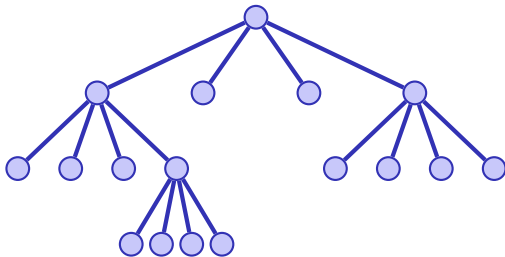


19. Quadtrees

Quadtrees, Collision Detection, Image Segmentation

Quadtree

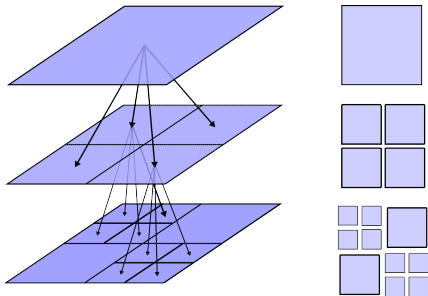
A quad tree is a tree of order 4.



... and as such it is not particularly interesting except when it is used for ...

Quadtree - Interpretation und Nutzen

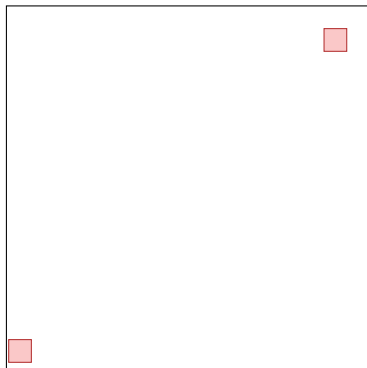
Separation of a two-dimensional range into 4 equally sized parts.



[analogously in three dimensions with an *octtree* (tree of order 8)]

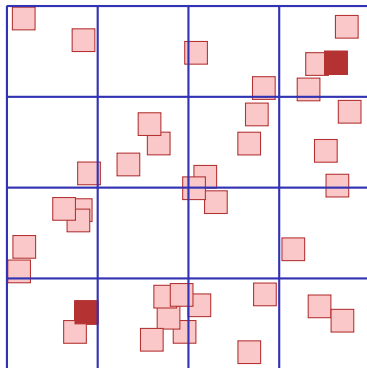
Example 1: Collision Detection

- Objects in the 2D-plane, e.g. particle simulation on the screen.
- Goal: collision detection



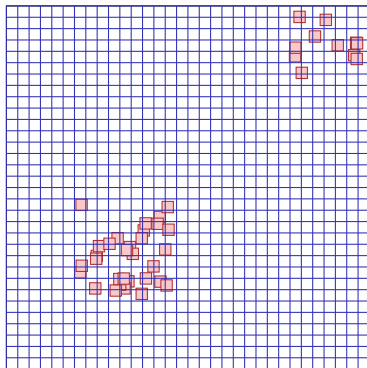
Idea

- Many objects: n^2 detections (naively)
- Improvement?
- Obviously: collision detection not required for objects far away from each other
- What is „far away“?
- Grid ($m \times m$)
- Collision detection per grid cell



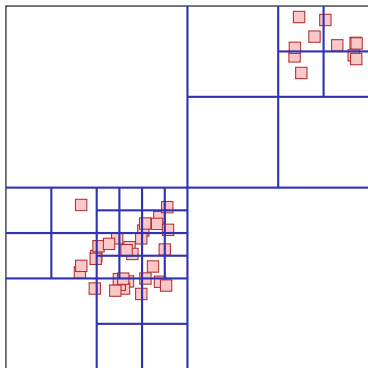
Grids

- A grid often helps, but not always
- Improvement?
- More finegrained grid?
- Too many grid cells!



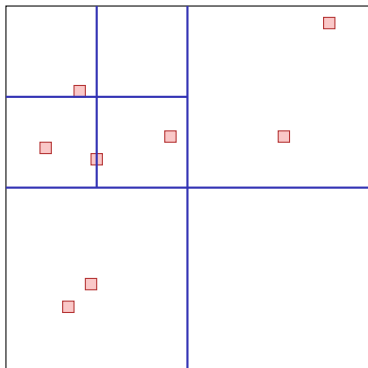
Adaptive Grids

- A grid often helps, but not always
- Improvement?
- *Adaptively* refine grid
- Quadtree!



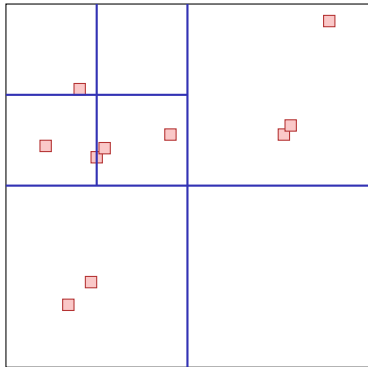
Algorithm: Insertion

- Quadtree starts with a single node
- Objects are added to the node. When a node contains too many objects, the node is split.
- Objects that are on the boundary of the quadtree remain in the higher level node.

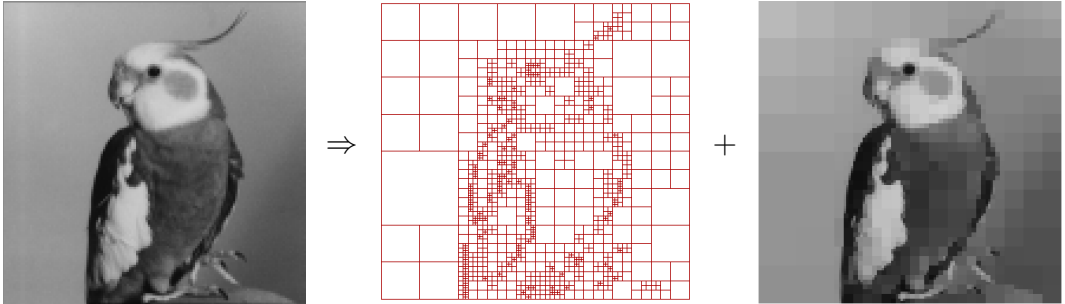


Algorithm: Collision Detection

- Run through the quadtree in a recursive way. For each node test collision with all objects contained in the same or (recursively) contained nodes.

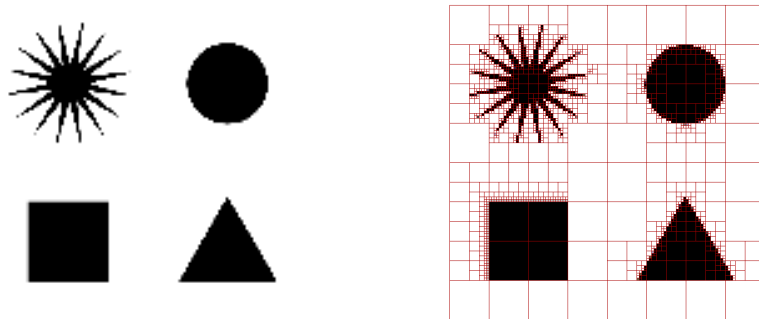


Example 2: Image Segmentation



(Possible applications: compression, denoising, edge detection)

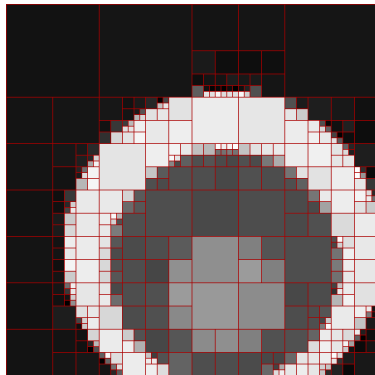
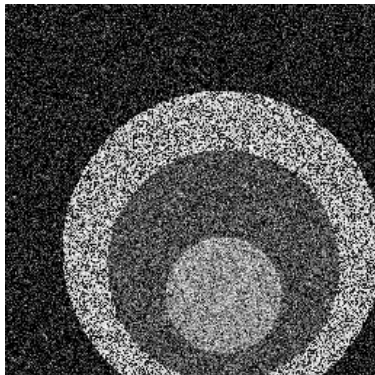
Quadtree on Monochrome Bitmap



Similar procedure to generate the quadtree: split nodes recursively until each node only contains pixels of the same color.

Quadtree with Approximation

When there are more than two color values, the quadtree can get very large. \Rightarrow Compressed representation: *approximate* the image piecewise constant on the rectangles of a quadtree.



Piecewise Constant Approximation

(Grey-value) Image $y \in \mathbb{R}^S$ on pixel indices S .²⁸

Rectangle $r \subset S$.

Goal: determine

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

Solution: the arithmetic mean $\mu_r = \frac{1}{|r|} \sum_{s \in r} y_s$

²⁸we assume that S is a square with side length 2^k for some $k \geq 0$

Intermediate Result

The (w.r.t. mean squared error) best approximation

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

and the corresponding error

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

can be computed quickly after a $\mathcal{O}(|S|)$ tabulation: prefix sums!

Which Quadtree?

Conflict

- As close as possible to the data \Rightarrow small rectangles, large quadtree .
Extreme case: one node per pixel. Approximation = original
- Small amount of nodes \Rightarrow large rectangles, small quadtree Extreme case: a single rectangle. Approximation = a single grey value.

Which Quadtree?

Idea: choose between data fidelity and complexity with a regularisation parameter $\gamma \geq 0$

Choose quadtree T with leaves²⁹ $L(T)$ such that it minimizes the following function

$$H_\gamma(T, \mathbf{y}) := \gamma \cdot \underbrace{|L(T)|}_{\text{Number of Leaves}} + \underbrace{\sum_{r \in L(T)} \|y_r - \mu_r\|_2^2}_{\text{Cumulative approximation error of all leaves}}.$$

²⁹here: leaf: node with null-children

Regularisation

Let T be a quadtree over a rectangle S_T and let $T_{ll}, T_{lr}, T_{ul}, T_{ur}$ be the four possible sub-trees and

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

Extreme cases:

$\gamma = 0 \Rightarrow$ original data;

$\gamma \rightarrow \infty \Rightarrow$ a single rectangle

Observation: Recursion

- If the (sub-)quadtree T represents only one pixel, then it cannot be split and it holds that

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

- Let, otherwise,

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

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

then

$$\widehat{H}_\gamma(T, y) = \min\{\underbrace{M_1(T, \gamma, \mathbf{y})}_{\text{no split}}, \underbrace{M_2(T, \gamma, \mathbf{y})}_{\text{split}}\}$$

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

Input: Image data $\mathbf{y} \in \mathbb{R}^S$, rectangle $r \subset S$, regularization $\gamma > 0$

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

$m' \leftarrow \infty$

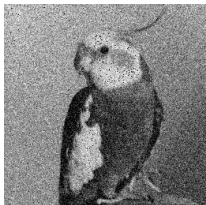
if $m' < m$ **then** $m \leftarrow m'$

return m

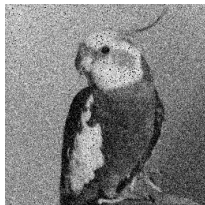
Analysis

The minimization algorithm over dyadic partitions (quadtrees) takes $\mathcal{O}(|S| \log |S|)$ steps.

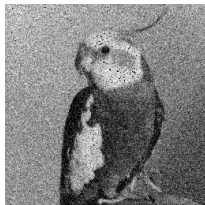
Application: Denoising (with additional Wedgelets)



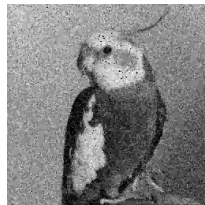
noised



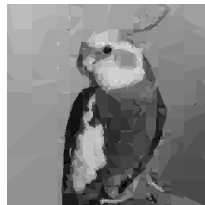
$\gamma = 0.003$



$\gamma = 0.01$



$\gamma = 0.03$



$\gamma = 0.1$



$\gamma = 0.3$



$\gamma = 1$

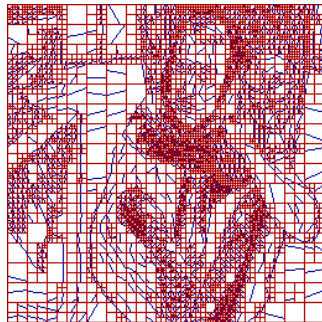


$\gamma = 3$



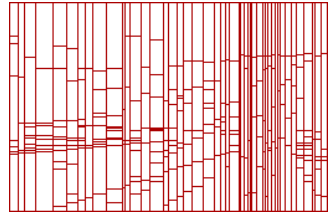
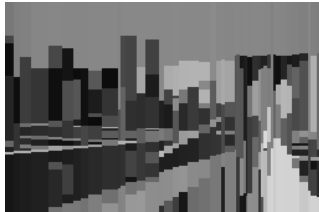
$\gamma = 10$

Extensions: Affine Regression + Wedgelets



Other ideas

no quadtree: hierarchical one-dimensional model (requires dynamic programming)



19.1 Appendix

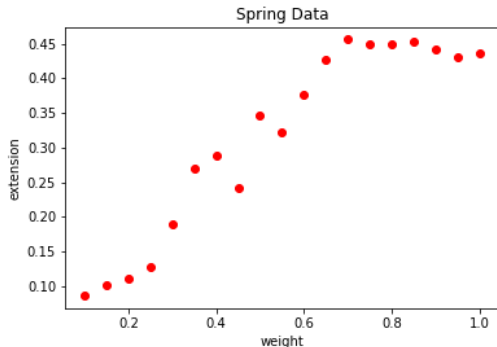
Linear Regression

The Learning Problem

Setup

- We observe N data points
- Input examples: $\mathbf{X} = (\mathbf{X}_1, \dots, \mathbf{X}_N)^\top$
- Output examples: $\mathbf{y} = (y_1, \dots, y_N)^\top$
- Assupmtion: there is an underlying truth

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



Goal: find a good approximation $h \approx g$ to make predictions $h(x)$ for new data points or to explain the data in order to find a compressed representation, for instance.

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

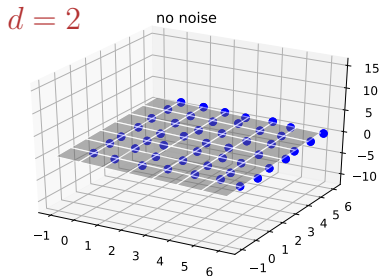
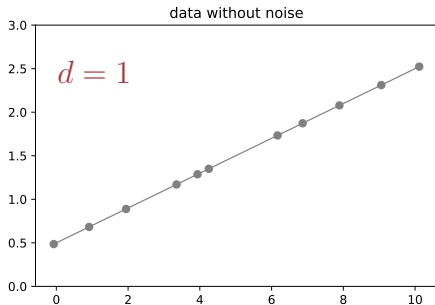
Model: Linear Regression

Assumption: The underlying truth can be represented as

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

⇒ We search for \mathbf{w} (sometimes also d).

linear in \mathbf{w} !



Trick for simplified notation

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

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

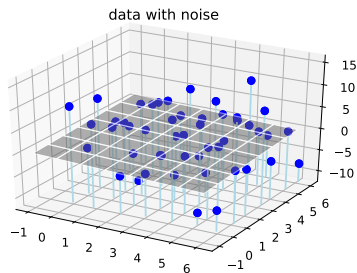
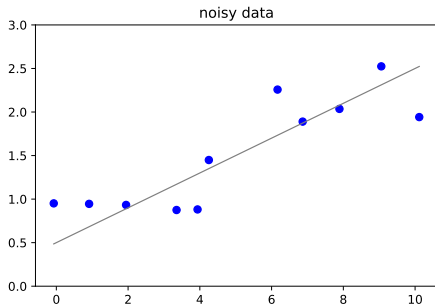
Data matrix

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

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

Imprecise observations

Reality: the data are imprecise or the model is only a model.



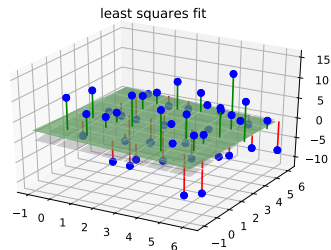
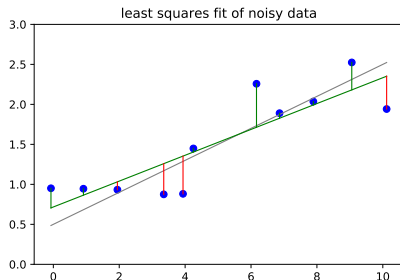
What to do?

Error function

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

Want a $\hat{\mathbf{w}}$ that minimizes E

Linearity of $h_{\mathbf{w}}$ in $\mathbf{w} \Rightarrow$ solution with linear algebra.



Solution from Linear 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

Fitting Polynomials

Also works with linear regression.

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

because $h_{\mathbf{w}}(x)$ remains being linear in \mathbf{w} !

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

Example: Constant Approximation

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

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

Example: Linear Approximation

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

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