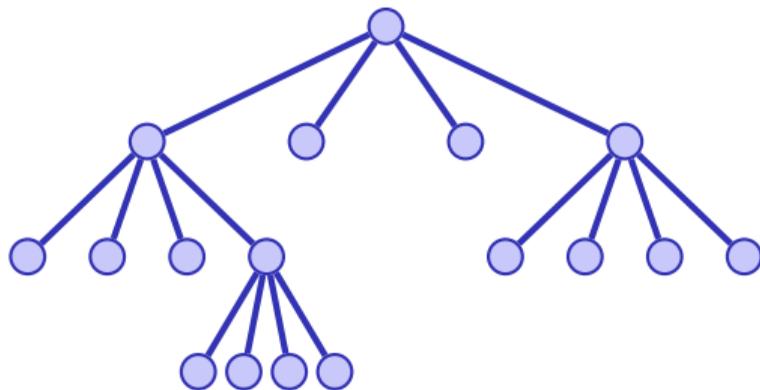


19. Quadrees

Quadrees, Kollisionsdetektion, Bildsegmentierung

Quadtree

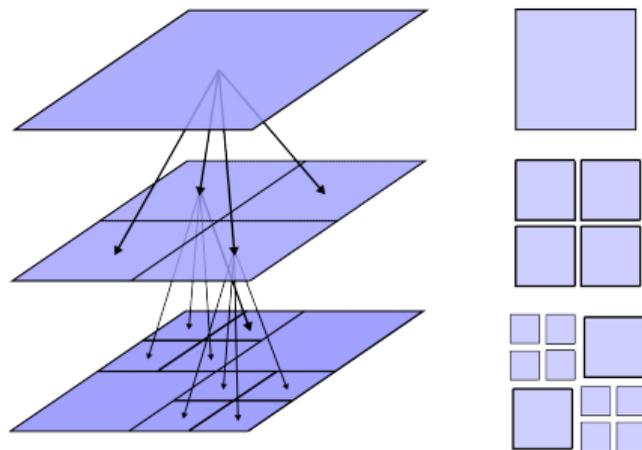
Ein Quadtree ist ein Baum der Ordnung 4.



... und ist als solcher nicht besonders interessant, ausser man verwendet ihn zur...

Quadtree - Interpretation und Nutzen

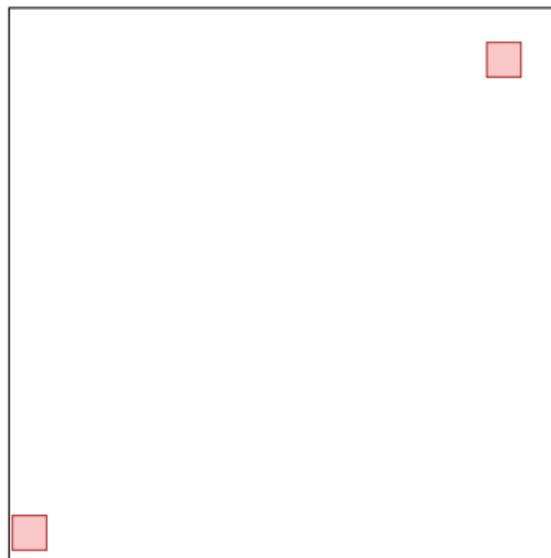
Partitionierung eines zweidimensionalen Bereiches in 4 gleich grosse Teile.



[Analog für drei Dimensionen mit einem *Octtree* (Baum der Ordnung 8)]

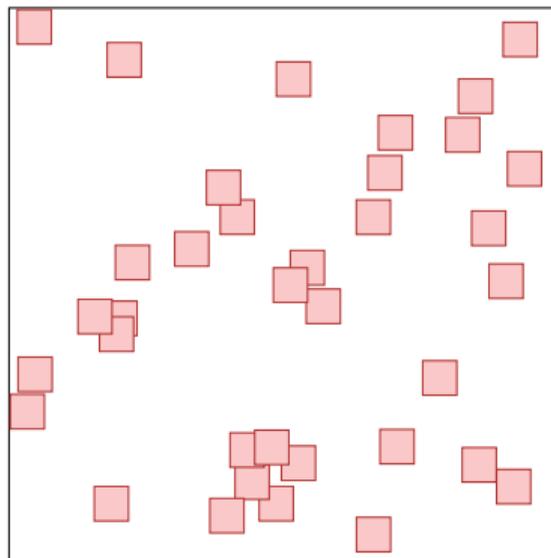
Beispiel 1: Erkennung von Kollisionen

- Objekte in der 2D-Ebene, z.B. Teilchensimulation auf dem Bildschirm.
- Ziel: Erkennen von Kollisionen



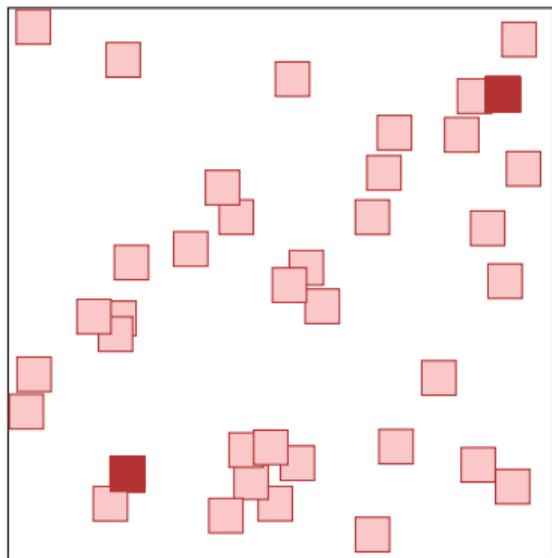
Idee

- Viele Objekte: n^2 Vergleiche (naiv)
- Verbesserung?



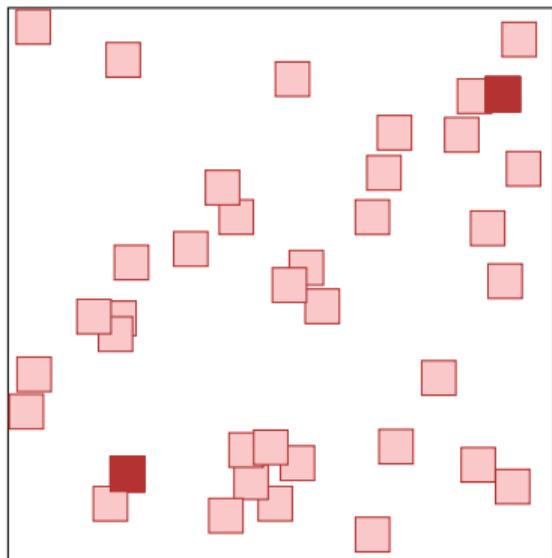
Idee

- Viele Objekte: n^2 Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.



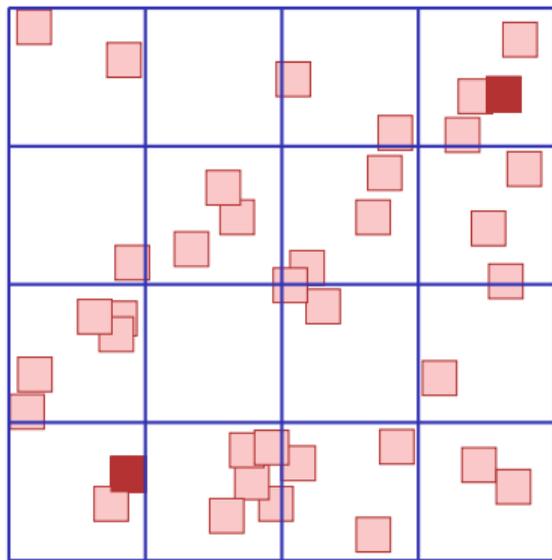
Idee

- Viele Objekte: n^2 Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?



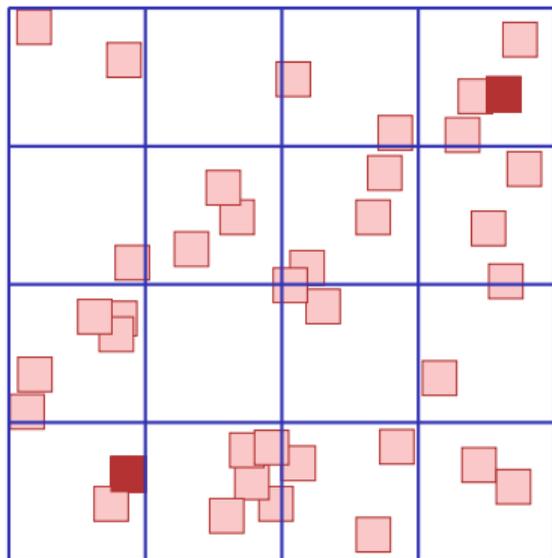
Idee

- Viele Objekte: n^2 Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?
- Gitter ($m \times m$)

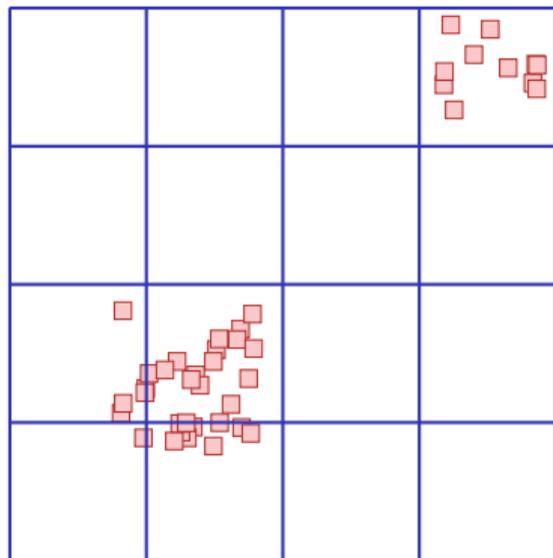


Idee

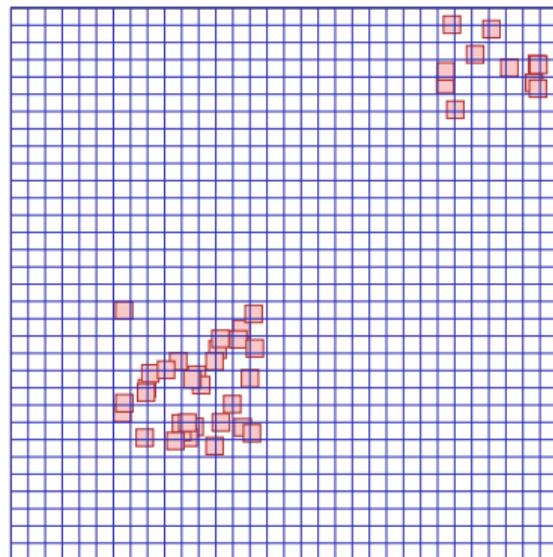
- Viele Objekte: n^2 Vergleiche (naiv)
- Verbesserung?
- Offensichtlich: keine Kollisionsdetektion für weit entfernte Objekte nötig.
- Was ist „weit entfernt“?
- Gitter ($m \times m$)
- Kollisionsdetektion pro Gitterzelle



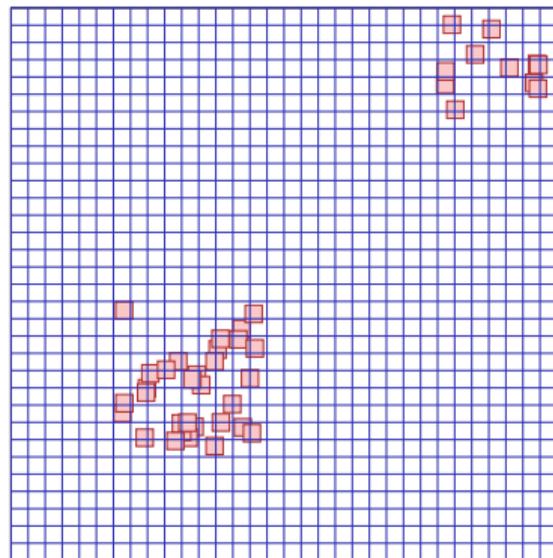
- Gitter hilft oft, aber nicht immer
- Verbesserung?



- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter verfeinern?

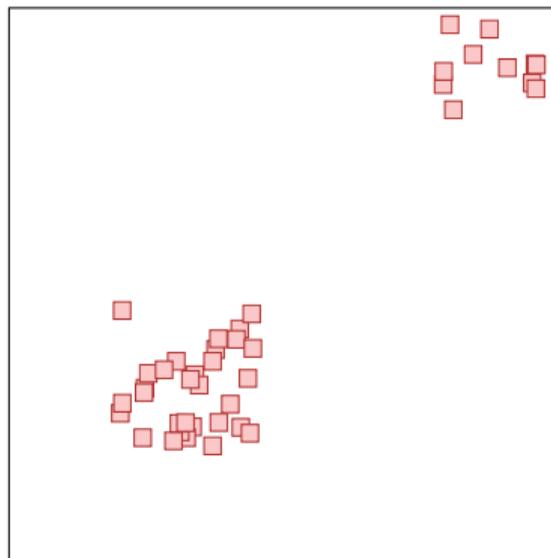


- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter verfeinern?
- Zu viele Gitterzellen!



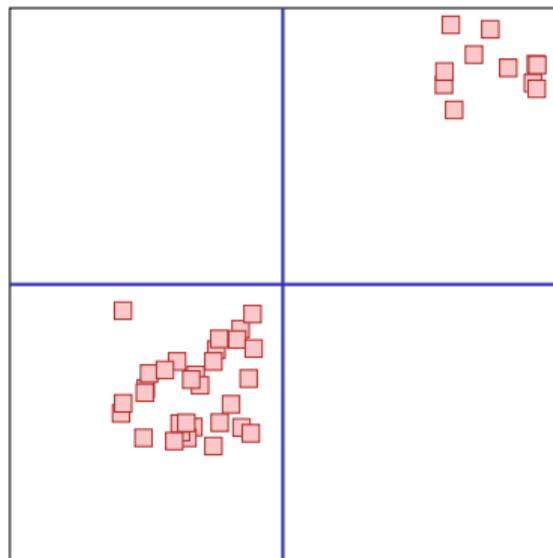
Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?



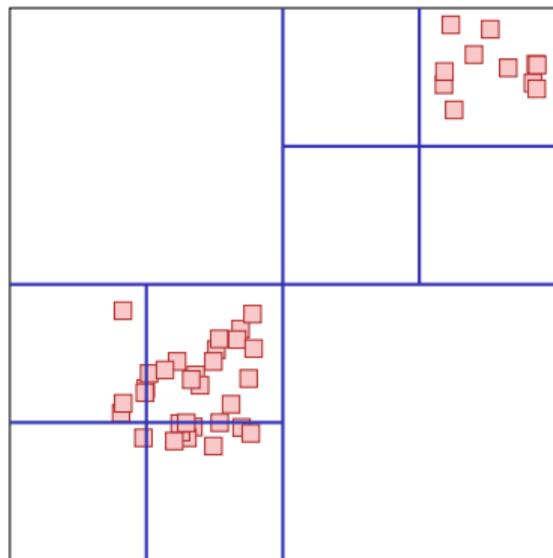
Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!



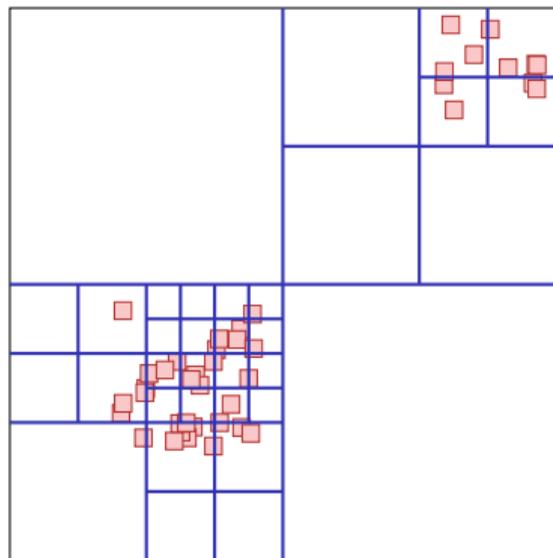
Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!



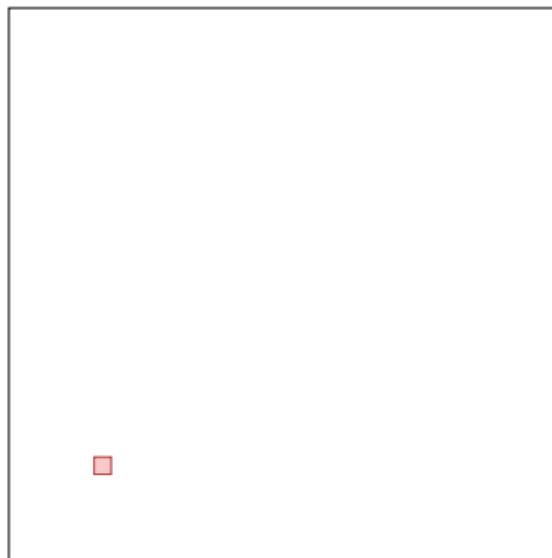
Adaptive Gitter

- Gitter hilft oft, aber nicht immer
- Verbesserung?
- Gitter *adaptiv* verfeinern!
- Quadtree!



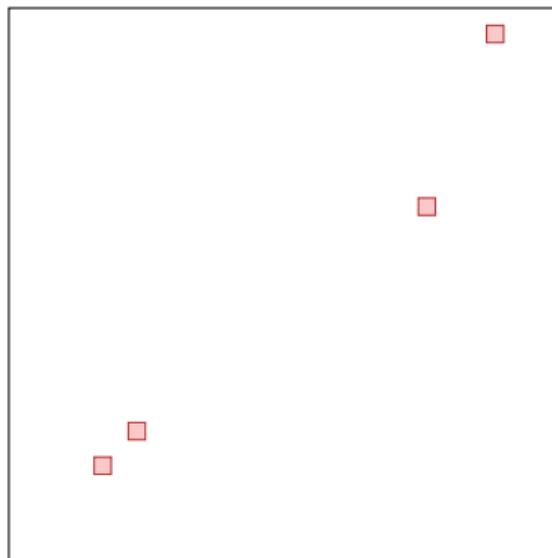
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten



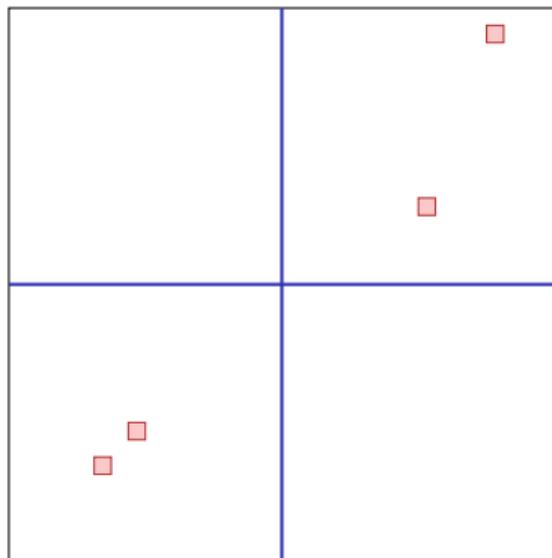
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



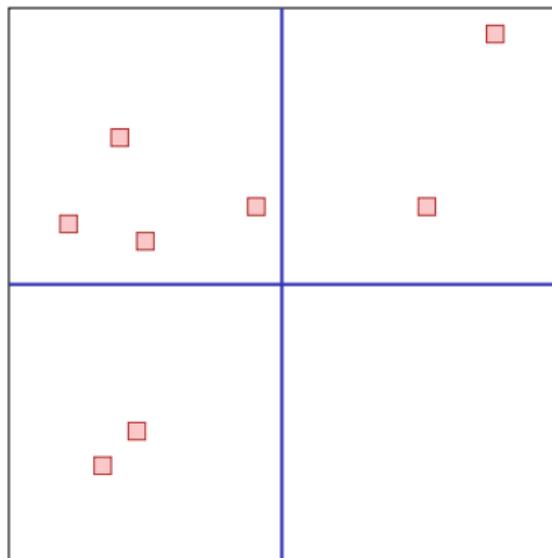
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



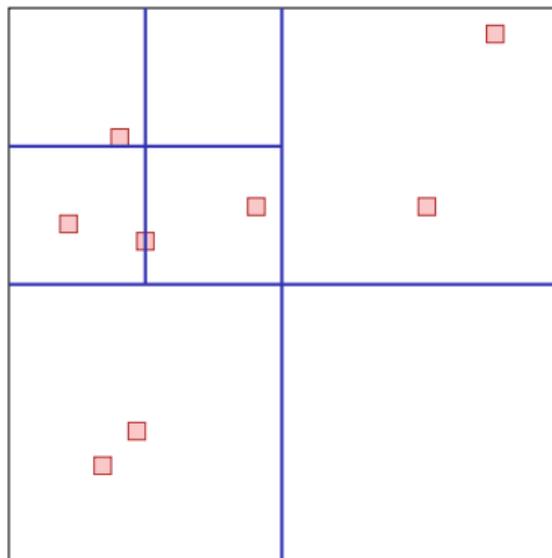
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



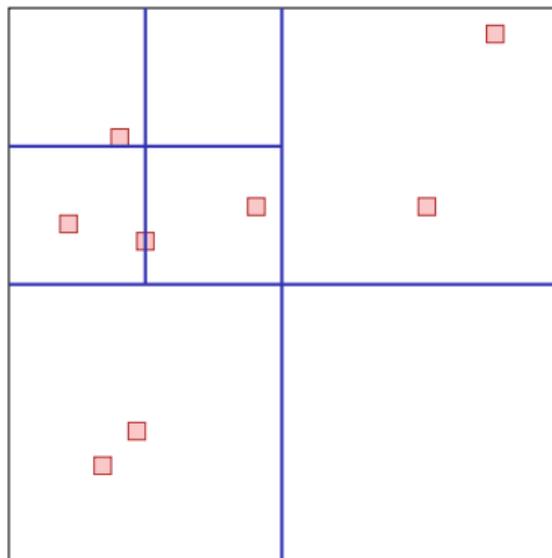
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.



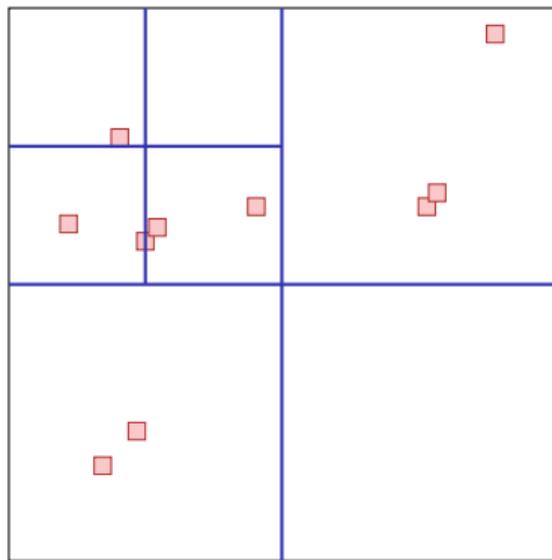
Algorithmus: Einfügen

- Quadtree startet mit einem einzigen Knoten
- Objekte werden zu dem Knoten hinzugefügt. Wenn in einem Knoten zu viele Objekte sind, wird der Knoten geteilt.
- Objekte, die beim Split auf dem Rand zu liegen kommen, werden im höher gelegenen Knoten belassen.

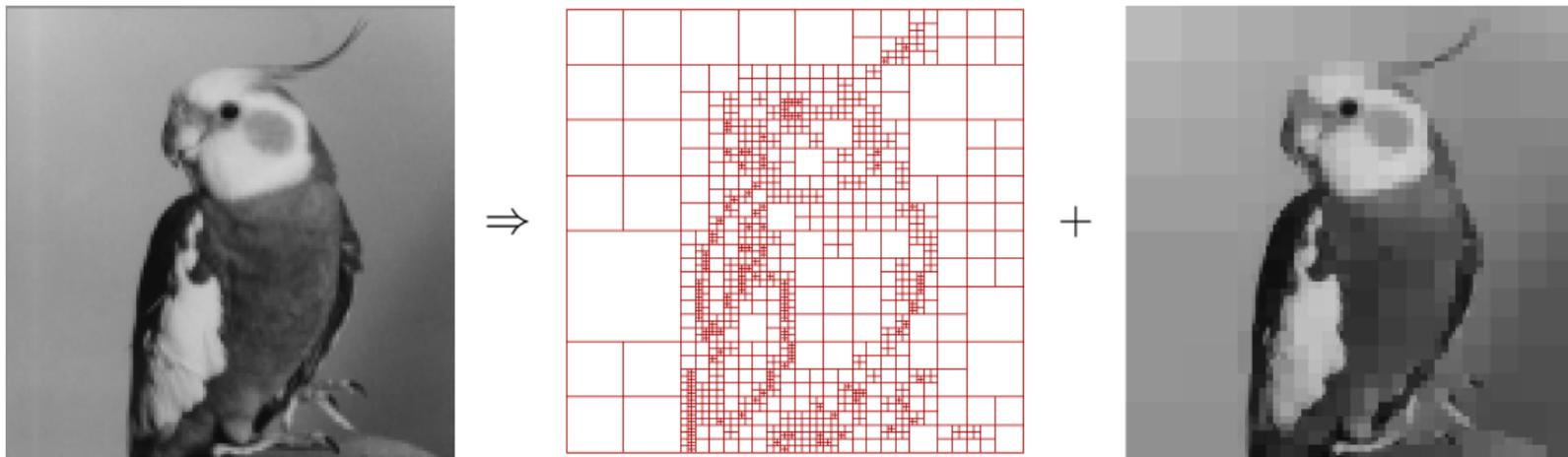


Algorithmus: Kollisionsdetektion

- Durchlaufe den Quadtree rekursiv. Für jeden Knoten teste die Kollision der enthaltenen Objekte mit Objekten im selben Knoten oder (rekursiv) enthaltenen Knoten.

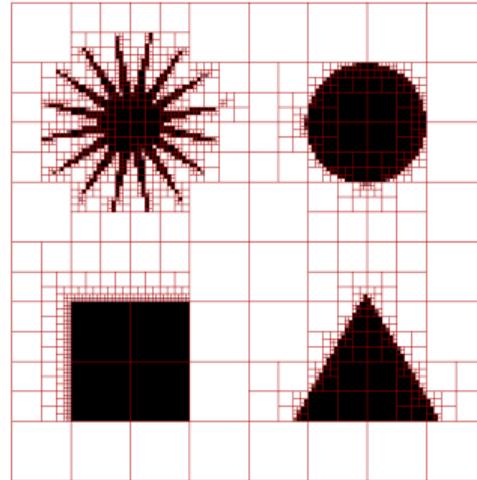
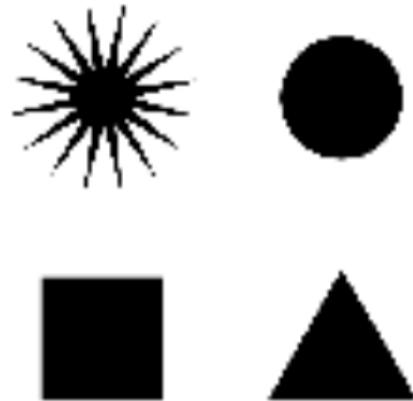


Beispiel 2: Bildsegmentierung



(Mögliche Anwendungen: Kompression, Entrauschen, Kantendetektion)

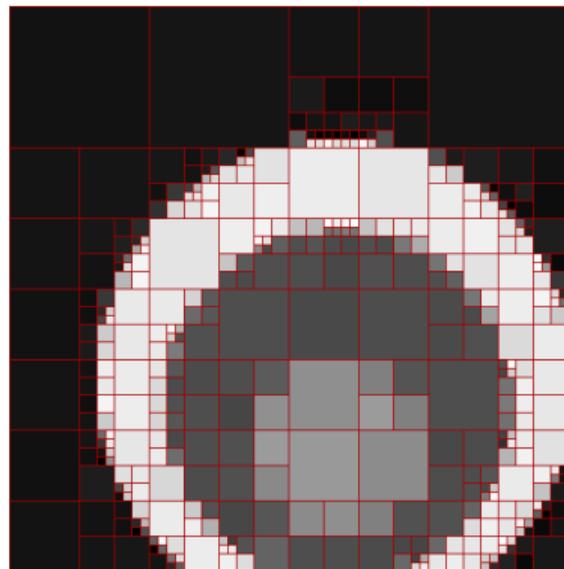
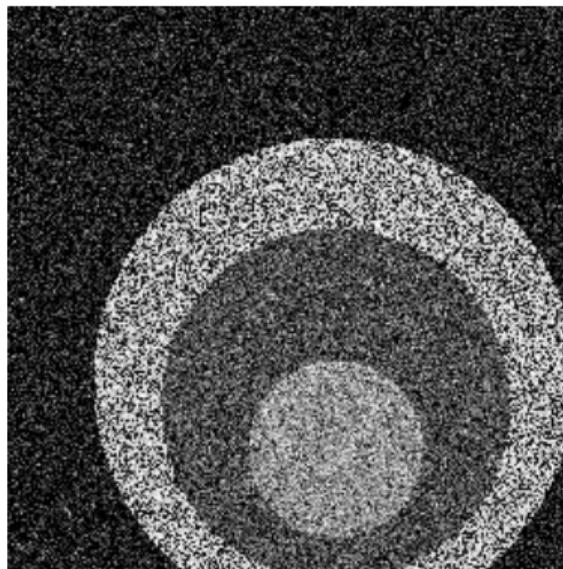
Quadtree auf Einfarbenbild



Erzeugung des Quadtree ähnlich wie oben: unterteile Knoten rekursiv bis jeder Knoten nur Pixel einer Farbe enthält.

Quadtree mit Approximation

Wenn mehr als zwei Farbewerte vorhanden sind, wird der Quadtree oft sehr gross. \Rightarrow Komprimierte Darstellung: *approximiere* das Bild stückweise konstant auf Rechtecken eines Quadtrees.



Stückweise konstante Approximation

(Graustufen-)Bild $\mathbf{y} \in \mathbb{R}^S$ auf den Pixelindizes S .³⁰

Rechteck $r \subset S$.

Ziel: bestimme

$$\arg \min_{v \in \mathbb{R}} \sum_{s \in r} (y_s - v)^2$$

³⁰Wir nehmen an, dass S ein Quadrat ist mit Seitenlänge 2^k für ein $k \geq 0$

Stückweise konstante Approximation

(Graustufen-)Bild $\mathbf{y} \in \mathbb{R}^S$ auf den Pixelindizes S .³⁰

Rechteck $r \subset S$.

Ziel: bestimme

$$\arg \min_{v \in \mathbb{R}} \sum_{s \in r} (y_s - v)^2$$

Lösung: das arithmetische Mittel $\mu_r = \frac{1}{|r|} \sum_{s \in r} y_s$

³⁰Wir nehmen an, dass S ein Quadrat ist mit Seitenlänge 2^k für ein $k \geq 0$

Zwischenergebnis

Die im Sinne des mittleren quadratischen Fehlers beste Approximation

$$\mu_r = \frac{1}{|r|} \sum_{s \in r} y_s$$

und der dazugehörige Fehler

$$\sum_{s \in r} (y_s - \mu_r)^2 =: \|\mathbf{y}_r - \boldsymbol{\mu}_r\|_2^2$$

können nach einer $\mathcal{O}(|S|)$ Tabellierung schnell berechnet werden:
Präfixsummen!

Welcher Quadtree?

Konflikt

- Möglichst nahe an den Daten \Rightarrow kleine Rechtecke, grosser Quadtree.
Extremer Fall: ein Knoten pro Pixel. Approximation = Original
- Möglichst wenige Knoten \Rightarrow Grosse Rechtecke, kleiner Quadtree
Extremfall: ein einziges Rechteck. Approximation = ein Grauwert

Welcher Quadtree?

Idee: wähle zwischen Datentreue und Komplexität durch Einführung eines Regularisierungsparameters $\gamma \geq 0$

Wähle Quadtree T mit Blättern³¹ $L(T)$ so, dass T folgenden Funktion minimiert

$$H_\gamma(T, \mathbf{y}) := \gamma \cdot \underbrace{|L(T)|}_{\text{Anzahl Blätter}} + \underbrace{\sum_{r \in L(T)} \|y_r - \mu_r\|_2^2}_{\text{Summierter Approximationsfehler aller Blätter}} .$$

³¹hier: Blatt = Knoten mit Nullkindern,

Sei T ein Quadtree über einem Rechteck S_T und seien $T_{ll}, T_{lr}, T_{ul}, T_{ur}$ vier mögliche Unterbäume und

$$\widehat{H}_\gamma(T, y) := \min_T \gamma \cdot |L(T)| + \sum_{r \in L(T)} \|y_r - \mu_r\|_2^2$$

Extremfälle:

$\gamma = 0 \Rightarrow$ Originaldaten;

$\gamma \rightarrow \infty \Rightarrow$ ein Rechteck

Beobachtung: Rekursion

- Wenn der (Sub-)Quadtree T nur ein Pixel hat, so kann nicht aufgeteilt werden und es gilt

$$\widehat{H}_\gamma(T, \mathbf{y}) = \gamma$$

- Andernfalls seien

$$M_1 := \gamma + \|\mathbf{y}_{S_T} - \boldsymbol{\mu}_{S_T}\|_2^2$$

$$M_2 := \widehat{H}_\gamma(T_{ll}, \mathbf{y}) + \widehat{H}_\gamma(T_{lr}, \mathbf{y}) + \widehat{H}_\gamma(T_{ul}, \mathbf{y}) + \widehat{H}_\gamma(T_{ur}, \mathbf{y})$$

Dann

$$\widehat{H}_\gamma(T, \mathbf{y}) = \min\left\{\underbrace{M_1(T, \gamma, \mathbf{y})}_{\text{kein Split}}, \underbrace{M_2(T, \gamma, \mathbf{y})}_{\text{Split}}\right\}$$

Algorithmus: Minimize(\mathbf{y}, r, γ)

Input: Bilddaten $\mathbf{y} \in \mathbb{R}^S$, Rechteck $r \subset S$, Regularisierung $\gamma > 0$

Output: $\min_T \gamma |L(T)| + \|\mathbf{y} - \boldsymbol{\mu}_{L(T)}\|_2^2$

if $|r| = 0$ **then return** 0

$m \leftarrow \gamma + \sum_{s \in r} (y_s - \mu_r)^2$

if $|r| > 1$ **then**

 Split r into $r_{ll}, r_{lr}, r_{ul}, r_{ur}$

$m_1 \leftarrow \text{Minimize}(\mathbf{y}, r_{ll}, \gamma)$; $m_2 \leftarrow \text{Minimize}(\mathbf{y}, r_{lr}, \gamma)$

$m_3 \leftarrow \text{Minimize}(\mathbf{y}, r_{ul}, \gamma)$; $m_4 \leftarrow \text{Minimize}(\mathbf{y}, r_{ur}, \gamma)$

$m' \leftarrow m_1 + m_2 + m_3 + m_4$

else

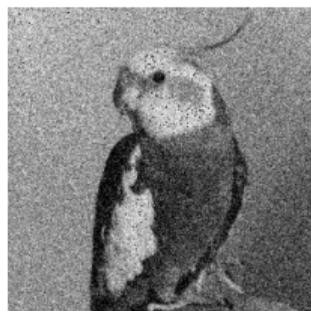
$m' \leftarrow \infty$

if $m' < m$ **then** $m \leftarrow m'$

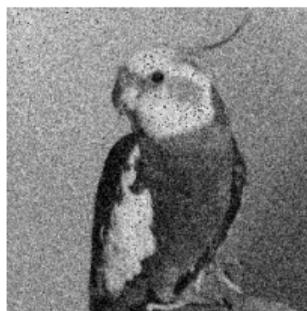
return m

Der Minimierungsalgorithmus über dyadische Partitionen (Quadtree) benötigt $\mathcal{O}(|S| \log |S|)$ Schritte.

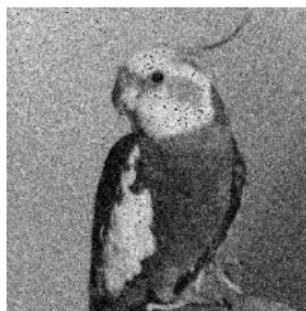
Anwendung: Entrauschen (zusätzlich mit Wedgelets)



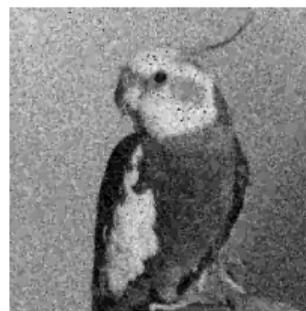
noised



$\gamma = 0.003$



$\gamma = 0.01$



$\gamma = 0.03$



$\gamma = 0.1$



$\gamma = 0.3$



$\gamma = 1$

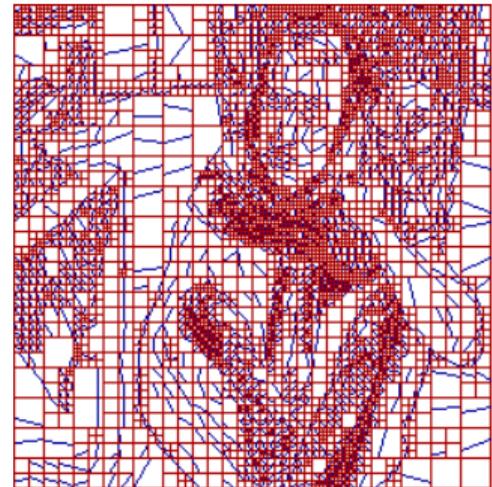


$\gamma = 3$



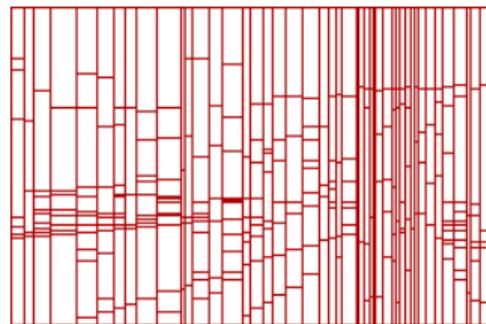
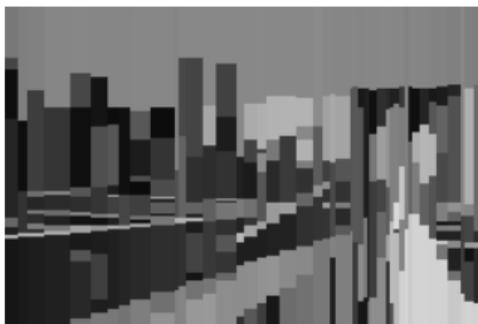
$\gamma = 10$

Erweiterungen: Affine Regression + Wedgelets



Andere Ideen

kein Quadtree: hierarchisch-eindimensionales Modell (benötigt Dynamic Programming)



19.1 Anhang

Lineare Regression

Das Lernproblem

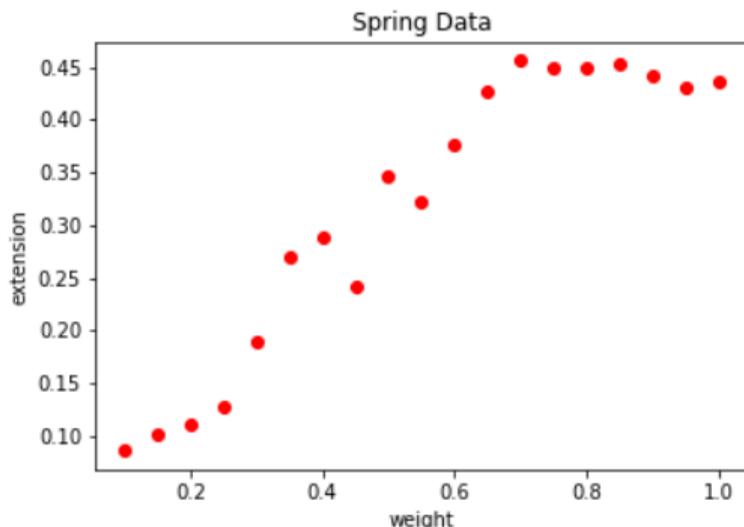
Ausgangslage

- Wir beobachten N Datenpunkte
- Eingaben: $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_N)^\top$
- Beobachtete Ausgaben:
 $\mathbf{y} = (y_1, \dots, y_N)^\top$
- Annahme: es gibt eine zugrundeliegende Wahrheit

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Ziel: Finden einer approximativen Wahrheit $h \approx f$, um Prädiktionen $h(x)$ für neue Datenpunkte x zu machen oder um die Daten zu erklären und beispielsweise komprimiert darzustellen.

Hier $\mathcal{X} = \mathbb{R}^d$. $\mathcal{Y} = \mathbb{R}$ (Regression).



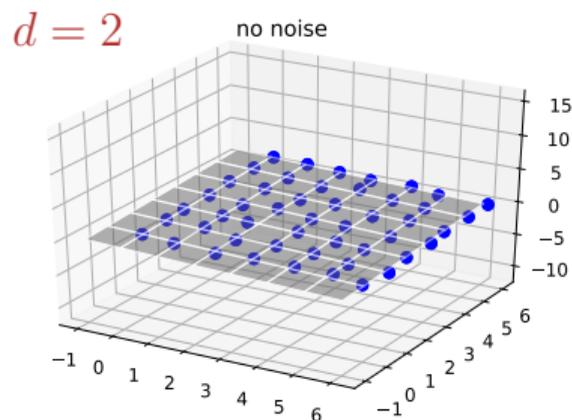
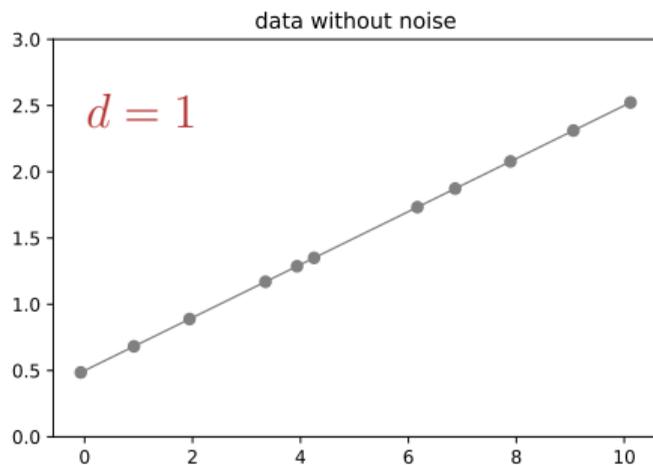
Modell: Lineare Regression

Annahme: Die zugrunde liegende Wahrheit lässt sich darstellen als

$$h_w(\mathbf{x}) = w_0 + w_1x_1 + \dots + w_dx_d = w_0 + \sum_{i=1}^d w_ix_i.$$

⇒ Wir suchen w (manchmal auch d).

linear in w !



Trick für vereinfachte Notation

$$\mathbf{x} = (x_1, \dots, x_d) \rightarrow (\underbrace{x_0}_{\equiv 1}, x_1, \dots, x_d)$$

$$\begin{aligned} h_{\mathbf{w}}(\mathbf{x}) &= w_0 x_0 + w_1 x_1 + \dots + w_d x_d \\ &= \sum_{i=0}^d w_i x_i \\ &= \mathbf{w}^\top \mathbf{x} \end{aligned}$$

Datenmatrix

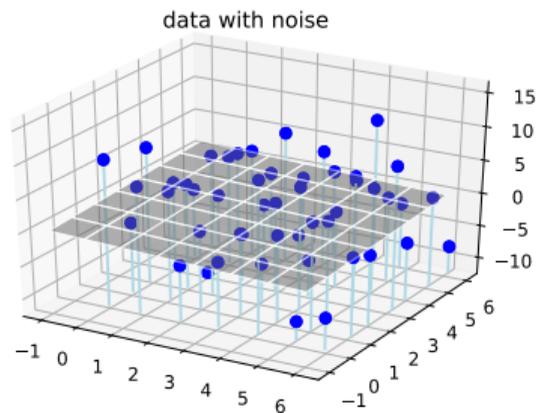
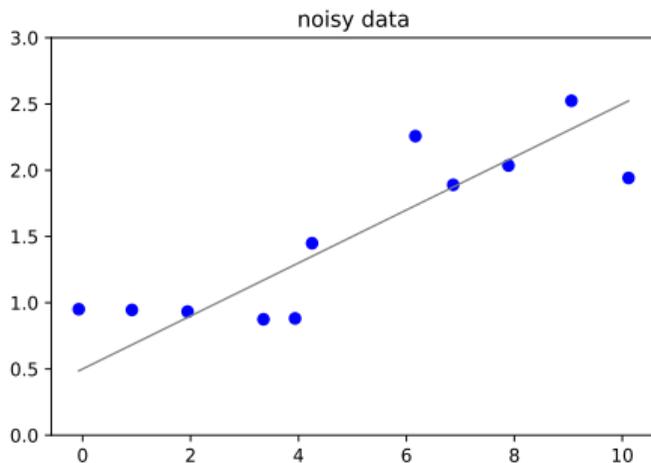
$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_n \end{bmatrix} = \begin{bmatrix} X_{1,0} & X_{1,1} & X_{1,2} & \dots & X_{1,d} \\ X_{2,0} & X_{2,1} & X_{2,2} & \dots & X_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{n,0} & X_{n,1} & X_{n,2} & \dots & X_{n,d} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$$

$\equiv 1$
↓

$$\mathbf{X}\mathbf{w} \approx \mathbf{y}?$$

Ungenau Beobachtungen

Realität: die Daten sind ungenau bzw. das Modell ist nur ein Modell.



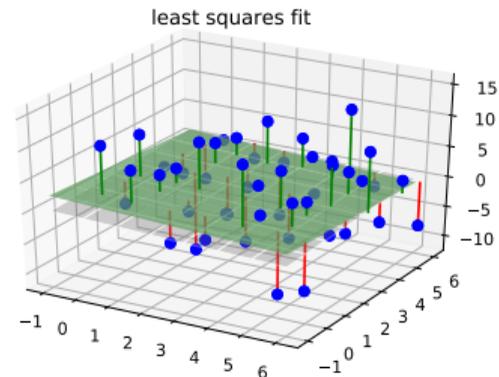
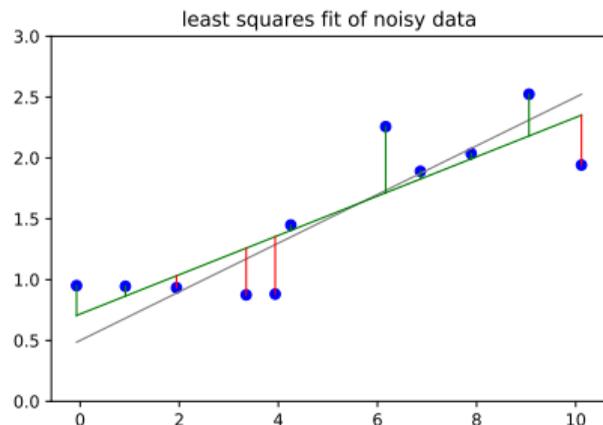
Was tun?

Fehlerfunktion

$$E(\mathbf{w}) = \sum_{i=1}^N (h_{\mathbf{w}}(\mathbf{X}_i) - y_i)^2$$

Suchen ein $\hat{\mathbf{w}}$ das $E(\mathbf{w})$ minimiert.

Linearität von $h_{\mathbf{w}}$ in $\mathbf{w} \Rightarrow$ Lösung mit linearer Algebra.



$$\widehat{\boldsymbol{w}} = \underbrace{\left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top}_{=:\boldsymbol{X}^\dagger} \boldsymbol{y}.$$

\boldsymbol{X}^\dagger : Moore-Penroe Pseudo-Inverse

Polynome fitten

Geht genauso mit linearer Regression.

$$h_{\mathbf{w}}(x) = w_0 + w_1x^1 + w_2x^2 + \cdots + w_dx^d = w_0 + \sum_{i=1}^d w_ix^i.$$

Denn $h_{\mathbf{w}}(x)$ ist immer noch linear in \mathbf{w} !

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & (x_1)^2 & \cdots & (x_1)^d \\ 1 & x_2 & (x_2)^2 & \cdots & (x_2)^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & (x_n)^2 & \cdots & (x_n)^d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix}$$

Beispiel: Konstante Approximation

$$\mathbf{X} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{w} = [w_0]$$

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \left[\frac{1}{n} \sum y_i \right].$$

Beispiel: Lineare Approximation

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} \\ 1 & x_2^{(1)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{w} = [w_0]$$

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \begin{bmatrix} N & \sum x_i^{(1)} & \sum x_i^{(2)} \\ \sum x_i^{(1)} & \sum (x_i^{(1)})^2 & \sum x_i^{(1)} \cdot x_i^{(2)} \\ \sum x_i^{(2)} & \sum x_i^{(1)} \cdot x_i^{(2)} & \sum (x_i^{(2)})^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum y_i \\ \sum y_i \cdot x_i^{(1)} \\ \sum y_i \cdot x_i^{(2)} \end{bmatrix}$$