Linear regression with gradient descent

Ingmar Schuster

Patrick Jähnichen

using slides by Andrew Ng

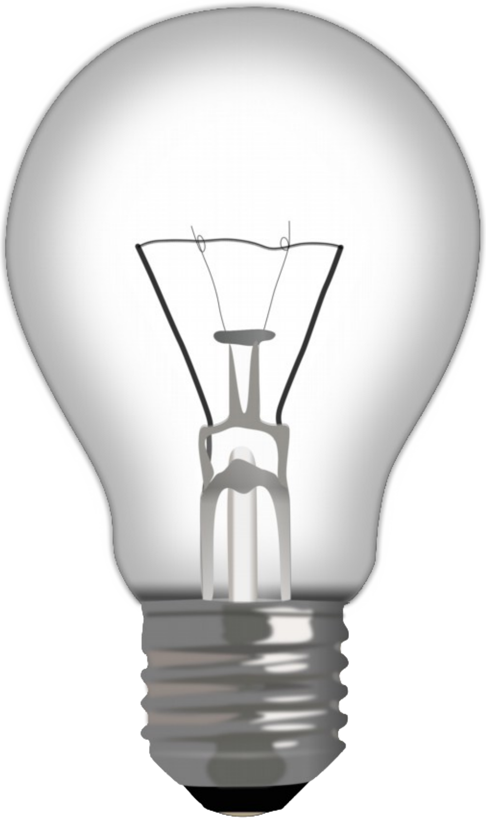
UNIVERSITÄT LEIPZIG

Institut für Informatik

ASV

Automatische Sprachverarbeitung

This lecture covers



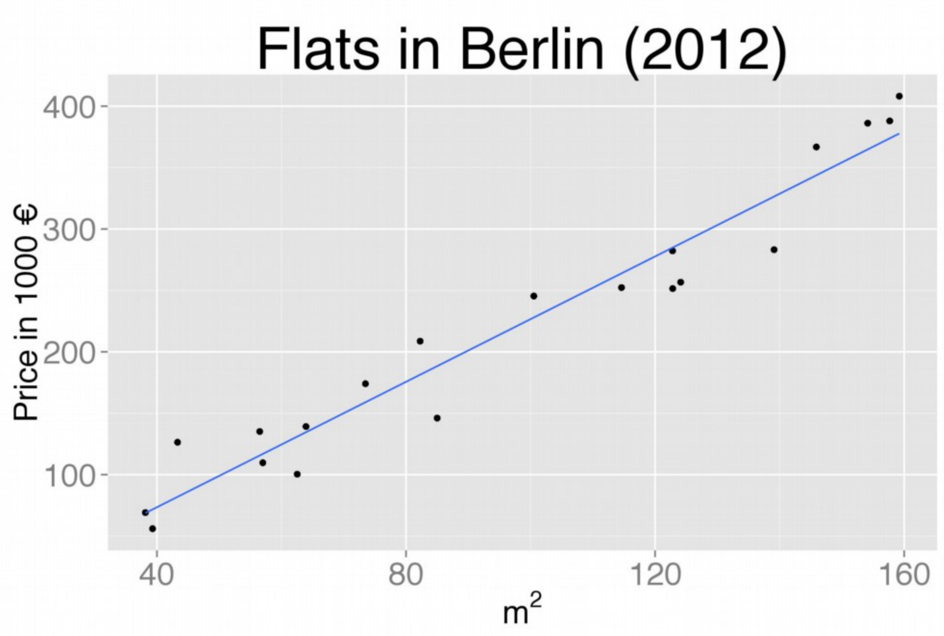
- Linear Regression
  - Hypothesis formulation, hypthesis space
- Optimizing Cost with Gradient Descent
- Using multiple input features with Linear Regression
- Feature Scaling
- Nonlinear Regression
- Optimizing Cost using derivatives

Linear Regression

UNIVERSITÄT LEIPZIG
Institut für Informatik

ASV
Automatische Sprachverarbeitung

Price for buying a flat in Berlin



- **Supervised learning problem**
  - **Expected answer available for each example in data**
- **Regression Problem**
  - **Prediction of continuous output**

Training data of flat prices

- **m** Number of training examples
- **x** is input (predictor) variable
  „features“ in ML-speek
- **y** is output (response) variable

Square meters	Price in 1000€
73	174
146	367
38	69
124	257
...	...

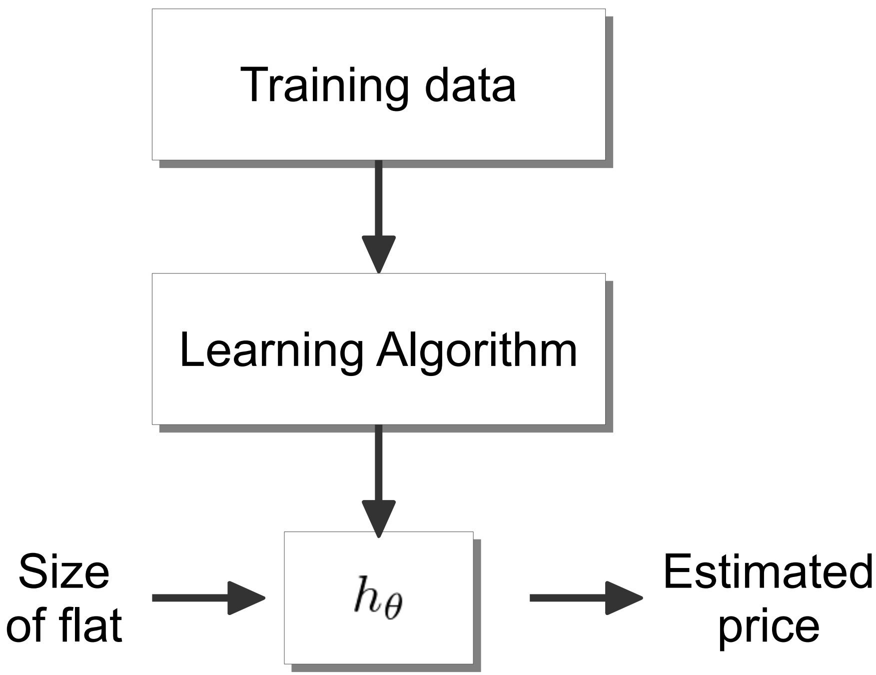
$(x^{(3)}, y^{(3)})$

$(x^{(4)}, y^{(4)})$

- Notation

$$(x, y) \quad - \quad \text{one training example}$$
$$(x^{(i)}, y^{(i)}) \quad - \quad i\text{th training example}$$

Learning procedure



hypothesis (mapping between input and output)

- **Hypothesis parameters**

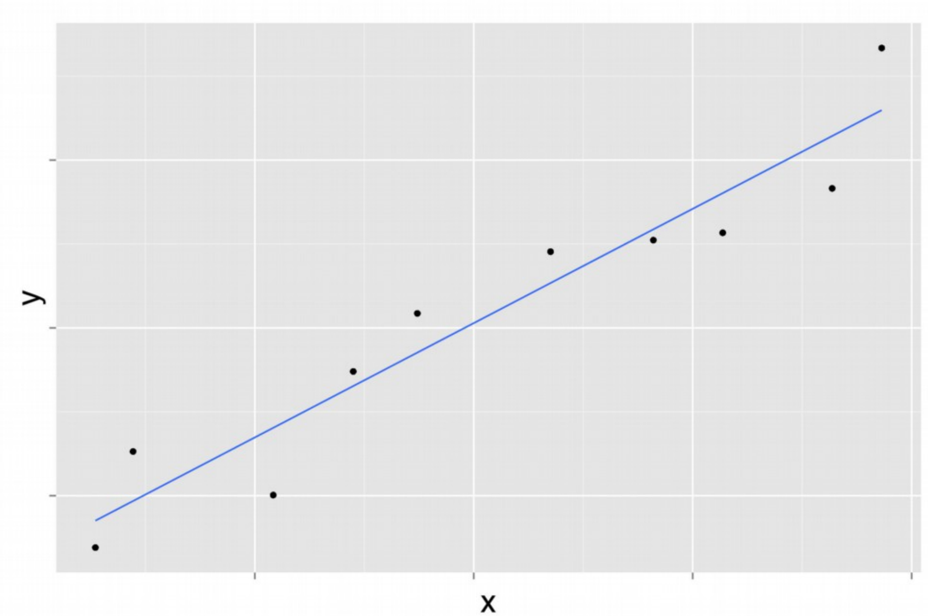
$$h_\theta(x) = \theta_0 + \theta_1 x$$

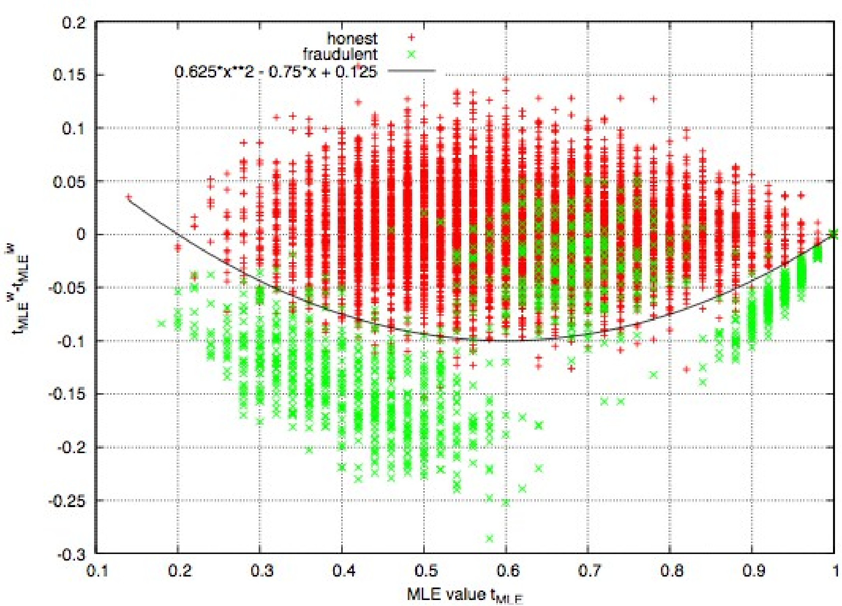
- **linear regression, one input variable (univariate)**

How to choose parameters?

Optimization objective

- **Purpose of learning algorithm expressed in *optimization objective* and *cost function* (often called *J*)**
  - **Fit data well**
  - **Few false positives**
  - **Few false negatives**
  - **...**





Fitting data well: least squares cost function

- **In regression almost always want to fit data well**
  - **smallest average distance to points in training data (*h(x)* close to *y* for *(x,y)* in training data)**
  - **Cost function often named *J***

Number of training instances

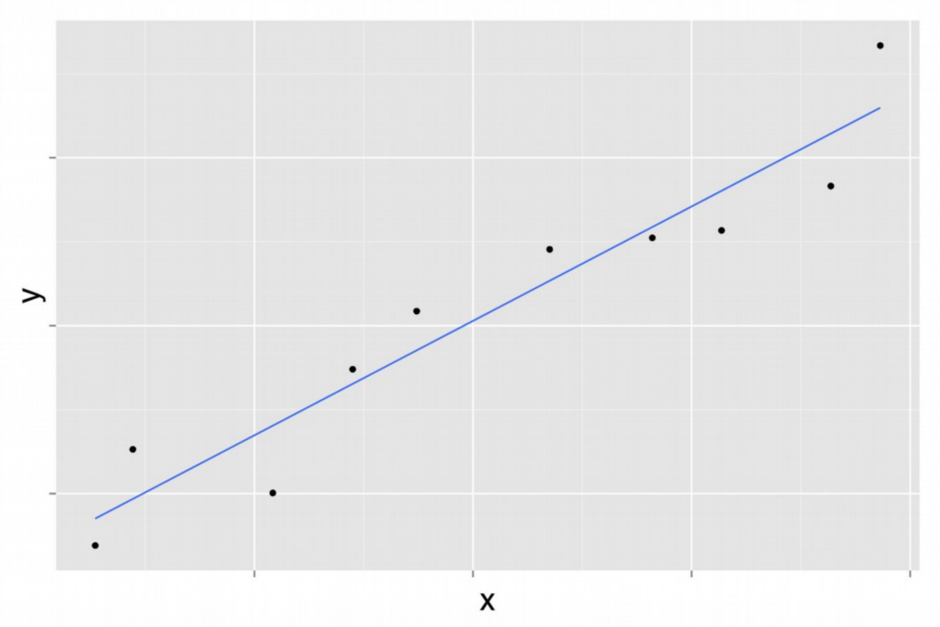
$$
\begin{aligned}
J(\theta_0, \theta_1) &= \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \\
&= \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)} \right)^2
\end{aligned}
$$

  - **Squaring**
    - **Penalty for positive and negative deviations the same**
    - **Penalty for large deviations stronger**

Optimizing Cost with Gradient Descent

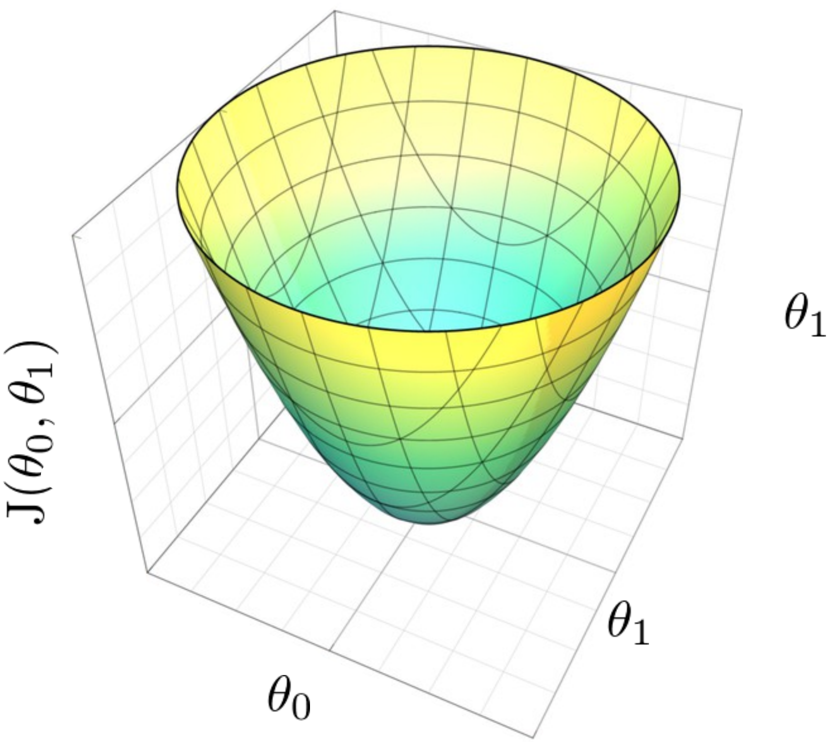
Gradient Descent Outline

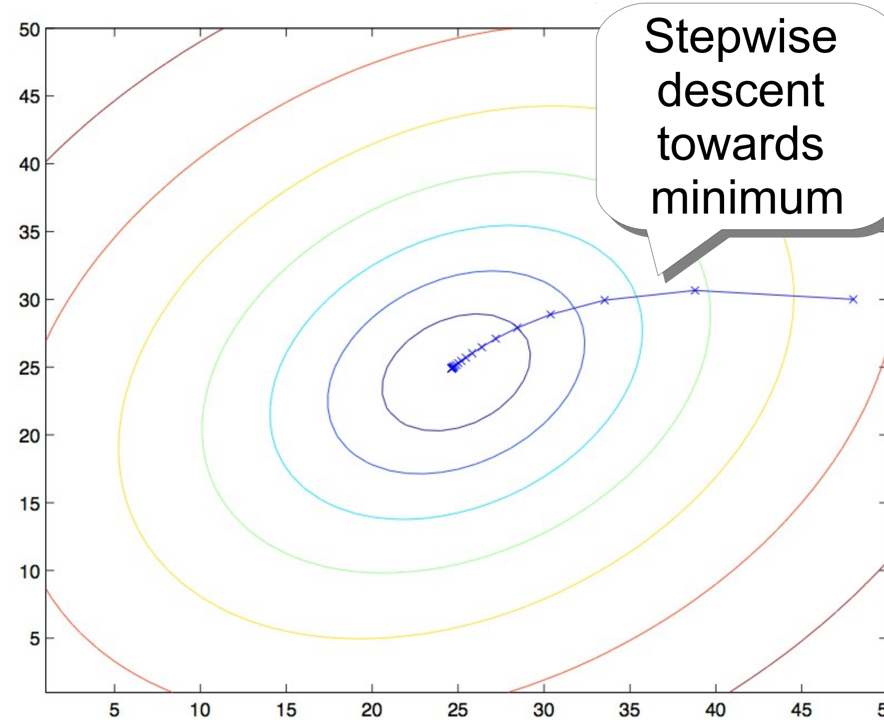
- **Want to minimize** $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)}\right)^2$
- **Start with random** θ_0, θ_1
- **Keep changing** θ_0, θ_1 **to reduce** $J(\theta_0, \theta_1)$ **until we end up at minimum**



3D plots and contour plots

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 + \theta_1 x^{(i)}) - y^{(i)} \right)^2$$





Stepwise descent towards minimum

[plot by Andrew Ng]

Derivatives work only for few parameters

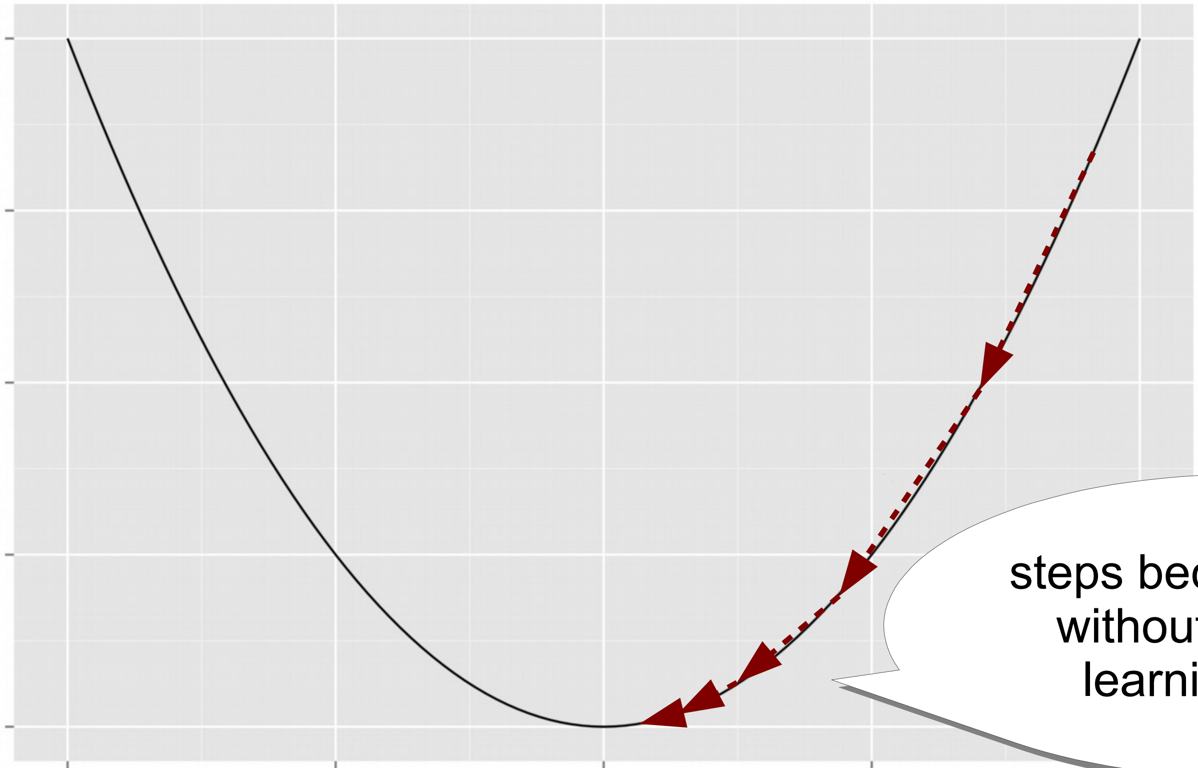
$$\underset{\theta_0 \theta_1}{minimize}\, J(\theta_0, \theta_1) = arg\ \underset{\theta}{min} J(\theta)$$

Gradient descent

$$
\begin{aligned}
&\textbf{while} \text{ not converged:} \\
&\quad \textbf{for} \text{ all } j: \\
&\qquad tmp_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\quad [\theta_0 \; \theta_1] := [tmp_0 \; tmp_1]
\end{aligned}
$$

partial derivative

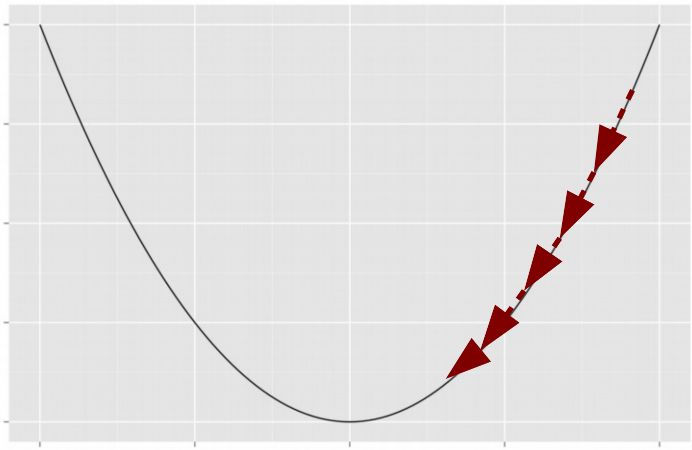
beware: incremental update incorrect!



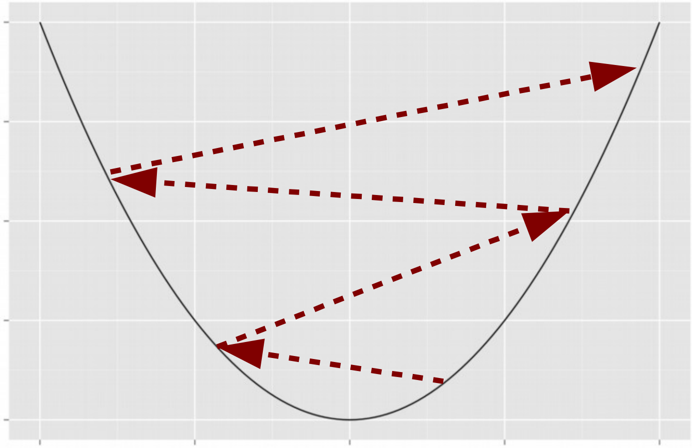
Learning Rate considerations

- **Small learning rate leads to slow convergence**

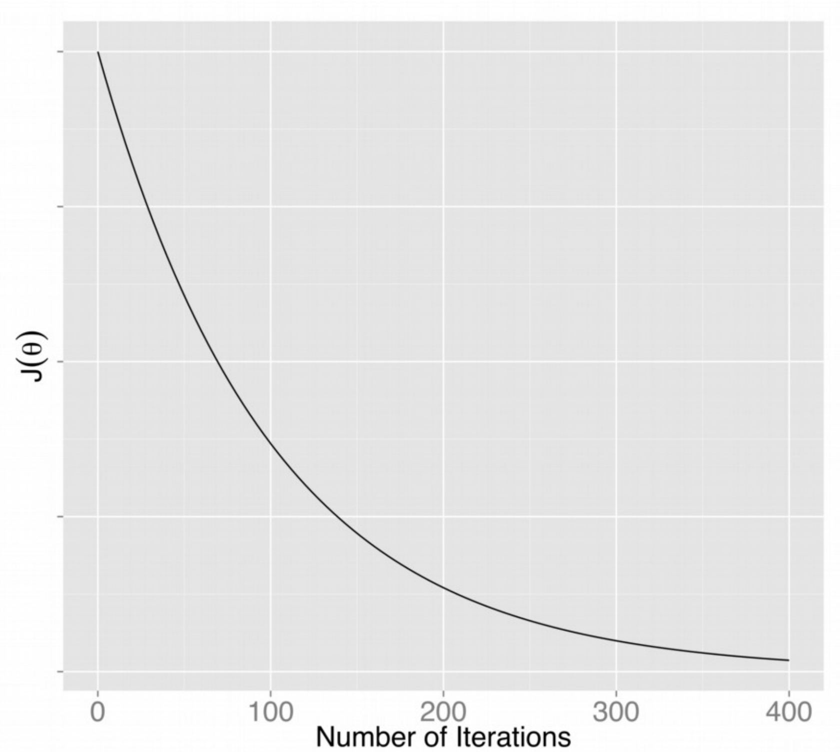
$$\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



- **Overly large learning rate may not lead to convergence or to divergence**
- **Often** $\alpha \in [0.001, 1]$



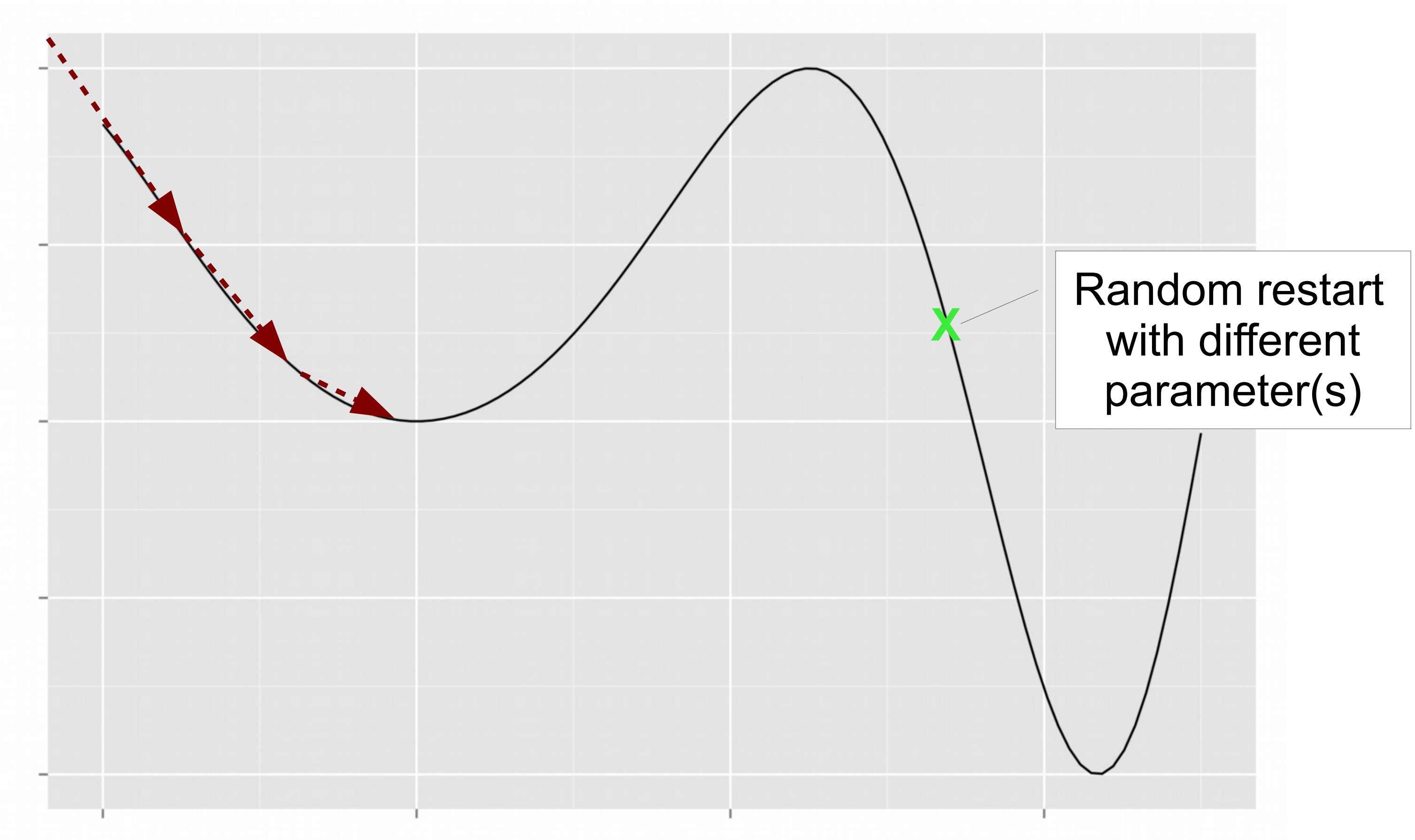
Checking convergence



- **Gradient descent works correctly if $\mathrm{J}(\theta)$ decreases with every step**
- **Possible convergence criterion: converged if $\mathrm{J}(\theta)$ decreases by less than constant ϵ**

Local Minima

- **Gradient descent can get stuck at local minima (e.g. *J* not squared error for regression with only *one* variable)**



Variants of Gradient Descent

Using multiple input features

Multiple features

Square meters	Bedrooms	Floors	Age of building (years)	Price in 1000€
x1	*x2*	*x3*	*x4*	*y*
200	5	1	45	460
131	3	2	40	232
142	3	2	30	315
756	2	1	36	178
…	…	…	…	…

- **Notation**

$$
\begin{array}{lll}
n & - & \text{number of features (here } n = 4) \\
x^{(i)} & - & \text{input features of } i\text{th training example} \\
x_j^{(i)} & - & \text{feature } j \text{ in } i\text{th training example}
\end{array}
$$

$$
x^{(3)} = \begin{bmatrix} 142 \\ 3 \\ 2 \\ 40 \end{bmatrix}
$$

$$
x_1^{(4)} = 756
$$

Hypothesis representation

- $\mathrm{h}_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- **More compact**

$$\mathrm{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$ **with definition** $\mathrm{x}_0 := 1$

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \ldots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \theta^T x$$

Gradient descent for multiple variables

- **Generalized cost function** $\mathrm{J}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- **Generalized gradient descent**

$$
\begin{aligned}
&\textbf{while}\ \text{not converged:}\\
&\quad \textbf{for}\ \text{all}\ j\text{:}\\
&\qquad tmp_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\\
&\quad \theta := \begin{bmatrix} tmp_0 \\ \vdots \\ tmp_n \end{bmatrix}
\end{aligned}
$$

Partial derivative of cost function for multiple variables

- **Calculating the partial derivative**

$$\begin{aligned}
\frac{\partial}{\partial\theta_j} J(\theta) &= \frac{\partial}{\partial\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{\partial}{\partial\theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left((\theta_0 x_0^{(i)} + \cdots + \theta_n x_n^{(i)}) - y^{(i)}\right)^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}
\end{aligned}$$

Gradient descent for multiple variables

- **Simplified gradient descent**

$$
\begin{array}{l}
\textbf{while not } \text{converged}: \\
\quad \textbf{for } \text{all } j: \\
\qquad tmp_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} \\
\quad \theta := \begin{bmatrix} tmp_0 \\ \vdots \\ tmp_n \end{bmatrix}
\end{array}
$$

Conversion considerations for multiple variables

- **With multiple variables, comparison of variance in data is lost (scales can vary strongly)**

Square meters	30 - 400
Bedrooms	1 - 10
Price	80 000 - 2 000 000

- **Gradient descent converges faster for features on similar scale**

Feature Scaling

Feature scaling

- **Different approaches for converting features to comparable scale**
  - **Min-Max-Scaling makes *all* data fall into range [0, 1]**

$$\mathrm{x}_j^{(i)\prime} := \frac{x_j^{(i)} - min(x_j)}{max(x_j) - min(x_j)}$$

(for single data point of feature *j*)

  - **Z-score conversion**

Z-Score conversion

- **Center data on 0**
- **Scale data so *majority* falls into range [-1, 1]**

$$\mu(x_j) = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

mean / empirical expected value (mu)

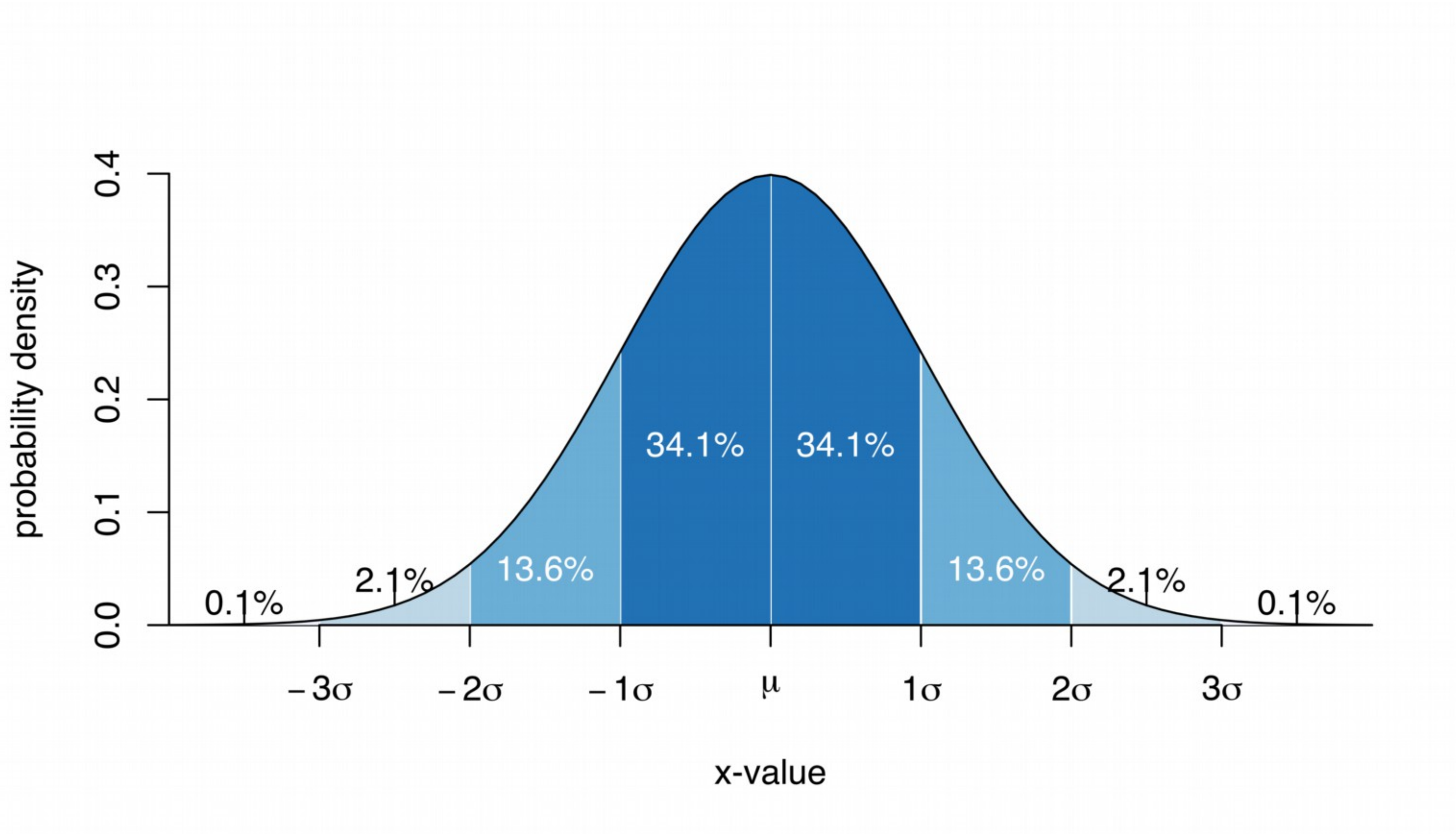
$$\sigma(x_j) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu(x_j))^2}$$

empirical standard deviation (sigma)

- **Z-score conversion of single data point for feature *j***

$$\mathrm{x}_j^{(i)\prime} := \frac{x_j^{(i)} - \mu(x_j)}{\sigma(x_j)}$$

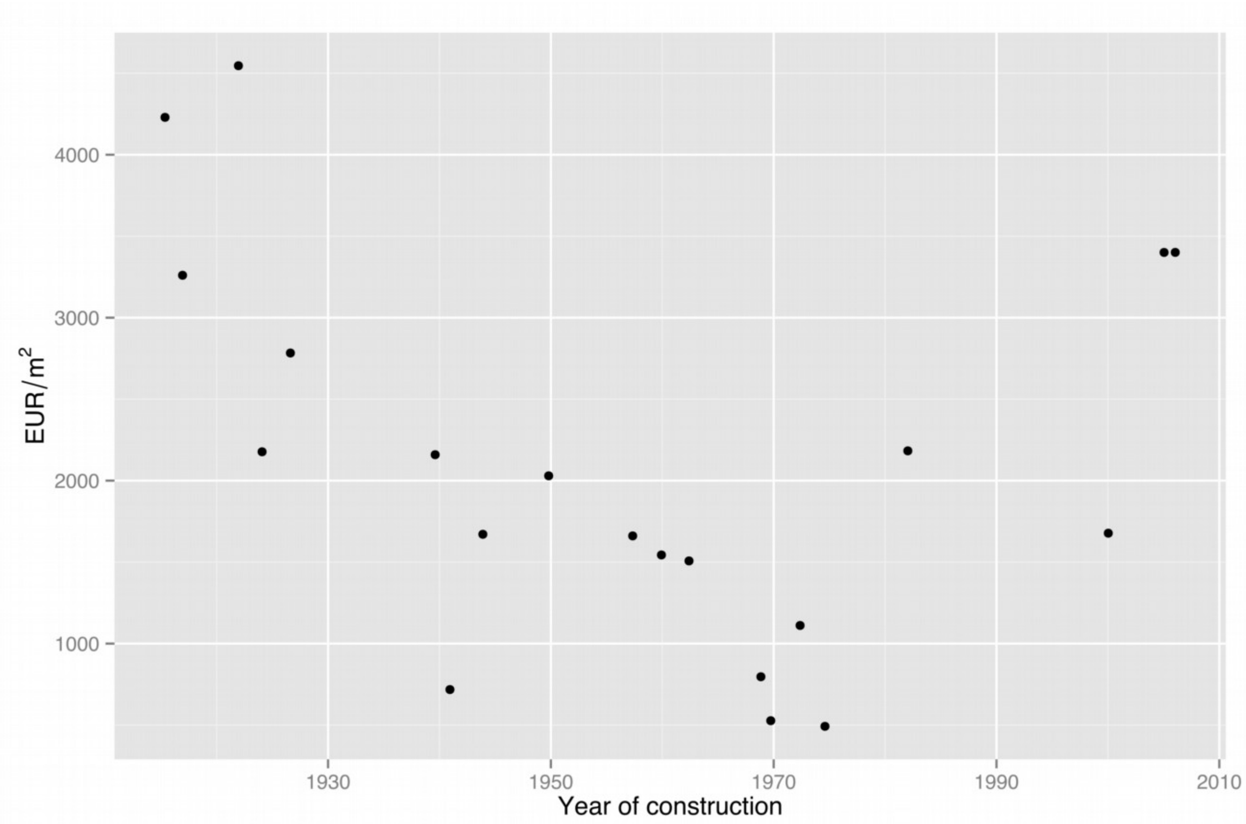
Visualizing standard deviation



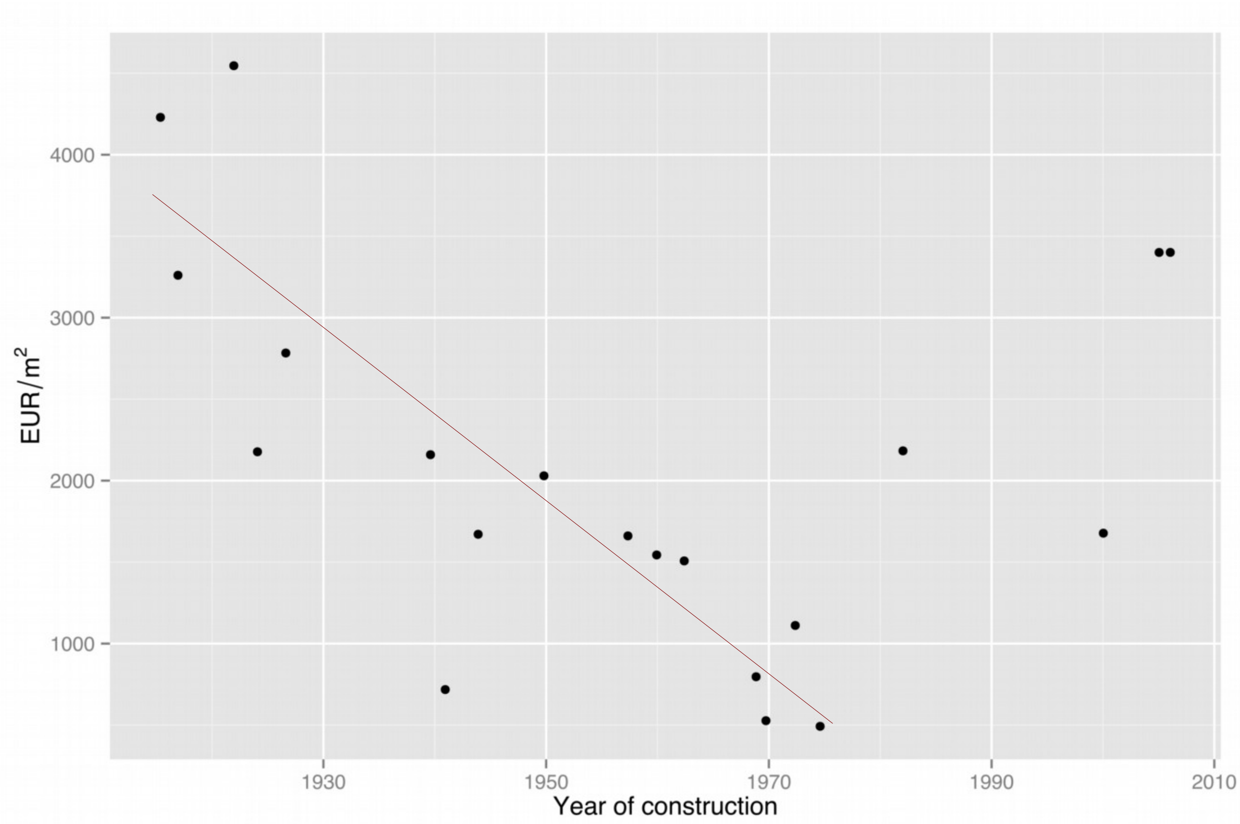
Nonlinear Regression (by cheap trickery)

Automatische Sprachverarbeitung

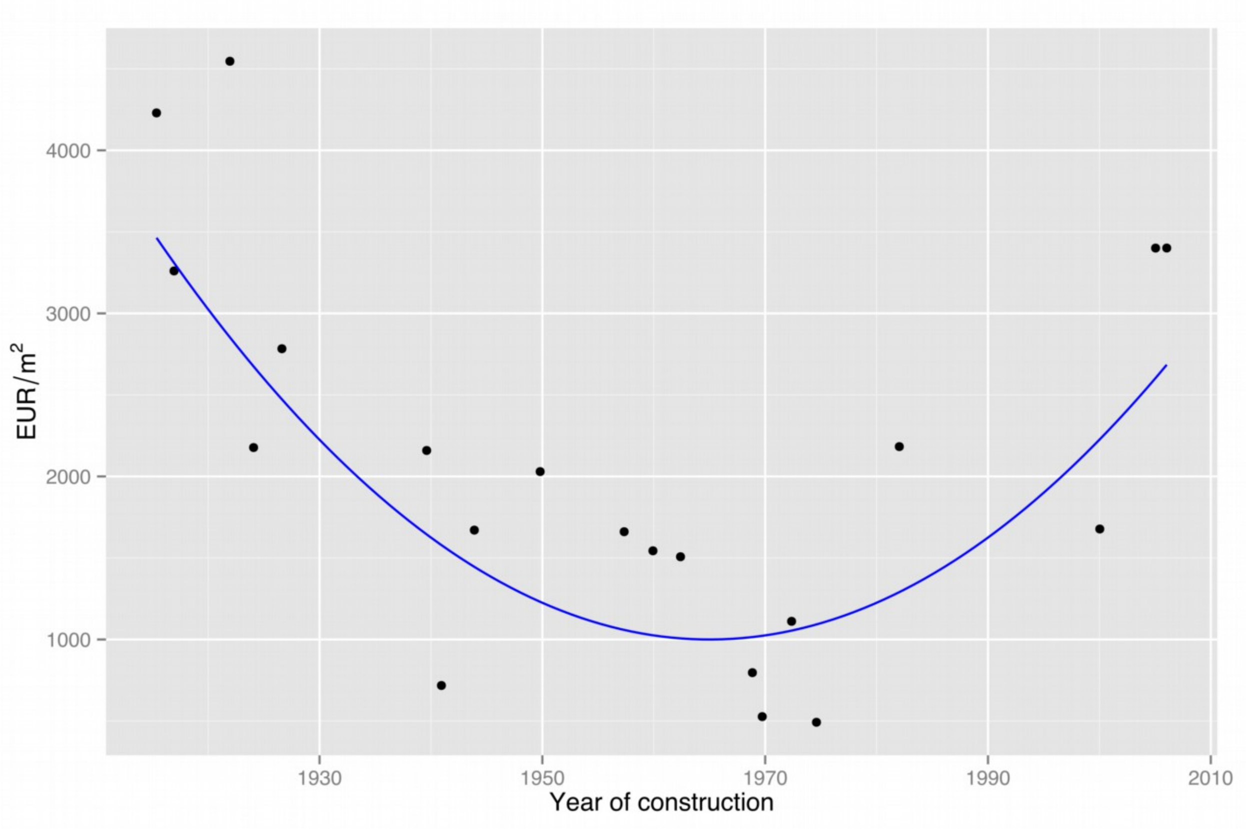
Nonlinear Regression Problems



Nonlinear Regression Problems (linear approximation)



Nonlinear Regression Problems (nonlinear hypothesis)



Nonlinear Regression with cheap trickery

- **Linear Regression can be used for Nonlinear Problems**
- **Choose nonlinear hypothesis space**
  - $\mathrm{h}_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \ldots$
  - $\mathrm{h}_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_2^3 + \theta_4 x_2^5 + \ldots$
  - $\mathrm{h}_\theta(x_1, \ldots, x_n) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_2} + \ldots$

Optimizing cost using derivatives

Comparison Gradient Descent vs. Setting derivative = 0

- **Instead of Gradient descent solve**

$$\frac{\partial}{\partial \theta_i} J(\theta) = 0$$

for all *i*

Comparison Gradient Descent vs. Setting derivative = 0

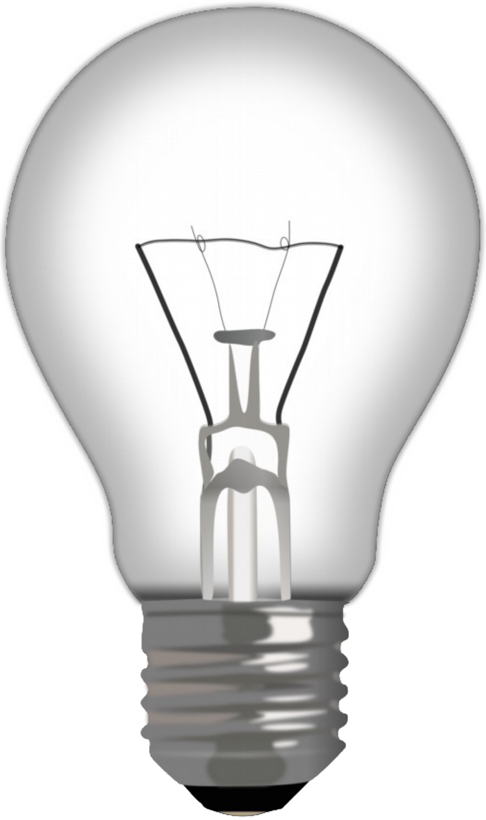
Gradient Descent

- **Need to choose α**
- **Needs many iterations, random restarts etc.**
- **Works well for many features**

Derivation

- **No need to choose α**
- **No iterations**
- $\mathrm{O}(\mathrm{n}^3)$
- **Slow for many features**

This lecture covers



- **Linear Regression**
  - **Hypothesis formulation, hypthesis space**
- **Optimizing Cost with Gradient Descent**
- **Using multiple input features with Linear Regression**
- **Feature Scaling**
- **Nonlinear Regression**
- **Optimizing Cost using derivatives**

Pictures

- Some public domain plots from en.wikipedia.org and de.wikipedia.org