...
```

(Man beachte: die Heuristiken finden hier schon das Optimum exakt!)

library(edesign)

Anmerkungen

- Die Algorithmen sind in C und Fortran implmentiert.
- Obige Beispiele sind Teil der jeweiligen Funktionsdokumentation.
- `greedy()` und `dualgreedy()` liefern in diesem (einfachen) Beispiel schon die exakte Lösung.
- Greedy und dual greedy sind wesentlich schneller und weniger speicherinterinsiv als der exakte Branch-and-Bound Algorithmus.
- $\Rightarrow$ `greedy()` und `dualgreedy()` liefern auch dann noch (zumindest Näherungs-) Lösungen wenn B&B fehlschlägt (zu lange Rechenzeit, zuviel Speicherbedarf)

R Programmierung am Beispiel edesign

Verzeichnisstruktur einer R Library

```
alge@aleph:~/edesign$ ls -l # (gekuerzt)
-rw- ...  DESCRIPTION  # Beschreibung, Name, Author,
                       # Abhaengigkeiten!, usw.
-rw- ...  INDEX        # Index der Dokumentation
drwx ...  man          # Dokumentation
drwx ...  R            # R Files
drwx ...  src          # C/Fortran Files
-rw- ...  TITLE        # Kurztitel
```

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

R Programmierung am Beispiel edesign

C Routinen

z.B. B&B Hauptroutine in `src/entrp.c`:

```
void entrp(double *A, int *lda, int *na, int *nf,
  int *ne, int *ns, int *S, double *opt, int *S_Work,
    ...){
  ...
}
```

Beachte:

- Argumente sind Zeiger, Rückgabetyp ist `void`, Ein- UND Ausgabe erfolgt ausschließlich über Zeigervariablen.
- Hilfsarrays werden ebenfalls über Zeigervariablen übergeben (und von R angelegt)

R Programmierung am Beispiel edesign

Fortran Routinen

z.B. greedy Routine in `src/grd.f`:

```
      SUBROUTINE GRD(A,LDA,NA,NF,NE,NS,S,OPT,IND, ...)

      IMPLICIT NONE

      INTEGER  NA, NF, NE, NS, IERR, LDA, ...
      DOUBLE PRECISION   OPT, ...
      DOUBLE PRECISION   A(LDA,*), ...
      INTEGER S(*), IND(*) ...

      ...

      RETURN
      END
```

R Programmierung am Beispiel edesign

Fortran Routinen ...

Beachte:

- Nur SUBROUTINE möglich, Ein- und Ausgabe nur über Argumentliste, keine FUNCTION!
- kein dynamischer Speicher (Fortran), muss in R vorher alloziert werden.

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

R Programmierung am Beispiel edesign

Verwendung der C API: .C

```
maxentropy <- function(A,nf,ns,
                       method="d",S.start=NULL, ... )
  {
    ... Vorbereitungen, Checks ...

    ans.entrp <- .C("entrp",
      A=as.double(A), lda=as.integer(na),
      na=as.integer(na), nf=as.integer(nf),
      ne=as.integer(ne), ns=as.integer(ns),
      S=as.integer(S.start),
      opt=as.double(det.start),
      integer(ne), # S_Work
      ..., PACKAGE="edesign") ...
}
```

R Programmierung am Beispiel edesign

Verwendung der Fortran API: .Fortran

```
greedy <- function(A,nf,ns, ... )
  {
    ... Vorbereitungen, Checks ...
   ans<-.Fortran("grd",
     A=as.double(A), lda=as.integer(na),
     na=as.integer(na), nf=as.integer(nf),
     ne=as.integer(ne), ns=as.integer(ns),
     S=integer(ne), det=double(1),
     integer(na), # ind
     ..., PACKAGE="edesign")
   ...
}
```

R Programmierung am Beispiel edesign

Potentielle Fehlerquellen bei Verwendung der C/Fortran API

Beachte:

- Reihenfolge der C/Fortran Argumente einhalten,
- Typumwandlungen erzwingen,
- Speicher für Rückgabe- und Hilfsvariablen allozieren
- Vermeiden von Namenskonflikten (gleichnamige Routinen anderer Packages), deshalb `PACKAGE="edesign"` als letzter Parameter von `.C` und `.Fortran`

UNIVERSITÄT KLAGENFURT
Institut für Statistik

R Programmierung am Beispiel edesign

Optional: korrekte Funktionsdeklarationen

in src/edesign.h:

```
#include "R.h" /* for F77_NAME */
void entrp(double*, int*, ...);
extern void F77_NAME(grd)(double*, int*, ...);
...
```

F77_NAME beschreibt das jeweilige C↔Fortran Interface (Underscore am Namen angehängt (gcc) o.ä.)

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

R Programmierung am Beispiel edesign

Optional: Deklaration des Fortran Interfaces

in src/init.c:

```
#include <R.h>
#include <Rinternals.h>
#include "edesign.h"
#include <R_ext/Rdynload.h>
/* Fortran interface descriptions: */
static R_NativePrimitiveArgType grd_t[19] = {
    REALSXP, /* A   */
    INTSXP,  /* LDA */
    INTSXP,  /* NA  */    ...
}
static R_FortranMethodDef fortranMethods[] = {
    {"grd", (DL_FUNC) &F77_SUB(grd), 19, grd_t}, ...
    {NULL, NULL, 0} };
```

R Programmierung am Beispiel edesign

Optional: Deklaration des C Interfaces

weiter in src/init.c:

```
...
/* C interface descriptions: */
static R_NativePrimitiveArgType entrp_t[29] = {
    REALSXP, /* A   */
    INTSXP,  /* lda */
    INTSXP,  /* na  */    ...
};
static R_CMethodDef cMethods[] = {
    {"entrp", (DL_FUNC) entrp, 29, entrp_t}, ...
    {NULL, NULL, 0} };
```

R Programmierung am Beispiel edesign

Optional: Registrieren der exportierten Funktionen

weiter in `src/init.c`:

```
...
void R_init_edesign(DllInfo *info)
{
  R_registerRoutines(info,
                     cMethods,
                     NULL /*callMethods*/,
                     fortranMethods,
                     NULL /*externalMethods*/);
}
```

Zweck: Überprüfung der korrekten Typisierung der C/Fortran Aufrufe.

R Programmierung am Beispiel edesign

LAPACK und BLAS Routinen

Korrekte Verwendung der in R schon enthaltenen Matrixroutinen:
Datei `src/Makevars`

```
## private CPP flags:
PKG_CPPFLAGS = # evtl. eigene CPPFLAGS setzen

## we use the BLAS and the LAPACK library:
PKG_LIBS = $(LAPACK_LIBS) $(BLAS_LIBS) $(FLIBS)

MkInclude = $(R_HOME)/etc${R_ARCH}/Makeconf
```

Damit sind Standardfunktionen von LAPACK und BLAS (z.b. Matrixmult. `DGEMM`, usw) sofort in C/Fortran verfügbar.

Beispiel

Vergleich verschiedener Sampling Strategien

siehe Baume, et. al 2009:
verwendete Optimalitätskriterien:

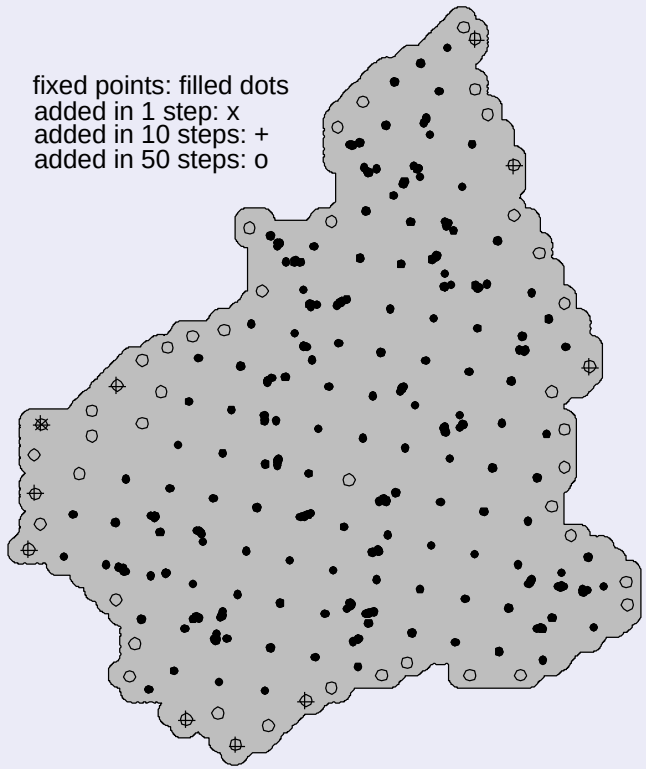
- Mittlere Vorhersagevarianz (ord. kriging), minimiert mit simlulated annealing und Greedy-Heuristiken
- Entropie, maximiert mit Greedy Heuristiken (Datensatz zu groß für exakten B&B Algorithmus)

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

Beispiel

Jura Datensatz + Max.-Entropie Greedy-Lsg.

- Bodenminerale, bestimmt an Gitter + zusätzlichen Verdichtungspunkten,
- Aufgabe: Füge 1, 10, 50 aus ca 3000 Punkten hinzu:

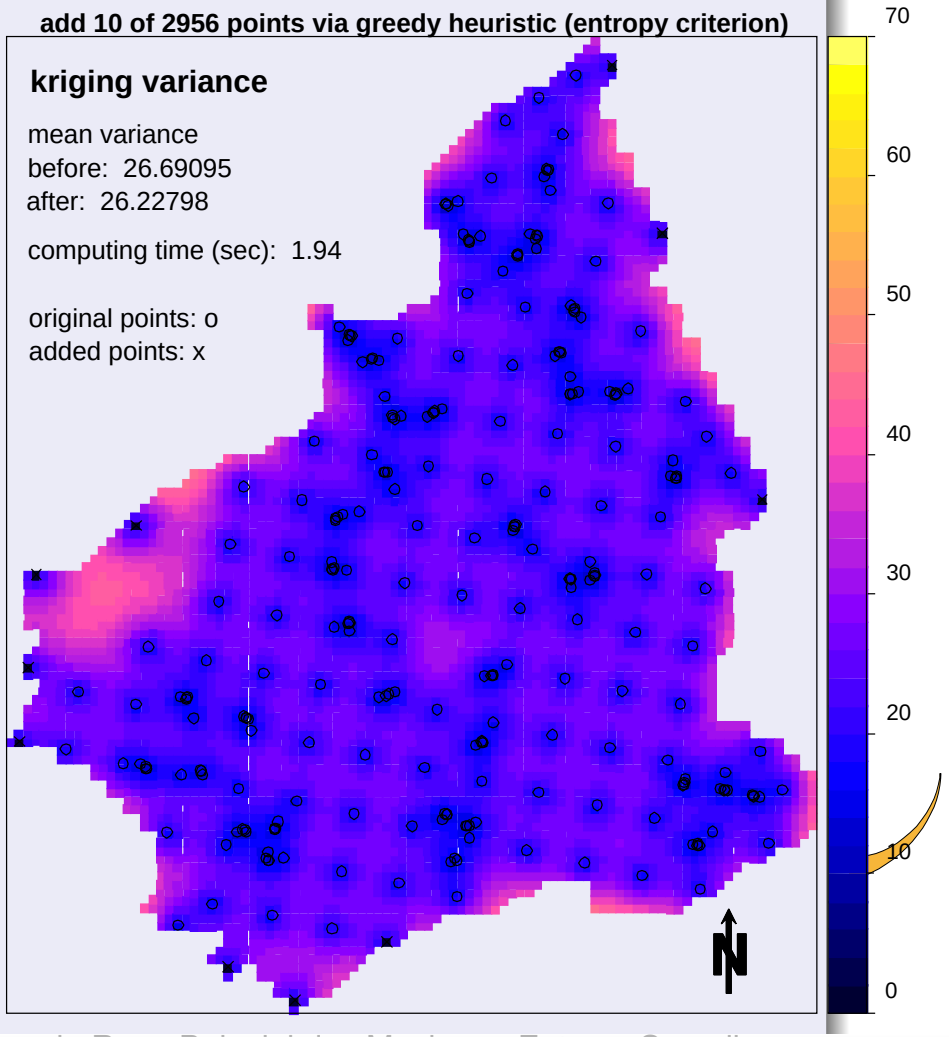


points added by greedy heuristic, entropy criterion

Beispiel

Jura Datensatz, Verbesserung der Krigevarianzen

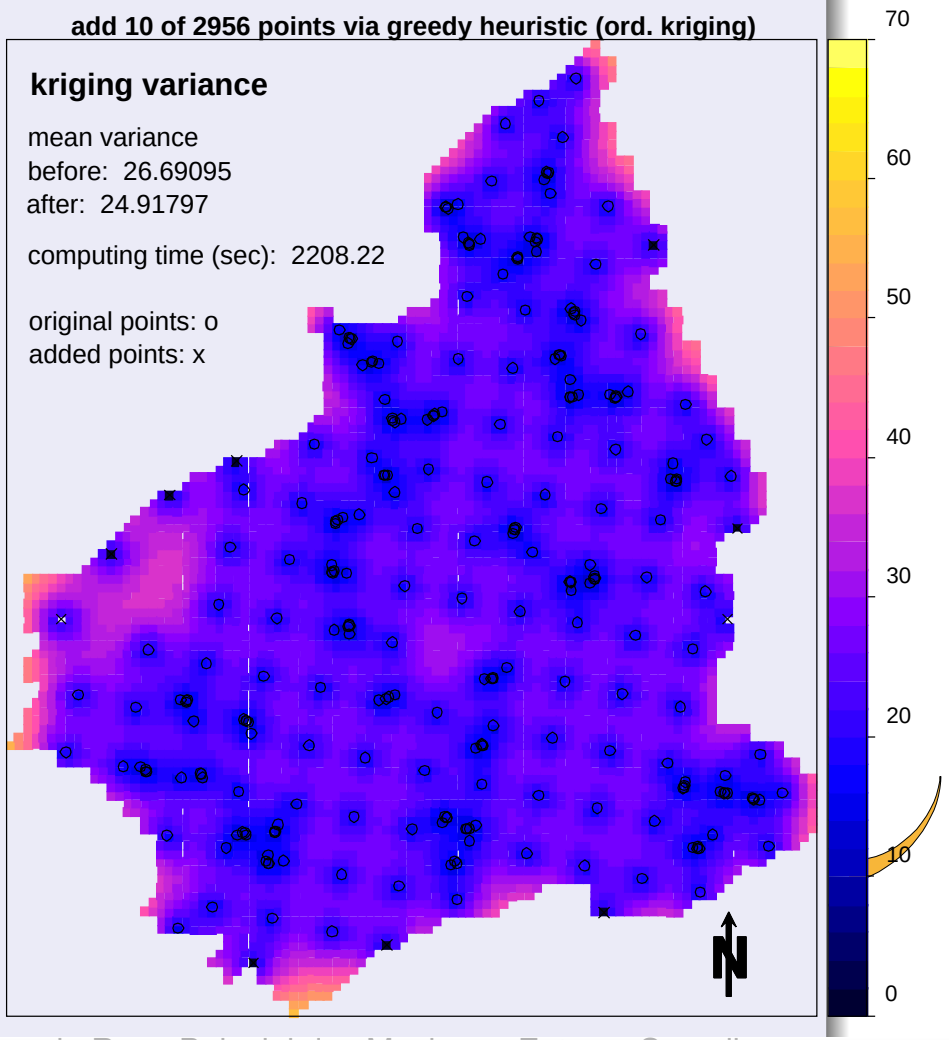
- Maximum Entropie - Greedy Lösung für 10 Punkte
- anschließend Krigevarianzen bestimmen



Beispiel

Jura Datensatz, Verbesserung der Krigevarianzen

- Greedy Lösung für Mittlere Krigevarianz für 10 Punkte:
- größere Varianzverbesserung als bei Max.-Entropie, aber immens höhere Rechenzeit (Varianzen müssen pro Schritt auf vollständigem Gitter bestimmt werden)
- kaum sichtbare Unterschiede im resultierenden Messnetz.



Beispiel

Zusammenfassung

- B&B Algorithmus für große Aufgabenstellungen (10 aus 3000) nicht anwendbar (10 aus 50 wäre noch ok)
- greedy und dual greedy Heuristik schneller als simulated annealing
- Entropie Kriterium rechentechnisch attraktiver als Mittlere Vorhersagevarianz, trotzdem vergleichbare Ergebnisse

ALPEN – ADRIA
UNIVERSITÄT
KLAGENFURT
Institut für Statistik

Literatur

Literature / Links

[1] Ko, Lee, Queyranne, An exact algorithm for maximum entropy sampling, Operations Research 43 (1995), 684-691.
[2] C. Gebhardt, Bayesian Methods for Geostatistical Design, PhD Thesis, Univ. Klagenfurt, 2003
[3] O. Baume, A. Gebhardt, C. Gebhardt, G. Heuvelink, J. Pilz, Network optimization algorithms and scenarios in the context of automatic mapping, StatGIS 2009, Milos, Greece
[5] CRAN, Writing R Extensions, `http://cran.r-project.org/doc/manuals/R-exts.html`, 2009
[4] `ftp://ftp.uni-klu.ac.at/pub/R/contrib` for edesign (formerly named entropy) and other R packages