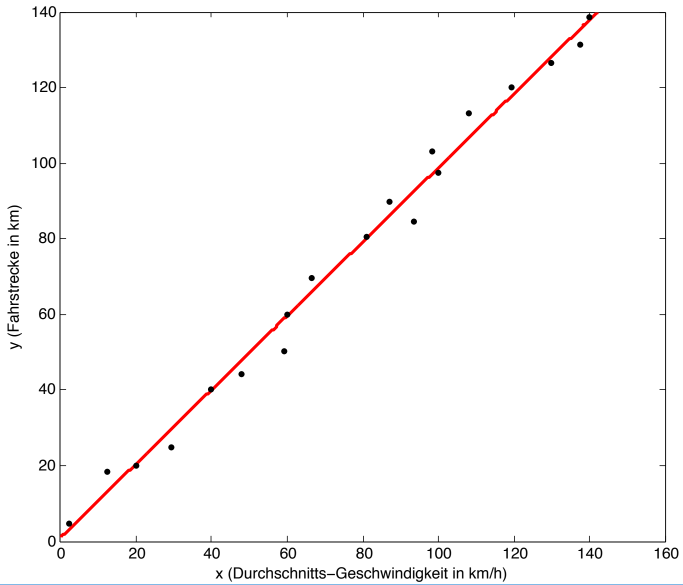
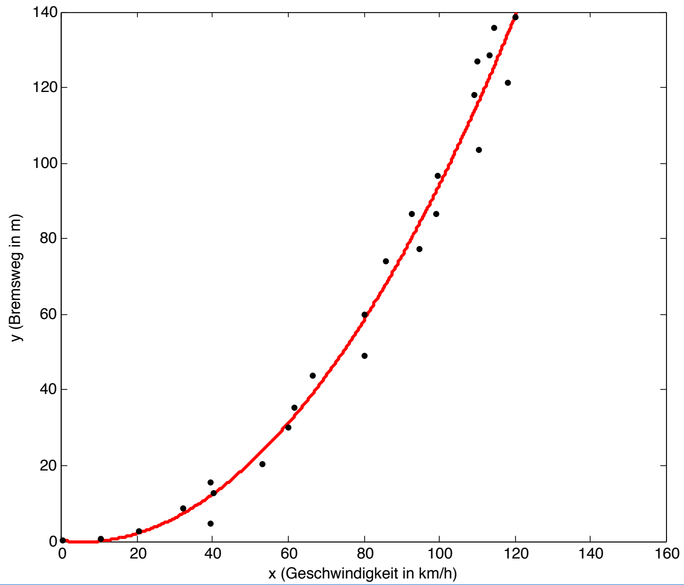
Beispiel-Problem: Geschwindigkeit vs. Fahrstrecke



Beispiel-Problem: Geschwindigkeit vs. Bremsweg



Problemstellung Least Squares

- **Gegeben:** Datenreihe mit m Datenpunkten

$$(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$$

  mit $x_j, y_j \in \mathbb{R}$ für $j = 1, \dots, m$
- **Erwartung:** y_j enthalten Messfehler ($j = 1, \dots, m$)
- **Gesucht:** Funktion $F : \mathbb{R} \to \mathbb{R}$ so dass Approximationsfehler

$$\eta_j = F(x_j) - y_j$$

  für alle $j = 1, \dots, m$ möglichst gering
- **Annahme:** F lässt sich darstellen als Summe von Basisfunktionen f_i,

$$F(x) = \sum_{i=1}^{n} c_i f_i(x)$$

Wahl der Basisfunktionen

Wahl der Basisfunktionen f_i für F:

- typische Wahl: $f_i(x) = x^{i-1}$, dann

$$F(x) = c_1 + c_2 x + c_3 x^2 + \dots + c_n x^{n-1}$$

  (Polynom in x vom Grad $n - 1$)
- auch oft mit $n = 2$, dann

$$F(x) = c_1 + c_2 x$$

  (Gerade) auch genannt lineare Regression
- im Fall von Tomographie: Pixel- oder Voxel-Basisfunktionen

Matrix-Notation

- für die verschiedenen Datenpunkte x_j, $j = 1, \dots, m$, kann F geschrieben werden als:

$$\begin{pmatrix} F(x_1) \\ \vdots \\ F(x_m) \end{pmatrix} = \underbrace{\begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & & \vdots \\ f_1(x_m) & \cdots & f_n(x_m) \end{pmatrix}}_{=:A} \underbrace{\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}}_{=:c}$$

  mit $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$.
- untersucht wird dann der Approximationsfehler η

$$\eta = Ac - y$$

  mit $\eta = (\eta_1, \dots, \eta_m) \in \mathbb{R}^m$, $y = (y_1, \dots, y_m) \in \mathbb{R}^m$.

Daten mit Meßfehler

ist $m = n$ und A invertierbar, so kann direkt

$$Ac - y = 0$$

gelöst werden.

Problem:

- selbst wenn A invertierbar ist, y enthält Meßfehler
- Lösung F ist dann meist nicht die **gewünschte** Lösung
- passt sich zu sehr an "Ausreißer" an

besser: mehr Datenpunkte, $m >> n$

Minimierung des Approximationsfehlers

- zur Minimierung des Approximationsfehlers kann z.B. die Norm $\|\eta\|$ betrachtet werden

$$\|\eta\| = \left(\sum_{j=1}^{m} \eta_j^2 \right)^{\frac{1}{2}} = \sqrt{\sum_{j=1}^{m} \eta_j^2}$$

- zur Vereinfachung betrachtet man üblicherweise

$$\|\eta\|^2 = \|Ac - y\|^2 = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} a_{ji} c_i - y_j \right)^2$$

- daher der Name: Methode der kleinsten Quadrate oder Least squares

Least-squares Lösung

Least-squares Lösung
Sei $A \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$ mit $m \geq n$. Eine Lösung $c \in \mathbb{R}^n$ des Minimierungs-Problems

$$\min_c \|Ac - y\| \quad \text{or} \quad \min_c \|Ac - y\|^2$$

heißt Least-squares Lösung.

Anwendungs-Beispiele:

- Tracking von Objekten mit Kameras
- Kalibrierung von Kameras, Robotern, etc.
- Iterative Rekonstruktion für Tomographie

Normalengleichung

- Berechnung der Least-squares Lösung mit Standard-Technik "Ableitung gleich null setzen"
- hier: partielle Ableitungen für $k = 1, \dots, n$

$$\frac{\partial \|\eta\|^2}{\partial c_k} = \sum_{j=1}^{m} 2\left(\sum_{i=1}^{n} a_{ji}c_i - y_j\right) a_{jk} = 0$$

- daraus folgt

$$\nabla \|\eta\|^2 = A^T(Ac - y) = 0$$

- umgeformt ergibt sich

$$A^TAc = A^Ty$$

auch genannt die Normalengleichung.

Normalengleichung

Normalengleichung
Sei $A \in \mathbb{R}^{m\times n}$, $y \in \mathbb{R}^m$ mit $m \geq n$. $c \in \mathbb{R}^n$ ist eine Least-squares Lösung von $\min_c \|Ac - y\|$ genau dann, wenn die Normalengleichung gilt:

$$A^TAc = A^Ty$$

Insbesondere ist die Normalengleichung lösbar, falls $\text{rank}(A) = n$.

- Die Matrix A^TA ist immer symmetrisch.
- Falls $\text{rank}(A) = n$ ist A^TA auch positiv definit und damit invertierbar
- Die Lösung der Normalengleichung ist dann

$$c = ((A^TA)^{-1}A^T)y.$$

Pseudoinverse

Pseudoinverse
Sei $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ und $\text{rank}(A) = n$. Die Matrix

$$A^+ := (A^TA)^{-1}A^T$$

heißt Pseudoinverse von A (auch Moore-Penrose Inverse genannt).

- Die Pseudoinverse verallgemeinert das Konzept der Inversen für nicht-quadratische Matrizen.
- Ist A invertierbar, dann gilt $A^+ = A^{-1}$.

Pseudoinverse

Für $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ löse

$$A^TAc = A^Ty$$

Umbennung der Variablen:

$$Mx = b$$

- $x = c$
- $M = A^TA$; Berechnung in Zeit $\mathcal{O}(n^2m)$; (trivialer Algorithmus)
- $b = A^Ty$; Berechnung in Zeit $\mathcal{O}(mn)$; (trivialer Algorithmus)

Lösen von linearen Systemen

Inverse Matrix
Sei $M \in \mathbb{R}^{n\times n}$ quadratische Matrix. Falls ein $M' \in \mathbb{R}^{n\times n}$ existiert mit

$$MM' = M'M = I_n$$

(wobei I_n die $n \times n$ Einheitsmatrix ist), dann heißt M invertierbar und wir nennen $M' =: M^{-1}$ das Inverse von M.

- Ist $Mx = b$ lineares System mit $M \in \mathbb{R}^{n\times n}$ invertierbar, dann ist

  $$x = M^{-1}b$$

  die Lösung des linearen Systems. Diese Lösung ist eindeutig.

Invertierbarkeit von Matrizen

Problem 1: wann ist eine Matrix $M \in \mathbb{R}^{n\times n}$ invertierbar?

- $Mx = 0$ hat nur die triviale Lösung $x = 0$

Problem 2: falls M invertierbar, wie findet man das Inverse M^{-1}?

- Berechnung mit Gauss-Jordan Elimination
- Umweg über Zerlegungen von M (z.B. LU-Zerlegung)
- **generell:** Berechnung der Inversen meist numerisch nicht stabil

Günstige Matrix-Formen

- M diagonal: Lösung kann abgelesen werden

$$\begin{pmatrix} M_{11} & & 0 \\ & \ddots & \\ 0 & & M_{nn} \end{pmatrix} x = b$$

- M obere Dreiecksmatrix: Rückwärtssubstitution

$$\begin{pmatrix} M_{11} & \cdots & M_{1n} \\ & \ddots & \vdots \\ 0 & & M_{nn} \end{pmatrix} x = b$$

- M untere Dreiecksmatrix: Vorwärtssubstitution

$$\begin{pmatrix} M_{11} & & 0 \\ \vdots & \ddots & \\ M_{n1} & \cdots & M_{nn} \end{pmatrix} x = b$$

Rückwärtssubstitution

Lineares System in oberer Dreiecksform:

$$\begin{array}{rcrcrcrcl} M_{11}x_1 & + & M_{21}x_2 & + & \ldots + M_{1,n-1}x_{n-1} & + & M_{1n}x_n & = & b_1 \\ & & M_{22}x_2 & + & \ldots + M_{2,n-1}x_{n-1} & + & M_{2n}x_n & = & b_2 \\ & & & & & & & \vdots & \\ & & & & M_{n-1,n-1}x_{n-1} & + & M_{n-1,n}x_n & = & b_{n-1} \\ & & & & & & M_{n,n}x_n & = & b_n \end{array}$$

Rückwärts nacheinander nach $x_n, x_{n-1}, \ldots, x_1$ auflösen:

$$x_i = \left(b_i - \sum_{j=i+1}^{n} M_{ij}x_j \right) / M_{ii}$$

Aufwand: $\mathcal{O}(n^2)$ arithmetische Operationen (FLOPs)

Vorwärtssubstitution

Lineares System in unterer Dreiecksform:

$$\begin{array}{rcl}
M_{11}x_1 & & = b_1 \\
M_{21}x_1 + M_{22}x_2 & & = b_2 \\
\vdots & & \vdots \\
M_{n-1,1}x_1 + M_{n-1,2}x_2 + \ldots + M_{n-1,n-1}x_{n-1} & & = b_{n-1} \\
M_{n1}x_1 + M_{n2}x_2 + \ldots + M_{n,n-1}x_{n-1} + M_{nn}x_n & & = b_n
\end{array}$$

Vorwärts nacheinander nach $x_1, x_2, \ldots, x_n$ auflösen:

$$x_i = \left(b_i - \sum_{j=1}^{i-1} M_{ij}x_j\right) / M_{ii}$$

Aufwand: $\mathcal{O}(n^2)$ FLOPs

Gauss Elimination

Gauss-Elimination: Überführung der Matrix A in Dreiecksform durch elementare Zeilenumformungen

- Komplexität: $\Theta(n^3)$

Gauss-Jordan-Elimination: Überführung der Matrix A in Diagonalform durch elementare Zeilenumformungen

- Komplexität: $\Theta(n^3)$

Elementare Zeilenumformungen:

- Vielfaches einer Zeile zu einer anderen Zeile addieren
- zwei Zeilen vertauschen
- Vielfaches einer Zeile berechnen

für lineare Systeme: Gauss-Elimination auf erweiterter Matrix

$$(A|b) = \left(\begin{array}{ccc|c} M_{11} & \cdots & M_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ M_{n1} & \cdots & M_{nn} & b_n \end{array}\right)$$

Gauss Elimination: Beispiel

$$\begin{array}{rcrcrcrl}
2x & + & y & - & z & = & 8 & (G_1) \\
-3x & - & y & + & 2z & = & -11 & (G_2) \\
-2x & + & y & + & 2z & = & -3 & (G_3)
\end{array}$$

$$\left(\begin{array}{rrr|r} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{array}\right) \xrightarrow{G_2 += \frac{3}{2}G_1} \left(\begin{array}{rrr|r} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ -2 & 1 & 2 & -3 \end{array}\right)$$

$$\xrightarrow{G_3 += G_1} \left(\begin{array}{rrr|r} 2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 2 & 1 & 5 \end{array}\right) \xrightarrow{G_3 += -4G_2} \left(\begin{array}{rrr|r} -2 & 1 & -1 & 8 \\ 0 & \frac{1}{2} & \frac{1}{2} & 1 \\ 0 & 0 & -1 & 1 \end{array}\right)$$

$$\xrightarrow{G_1/2,\ G_2\cdot 2,\ G_3\cdot(-1)} \left(\begin{array}{rrr|r} 1 & \frac{1}{2} & -\frac{1}{2} & 4 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

Lösung nun durch Rücksubstitution: $z = -1$, $y = 3$ und $x = 2$

Gauss Elimination: Algorithmus

```
1  Gauss(M)
2      for (j=0; j<n-1; j++)
3          foundNonZero = false;
4          for (i=j; i<n-1; i++)
5              if (M_ij != 0)
6                  swap(i,j);
7                  foundNonZero = true;
8                  break;
9          if (!foundNonZero)
10             printf("Matrix not invertible");
11             return;
12         for (i=j+1; i<n-1; i++)
13             M_i += -(M_ij/M_jj)*M_j;
```

- Laufzeit: Die äußere Schleife wird n mal durchlaufen; die inneren Schleifen $\leq n$ mal. D.h. jede Operation wird höchstens n^2 mal ausgeführt.
- Die swap-Operation in Zeile 5 und die Operation in Zeile 11 sind aber nicht elementar und benötigen Laufzeit $\mathcal{O}(n)$.
- $\mathcal{O}(n^3)$

Erklärung

- j geht über die Spalten; und i über die Zeilen;
- Zu Beginn der von Iteration j, haben alle Zeilen $i \geq j$ Nullen in Spalten ℓ mit $\ell < j$.
- In Iteration j bekommen die Zeilen $i > j$ Nullen in Spalte ℓ;
- Die Zeile j soll aber in Spalte j einen Wert $M_{jj} \neq 0$ haben; dazu wird erst nach einer geeigneten Zeile gesucht (in den Zeilen mit $i \geq j$) und ggfs. eine Vertauschung vorgenommen. Falls dies fehlschlägt haben alle Zeilen $i \geq j$ Nullen in Spalte ℓ mit $1 \leq \ell \leq j$. D.h., diese insgesamt $n - j$ Zeilen haben höchstens $n - j - 1$ Spalten, die nicht Null sind. Damit sind diese Zeilen linear abhängig und die Matrix ist nicht invertierbar.
- In einer Iteration der for-Schleife in Zeile 11 wird die j-te Zeile mit einem geeigneten Vielfachen multipliziert und zu der i-ten Zeile addiert, so daß letztere einen Nulleintrag in Spalte j bekommt.

Inverse via Gauss-Jordan Elimination

Sei $A \in \mathbb{R}^{n \times n}$.

- Bilde erweiterte Matrix

$$(A|I_n) = \left(\begin{array}{ccc|ccc} a_{11} & \cdots & a_{n1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} & 0 & \cdots & 1 \end{array}\right)$$

  wobei I_n die $n \times n$ Identitätsmatrix ist.
- Führe Gauss-Jordan Elimination für A aus (linker Teil) bis auf I_n reduziert, repliziere elementare Zeilenumformungen für I_n (rechter Teil).
- Am Ende entspricht rechter Teil A^{-1}.

Pseudoinverse

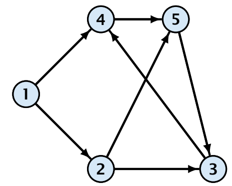
Falls A^TA nicht invertierbar ist die Pseudoinverse gegeben durch die Matrix A^+, die folgende Bedinungen erfüllt:

- $AA^+A = A$
- $A^+AA^+ = A+$
- $(AA^+)^T = AA^+$
- $(A^+A)^T = A^+A$

Theorem
Diese Matrix existiert immer.

Anwendung

gegeben: ungerichteter Graph $G = (V, E)$

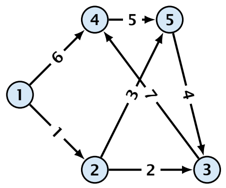


weise Kanten beliebige Richtungen zu (Zählpfeile)

Definition: Die Knoten-Kanten Inzidenzmatrix B ist wie folgt definiert:

- eine Zeile für jede Kante; eine Spalte für jeden Knoten
- die Zeile für Kante $e = (u, v)$ enthält ein 1 in Spalte u und eine -1 in Spalte v

Anwendung



$$\underbrace{\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \end{pmatrix}}_{B}$$

Angenommen Vektor x ordnet jedem Knoten ein Potential zu und die Kanten des Graphen haben Widerstand 1.

Bx = Vektor von Strömen entlang der Kanten

Anwendung

Bx = Vektor von Strömen entlang der Kanten

$\underbrace{B^T B}_{L} x$ = Vektor von Strömen aus Knoten

Gegeben: Vektor $b \in \mathbb{R}^n$ der Ein- und Ausgangsströme für alle Knoten angibt (z.B. einfach +1 an Knoten v und -1 an Knoten u bei einer Stromquelle zwischen diesen Knoten).

Löse

$$Lx = b$$

$L^+ b$ ist die Lösung der Gleichung die $\|x\|_2$ minimiert.