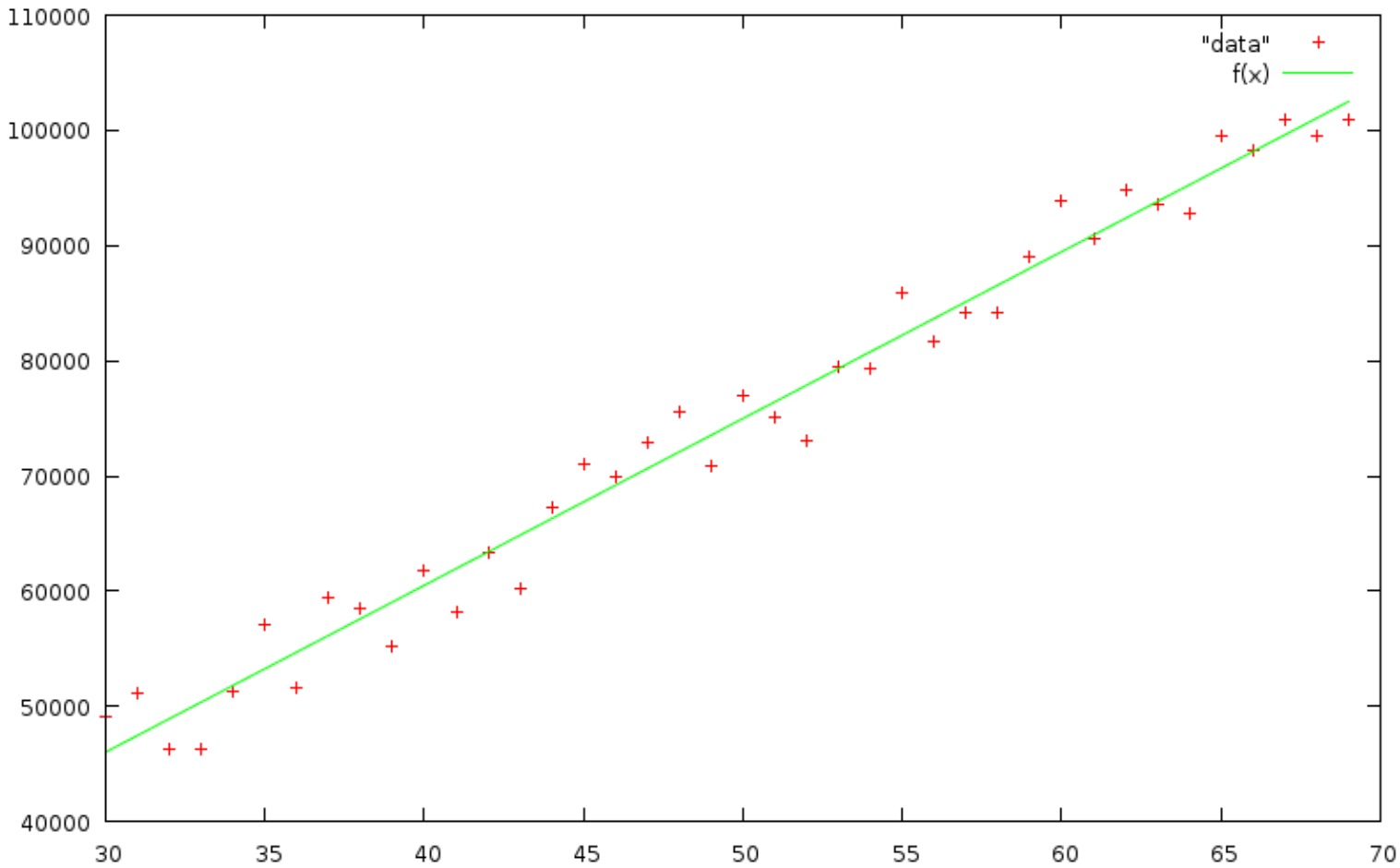
Simple linear regression

One attribute (flat size).
Hyphotesis set: $H = \{h_\Theta(x) = \Theta_0 + \Theta_1 x\}$
Error function: $\text{err}(y_p, y) = (y_p - y)^2$



Simple linear regression cont.

Looking for θ_0, θ_1, such that error is smallest as possible:

$$E(\theta_0, \theta_1) = \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)^2$$

Derivatives should be zero:

$$\frac{\partial E}{\partial \theta_0} = 0$$

$$\frac{\partial E}{\partial \theta_1} = 0$$

Example

Given training data:

x	y
3	6.5
4	7.9
5	9.9

Error would be:

$$E(\theta_0, \theta_1) = (\theta_0 + 3\theta_1 - 6.5)^2 + (\theta_0 + 4\theta_1 - 7.9)^2 + (\theta_0 + 5\theta_1 - 9.9)^2$$

Derivatives:

$$0 = \frac{\partial E}{\partial \theta_0} = 2(\theta_0 + 3\theta_1 - 6.5) + 2(\theta_0 + 4\theta_1 - 7.9) + 2(\theta_0 + 5\theta_1 - 9.9)$$

$$0 = \frac{\partial E}{\partial \theta_1} = 2(\theta_0 + 3\theta_1 - 6.5) \cdot 3 + 2(\theta_0 + 4\theta_1 - 7.9) \cdot 4 + 2(\theta_0 + 5\theta_1 - 9.9) \cdot 5$$

Example cont.

$$0 = \frac{\partial E}{\partial \theta_0} = 6\theta_0 + 24\theta_1 - 48.6$$

$$0 = \frac{\partial E}{\partial \theta_1} = 24\theta_0 + 100\theta_1 - 201.2$$

2 linear equations with 2 unknowns – boring and easy.

In general

$$E(\theta_0, \theta_1) = \sum_{i=1}^{n} (\theta_0 + \theta_1 x_i - y_i)^2$$

Derivatives:

$$0 = \frac{\partial E}{\partial \theta_0} = \sum_{i=1}^{n} 2(\theta_0 + \theta_1 x_i - y_i)$$

$$0 = \frac{\partial E}{\partial \theta_1} = \sum_{i=1}^{n} 2x_i(\theta_0 + \theta_1 x_i - y_i)$$

Generalization cont.

$$0 = \theta_0 n + \theta_1 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i$$

$$0 = \theta_0 \sum_{i=1}^{n} x_i + \theta_1 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i$$

$$0 = \theta_0 n \sum_{i=1}^{n} x_i + \theta_1 \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i$$

$$0 = \theta_0 n \sum_{i=1}^{n} x_i + \theta_1 n \sum_{i=1}^{n} x_i^2 - n \sum_{i=1}^{n} x_i y_i$$

Úpravy cont

$$0 = \theta_1 \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i - \theta_1 n \sum_{i=1}^{n} x_i^2 + n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i$$

$$\theta_1 = \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i - n \sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i - n \sum_{i=1}^{n} x_i^2}$$

From first equation:

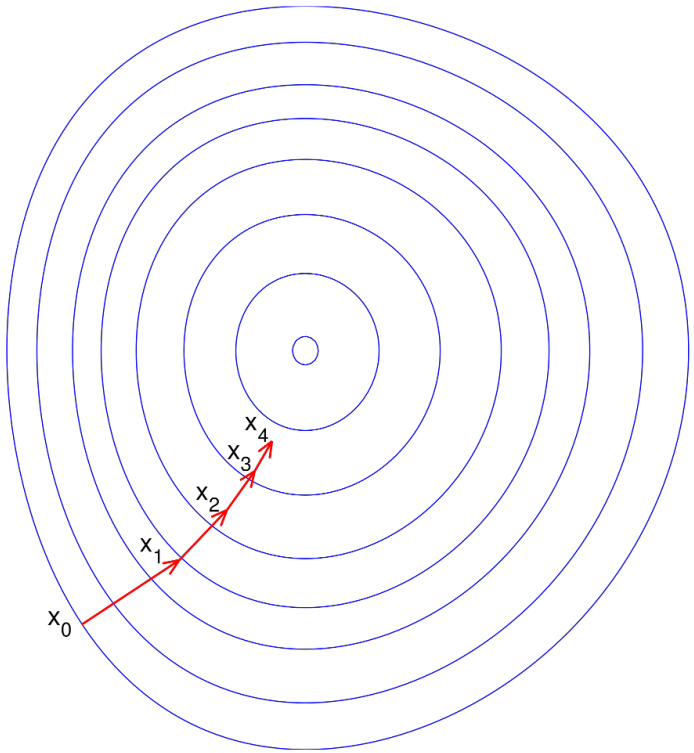
$$\theta_0 = \frac{1}{n} \left(\sum_{i=1}^{n} y_i - \theta_1 \sum_{i=1}^{n} x_i \right)$$

Other ways of minimalization

- Grid search
  - Try several grid spaced values. Zoom in.
  - Only for few variables.
- Numerical methods.

Numerical minimalization

Vector $\left(\frac{\partial E}{\partial \theta_0}, \frac{\partial E}{\partial \theta_1}\right)$ gives direction upwards (hovorí sa mu aj gradient).
Idea: Use gradient to move down.



Gradient descent

- $(\theta_0, \theta_1) =$ `Good initialization`
- `while (error changes):`
  - $\theta_0 = \theta_0 - \alpha \frac{\partial E}{\partial \theta_0}$
  - $\theta_1 = \theta_1 - \alpha \frac{\partial E}{\partial \theta_1}$

We need to pick α. Trial and error works well. Usual values $1, 0.1, 0.01, \ldots$. There are better ways.

Derivatives

Options:

- Manually
- Wolfram alpha
- Libraries, which do it for you (theano, tensorflow, pytorch, chainer). Keyword here is autograd.
- Numerical derivative
  - `scipy.optimize.approx_fprime`
  - $\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}$