Formelsammlung

für Algorithmik kontinuierlicher Systeme

Singulärwertzerlegung

Bestimmung der SVD

$A = USV^T$ (dim(A) = m×n)

mit $U = AA^T$ (dim m×m)

und V= A^TA (dim n×n)

S und V bestimmen

Eigenvektoren von A^TA bestimmen:

$(A^TA - \lambda_i E)\vec{x}_i = \vec{0}$ (meist normiert: $\|x_i\| = 1$)

Eigenvektoren ergeben Spalten von **V**.

Eigenwerte von A^TA bestimmen:

$\det(A^TA - \lambda E) = 0$ Lösungsformel: $\lambda_{1,2} = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right)$

Singulärwerte ($\sqrt{\lambda_i}$) auf Hauptdiagonale von **S** $(\dim m \times n)$ eintragen, sonst Nullen.

Faktorisierung: $A^TA = VDV^T$ mit D: Diagonalmatrix der Eigenwerte $\Leftrightarrow \sqrt{D} = S$

U bestimmen

$\vec{u}_i = \frac{1}{\sigma_i}A\vec{v}_i$ mit $\vec{u}_i, \vec{v}_i$= Spaltenvektor von U bzw. V und σ_i = Singulärwert

Fehlenden Vektor (für dim m×m) ergänzen:

im $\mathbb{R}^2$ für normalisierten Vektor $\vec{u}_1 = (a\ b)^T \Rightarrow \vec{u}_2 = (-b\ a)^T$

im $\mathbb{R}^3$: $\vec{u}_3 = \vec{u}_1 \times \vec{u}_2$

Anwendung der SVD

Rang r von A

Anzahl Singulärwerte (Werte auf Hauptdiagonale von S ≠ 0)

Bild von A

Erste r Spaltenvektoren von U

Kern von A

Letzte $n - r$ Spaltenvektoren von V, (leere Menge falls $n - r = 0$)

Konditionszahl bezüglich der Spektralnorm (= euklidischen Norm)

$\kappa_2(\mathrm{A}) = \frac{\sigma_1}{\sigma_n}$, also der größte Singulärwert geteilt durch den kleinsten

Pseudoinverse A~1

$A^{\sim 1} = VS^{\sim 1}U^T$ $S^{\sim 1}$: Kehrwerte auf Hauptdiagonaler von S, Ergebnis transponiert

Lösen von $A\vec{x} = \vec{b}$

$\vec{x} = A^{\sim 1}\vec{b} = (VS^{\sim 1}U^T)\vec{b}\ \epsilon\ O(n^3) = \boldsymbol{V\left(S^{\sim 1}\left(U^T\vec{b}\right)\right)}\ \epsilon\ O(n^2)$

Low Rank Approximation

Letzte $n - k$ Singulärwerte von S durch 0 ersetzen, dann $A_k = US_kV^T$ lösen.

Definition: $A_k = \min_{X:rank(X)=k}\|A - X\|_F$

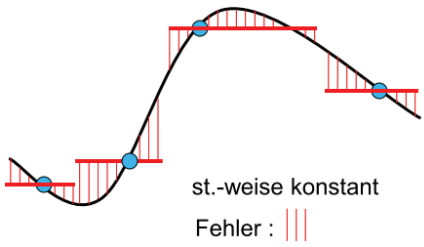
Interpolation

Lokale Verfahren

Nearest Neighbour

Stückweise konstante Funktion:

$$p(x) = \begin{cases} y_1 & x_1 \le x < \frac{x_1+x_2}{2} \\ \vdots & \\ y_i & \frac{x_{i-1}+x_i}{2} \le x < \frac{x_i+x_{i+1}}{2} \\ \vdots & \\ y_n & \frac{x_{n-1}+x_n}{2} \le x < x_n \end{cases}$$

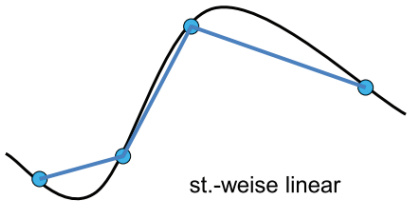


y_i bleibt in der $\frac{\Delta x}{2}$-Umgebung von x_i konstant, überschreitet aber die Randpunkte x_1 bzw. x_n nicht.

Lineare Interpolation

Stückweise lineare Funktion:

$p_i(x) = y_i + \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i)$ für $x_i \leq x < x_{i+1},\ i = 1 \ldots n-1$



$p_i(x)$ ist die Gleichung der Geraden in $[x_i, x_{i+1}[$ und $[x_{n-1}, x_n]$

Catmull-Rom Interpolation

Stückweise kubische Funktion:

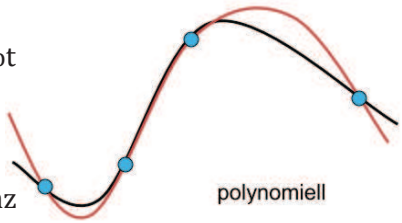
$$y'_i \approx \begin{cases} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \\ \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \\ \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}} \\ \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} y'_{fw} + \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} y'_{bw} \end{cases}$$

Vorwärts-Differenz

Rückwärts-Differenz } not-a-knot

Zentral-symmetrische Differenz

Zentral-unsymmetrische Differenz



Globale Verfahren

Polynom-Interpolation

Lagrange-Polynome

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

$$p(x) = \sum_{i=1}^{n} y_i \cdot L_i(x)$$

Newtonpolynome

$N_0(x) = 1,\ \ N_i(x) = N_{i-1}(x) \cdot (x - x_{i-1})\ \ (i = 1, \ldots n-1)$

Also haben die Polynome die Form:

$N_0(x) = 1$

$N_1(x) = (x - x_0)$

$N_2(x) = (x - x_0)(x - x_1)$

$N_k(x) = (x - x_0)(x - x_1) \cdots (x - x_{k-1})$

$N_{n-1}(x) = (x - x_0)(x - x_1) \cdots (x - x_{n-2})$

Interpolationspolynom (= Newton-Basis):

$$p(x) = \sum_{i=0}^{n-1} c_i \cdot N_i(x)$$

$$= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1) \ldots (x - x_{n-2})$$

Algorithmus von Aitken-Neville

Ziel: Koeffizienten c_i bestimmen

$$
\begin{array}{l|ll}
x_0 & y_0 = \boldsymbol{p_{0,0}} & > \boldsymbol{p_{0,1}} = \dfrac{p_{1,0} - p_{0,0}}{x_1 - x_0} \\
x_1 & y_1 = p_{1,0} & > p_{1,1} = \dfrac{p_{2,0} - p_{1,0}}{x_2 - x_1} \quad \dots \\
x_2 & y_2 = p_{2,0} & > \vdots \quad \dots \\
\vdots & \vdots & > p_{n-2,1} = \dfrac{p_{n-1,0} - p_{i,k-1}}{x_{i+k} - x_i} \quad \dots \\
x_{n-1} & y_{n-1} = p_{n-1,0} &
\end{array}
$$

Für die $p_{i,k}$ und $k > 0$ gilt:

$$p_{i,k} = \frac{p_{i+1,k-1} - p_{i,k-1}}{x_{i+k} - x_i}$$

Koeffizienten c_j in oberer Diagonale ablesen:

$$c_j = \boldsymbol{p_{0,j}}, \quad j = 0 \dots n-1$$

Also:

Im Zähler die vorherigen p, im Nenner die x *schräg rüber* ablesen, jeweils *unten minus oben* rechnen.

Rekonstruktion multivarianter Daten

Baryzentrische Koordinaten (Lokale Interpolation in Dreiecken)

$$\rho = \frac{\det(S - P, T - P)}{\det(S - R, T - R)}$$

$$\sigma = \frac{\det(P - R, T - R)}{\det(S - R, T - R)}$$

$$\tau = \frac{\det(S - R, P - R)}{\det(S - R, T - R)}$$

Im Zähler immer das mit P ausgetauscht, was gesucht ist (Griechisch→Latein).

Alternativ Lösung mit Gleichungssystem:

$$
\begin{aligned}
\rho + \sigma + \tau &= 1 \\
x_R\rho + x_S\sigma + x_T\tau &= x_P \\
y_R\rho + y_S\sigma + y_T\tau &= y_P
\end{aligned}
$$

$\vec{P} = \rho\vec{R} + \sigma\vec{S} + \tau\vec{T}$, Interpolation mit Funktionen analog: $f_p = \rho f_R + \sigma f_S + \tau f_R$

Bilineare Interpolation

$$\alpha = \frac{x_P - x_0}{x_1 - x_0}$$

$$f_0' = (1 - \alpha) f_{00} + \alpha f_{10}$$

$$f_1' = (1 - \alpha) f_{01} + \alpha f_{11}$$

$$\beta = \frac{y_P - y_0}{y_1 - y_0}$$

$$f_P = (1 - \beta) f_0' + \beta f_1'$$

Bézierkurven

Eigenschaften

- Interpolation der Endpunkte (Anfangs- und Endpunkt auf der Kurve)
- In den Endpunkten tangential an das Kontrollpolygon
- Kurve innerhalb der konvexen Hülle
- Affine Invarianz (Verschiebung, Streckung, Drehung wie Kontrollpolygon)
- Variationsreduzierend (eine Gerade schneidet Kurve höchstens so oft wie Kontrollpolygon)

Auswertung

Horner-Bézier-Schema

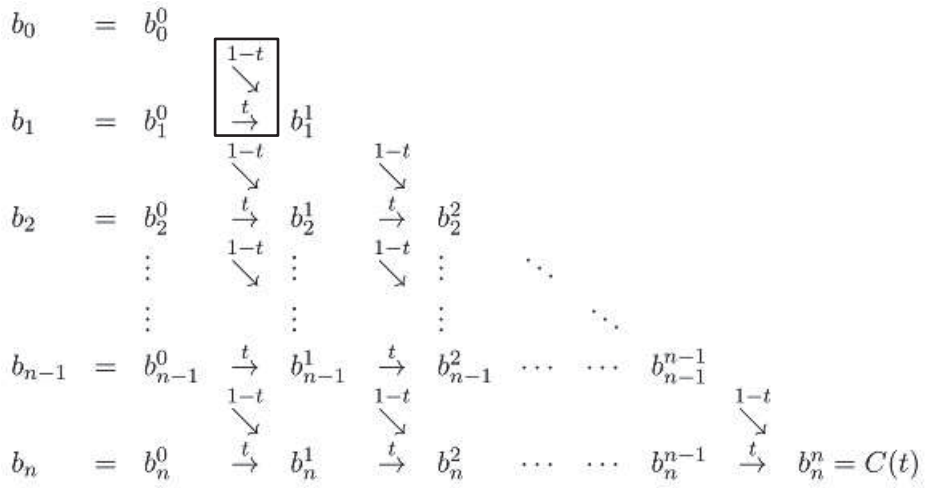
$$\boldsymbol{C(t) = \sum_{i=0}^{n} b_i B_i^n(t) = \sum_{i=0}^{n} b_i \binom{n}{i} (1-t)^{n-i} t^i}$$

$$\boldsymbol{= (1-t)^n \left(b_0 + b_1 \left(\frac{t}{1-t} \right) + b_2 \left(\frac{t}{1-t} \right)^2 + \cdots + b_n \left(\frac{t}{1-t} \right)^n \right)}$$

$$\boldsymbol{= t^n \left(b_0 \left(\frac{1-t}{t} \right)^n + b_1 \left(\frac{1-t}{t} \right)^{n-1} + \cdots + b_{n-1} \left(\frac{1-t}{t} \right) + b_n \right)}$$

Algorithmus von de Casteljau

Gesucht sei ein Punkt auf der Bézier-Kurve an der Stelle t, b_i seien die Vektoren der Kontrollpunkte



Der Vektor $b_i \cdot (1-t)$ *„von oben“* wird zu $b_{i+1} \cdot t$ addiert und rechts neben b_{i+1} geschrieben.

Coons Patch

Auch *transfinite Interpolation* genannt.

$$F(s,t) \coloneqq F_t(s,t) + F_s(s,t) - F_{st}(s,t)$$

$$F_t(s,t) = (1-s)D_0(t) + sD_1(t)$$
$$F_s(s,t) = (1-t)C_0(s) + tC_1(s)$$
$$F_{st}(s,t) = (1-s)\big((1-t)P_{00} + tP_{10}\big) + s((1-t)P_{01} + tP_{11})$$

$$P_{00} = C_0(0) = D_0(0)$$
$$P_{10} = C_0(1) = D_1(0)$$
$$P_{01} = C_1(0) = D_0(1)$$
$$P_{11} = C_1(1) = D_1(1)$$

Lösung linearer Gleichungssysteme

LU-Zerlegung (=LR-Zerlegung)

$$A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ k_2 & 1 & 0 \\ k_3 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & a'_{32} & a'_{33} \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 \\ k_2 & 1 & 0 \\ k_3 & k'_3 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{pmatrix}$$

mit $a'_{ij} = a_{ij} - k_j \cdot a_{1j}$ und $a''_{ij} = a'_{ij} - k'_j \cdot a_{2j}$

In die linke, als Einheitsmatrix vorinitialisierte, Matrix L werden die Koeffizienten k an die Stelle eingetragen, die in U zu 0 werden soll. k ist der Faktor, mit dem die obere Zeile multipliziert und dann von den aktuellen Zeilen subtrahiert wurde.

Lösung über $A\vec{x} = \vec{b} \Leftrightarrow LU\vec{x} = \vec{b} \Rightarrow \boldsymbol{L\vec{y} = \vec{b}} \Rightarrow \boldsymbol{U\vec{x} = \vec{y}}$

Daraus kann abgelesen werden: $\det(A) = \det(U) = U_{11} \cdot U_{22} \cdot \ldots \cdot U_{nn}$

Lineare Ausgleichsprobleme

QR-Zerlegung

Jacobi-Rotationen (= Givens-Rotationen)

In einer Einheitsmatrix der Größe $n \times n$ wird an der Stelle (i, j), die in der Matrix R zu 0 werden soll $-sin(\varphi)$ gesetzt. Auf die Einsen rechts und darüber (auf der Diagonale) wird $\cos(\varphi)$ gesetzt. Mit $\sin(\varphi)$ wird das Quadrat wie folgt zur *Givens-Matrix* vervollständigt:

$$G_{i,j} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos(\varphi) & \cdots & \sin(\varphi) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin(\varphi) & \cdots & \cos(\varphi) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

$$\text{mit } \cos(\varphi) = \frac{a_{lo}}{\sqrt{a_{lo}^2 + a_{lu}^2}} \text{ und } \sin(\varphi) = -\frac{a_{lu}}{\sqrt{a_{lo}^2 + a_{lu}^2}}$$

Dabei ist a_{lo} der Wert von A an der links-oberen Stelle von A und a_{lu} analog an der links-unteren Stelle von A, dem in der Einheitsmatrix die (Ko-)Sinus-Einträge gesetzt wurden.

Wir erhalten:
$G_{nm} \cdot \ldots \cdot G_{21} \cdot A = R$ (obere Dreiecksmatrix)
$Q^T = G_{nm} \cdot \ldots \cdot G_{21}$ (orthogonale, unitäre $n \times n$ –Matrix $\Longrightarrow \det(Q) = \pm 1$)

Householder-Spiegelungen

$\vec{v}_i$: i-te spalte der Matrix mit oben $i - 1$ Nullen.

$\vec{e}_i$: i-ter Einheitsvektor

$\vec{u}_i = \vec{v}_i \pm \|\vec{v}_i\|_2 \cdot \vec{e}_i$

Ob Addition oder Subtraktion: irrelevant, nach „schönerem Ergebnis“ auswählen.

$H_i = E - \frac{2}{\vec{u}_i^T \cdot \vec{u}_i} \cdot \vec{u}_i \cdot \vec{u}_i^T$

Bei einer $n \times m$ –Matrix muss das ganze m mal wiederholt werden:

$H_m \cdot \ldots \cdot H_1 \cdot A = R$ (obere Dreiecksmatrix)

$Q^T = H_2 \cdot H_1$

Vektoren und Matrizen

Normen

Vektornormen

Summennorm:

$$\|x\|_1 = \sum_i |x_i|$$

Euklidische Norm:

$$\|x\|_2 = \sqrt{\left(\sum_{i=1}^{n} x_i^2\right)}$$

Maximumsnorm:

$$\|x\|_\infty = max\{|x_i|\}$$

Allgemein: p-Norm $(1 \leq p < \infty)$:

$$\|x\|_p = \sqrt[p]{\left(\sum_i |x_i|^p\right)}$$

Matrixnormen

Summennorm:

$$\|A\|_1 = \max\left\{\sum_{i=1}^{n} |a_{ij}| : 1 \leq j \leq m\right\} = maximale\ Spaltensumme$$

Maximumsnorm:

$$\|A\|_\infty = max\left\{\sum_{j=1}^{m} |a_{ij}| : 1 \leq i \leq n\right\} = \text{maximale Zeilensumme}$$

Euklidische Norm:

$$\|A\|_2 = \sqrt{\lambda_{max}}\ \ (\text{größter Singulärwert von } A^T A \text{ bzw. } AA^T)$$

Dünnbesetzte Matrizen

CRS-Verfahren (Compressed Row Storage)

val: Nicht-Null-Einträge zeilenweise von links nach rechts (wie beim Lesen)

col_ind[i]: Spaltenindex des Eintrags val[i], *col_ind* genauso groß wie *val* (Anzahl der Nicht-Null-Einträge der Matrix)

row_ptr: Arrayindex des ersten Eintrags einer neuen Zeile. Größe: (Anzahl Zeilen + 1). Letztes Element ist (Anzahl Nicht-Null-Einträge + 1, zeigt also auf Stelle nach den Arrays). Zeile ohne Nicht-Null-Eintrag: Arrayindex des nächsten Nicht-Null-Eintrags wird so oft wiederholt, bis die richtige Zeile erreicht ist.

CCS (Compressed Column Storage)

analog CRS; mit vertauschten Zeilen/Spalten.

PCA

Principal Component Analysis

1. Mittelswertsvektor bestimmen: $\overline{\vec{x}} = \frac{\sum_{i=1}^{n} x_i}{n}$ mit n = Anzahl der Datenpunkte
2. Kovarianzmatrix C bestimmen:
   - Mittelswertsvektor von allen Datenpunkten abziehen
   - Datenpunkte zu Matrix A zusammenfassen
   - Kovarianzmatrix $C \coloneqq \frac{1}{n-1} AA^T$
3. Diagonalisierung (Zerlegung) von C:
   - Eigenwerte und Eigenvektoren von C bestimmen
   - W: Eigenwerte auf Hauptdiagonale (absteigend sortiert)
   - Q: Eigenvektoren als Spalten
   - $C = QWQ^T$
4. Hauptkomponenten
   - 1. Hauptkomponente: Eigenvektor zum größten Eigenwert (in Richtung der größten Varianz)
   - 2. Hauptkomponente: Eigenvektor zum zweigrößten Eigenwert (senkrecht auf 1.HK)
   - etc.
5. Koordinaten der balancierten Daten darstellen: $\overrightarrow{y_i} = Q^T \overline{\overrightarrow{x_i}}$

Bei gegebener SVD:

$$C = U\left(\frac{1}{n-1} S^2\right) U^T$$

Hauptkomponenten sind Spalten von U; Eigenwerte aus Singulärwerten: $\lambda_i = \frac{1}{n-1}\sigma_i^2$

Komplexität von Algorithmen

Gauß'sches Eliminationsverfahren: $O(n^3)$

Mehraufwand für Spaltenpivotisierung: $O(n)$

Vorwärts-/Rückwärtseinsetzen (Dreiecksmatrix): $O(n^2)$

→ $R\vec{x} = \vec{y}$ lösen, nach QR-Zerlegung: $O(n^2)$

→ Lineares Gleichungssystem lösen nach LU-Zerlegung: $O(n^2)$

LU-Zerlegung: $O(n^3)$ (trotzdem sinnvoll für mehrere $\vec{b}_n$)

→ für tridiagonale Matrizen: $O(n)$

Matrix-Matrix-Multiplikation: $O(n^3)$

Matrix-Vektor-Multiplikation: $O(n^2)$

Matrix-Skalar-Multiplikation: $O(n^2)$

Vektor-Vektor-Multiplikation: $O(n)$

Sortier-Algorithmen (best case): $O(n \cdot \log(n))$

Binärsuche: $O(\log(n))$

Lokale Interpolation (best case: äquidistante Intervalle, sortiert): $O(1)$

LGS lösen mit Vandermonde-Matrix: $O(n^3)$

Aitken-Neville-Algorithmus: $O(n^2)$

Polynome in Newton-Basis auswerten (Horner-Schema): $O(n)$

cg-Verfahren: $O(n^3)$

cg-Verfahren (dünnbesetzte Matrizen): $O(n^2)$

Faltung zweier n-Vektoren: $O(n^2)$

DFT mittels FFT: $O(n \cdot \log(n))$

Filter (separiert): $O(n^2)$

→ jede Zeile/Spalte mit 1D-Filter: $O(n)$

Allgemeine Rechenregeln

Binomialkoeffizient

$$\binom{n}{0} = \binom{n}{n} = 1 \qquad \binom{n}{k} = 0, \text{ wenn } k > n$$

$$\binom{n}{1} = \binom{n}{n-1} = n \qquad \binom{n}{2} = \binom{n}{n-2} = \sum_{J=1}^{n-1} j$$

$\binom{n}{k} = \binom{n}{n-k}$ (Symmetrie)

Nicht in der Formelsammlung enthalten:

Vandermonde-Matrix

Least Square Methode

B-Spline-Interpolation

erweitertes Horner-Bézier-Schema (Seite 5, Skript-Seite 77)

Numerische Integration

- Trapezregel
- Simpson-Regel

Nichtlineare Optimierung

- cg-Verfahren

Poisson-Gleichung

Iterative Verfahren für lineare Gleichungssysteme

- Jacobi-Verfahren
- Gauß-Seidel-Verfahren

Filter

Fourier-Transformation

Permutation in Blockmatrizen (Adjazenzmatrizen)

CRS/CCS: Multiplikation CRS*Vektor (an sich trivial)

Komplexe Einheitsmatrizen (FS. S.80) – Allg. Rechenregeln

…und andere