POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Week 10-2

Optimization

Big Data

Prof. Hwanjo Yu
POSTECH

Optimization

- Basis for Modern Machine Learning Techniques
  - Artificial Neural Network and Deep Learning
  - Support Vector Machine
  - Matrix Factorization and Recommender System
  - ...

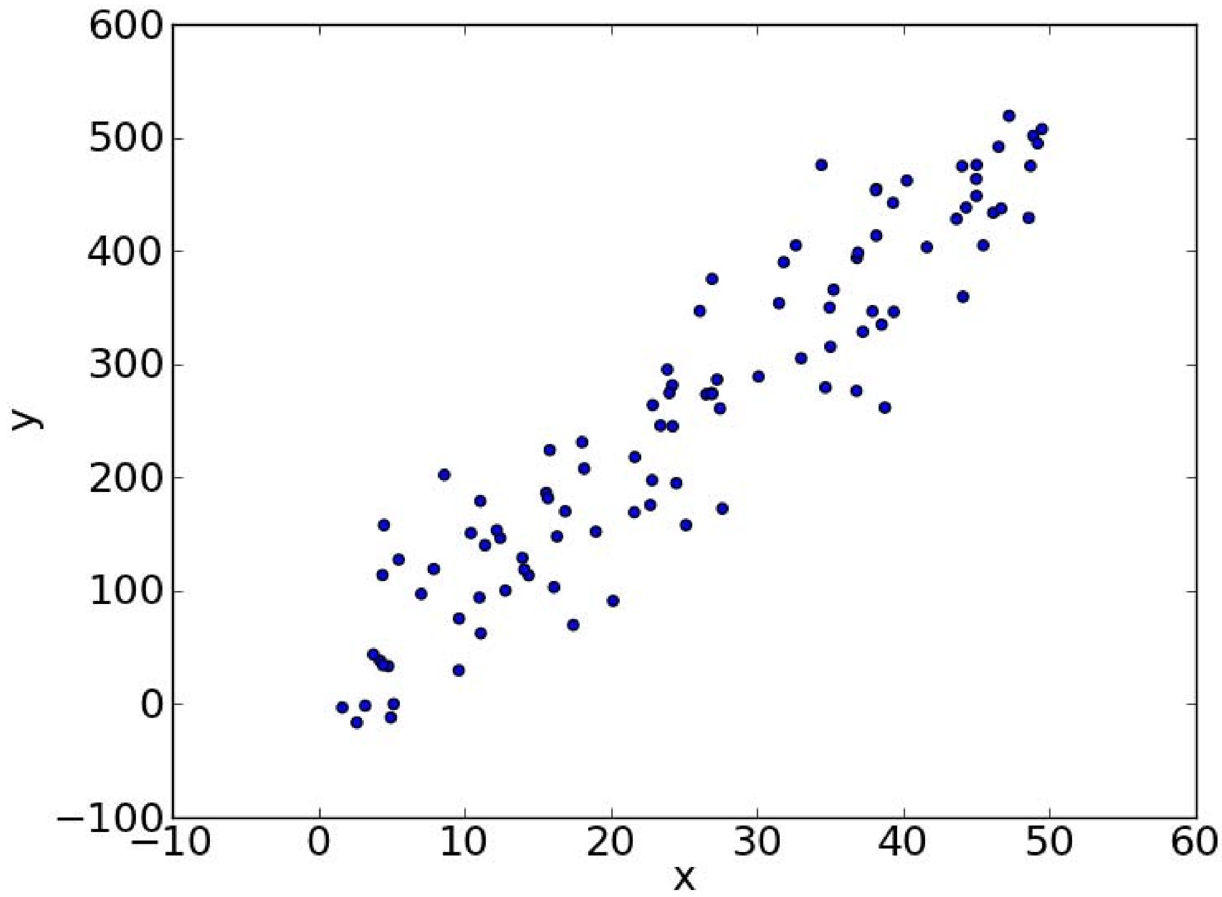
Optimization

- Gradient Descent and Stochastic Gradient Descent
  - Popularly used optimization methods
  - Basis for advanced optimization methods

Gradient descent in a nutshell

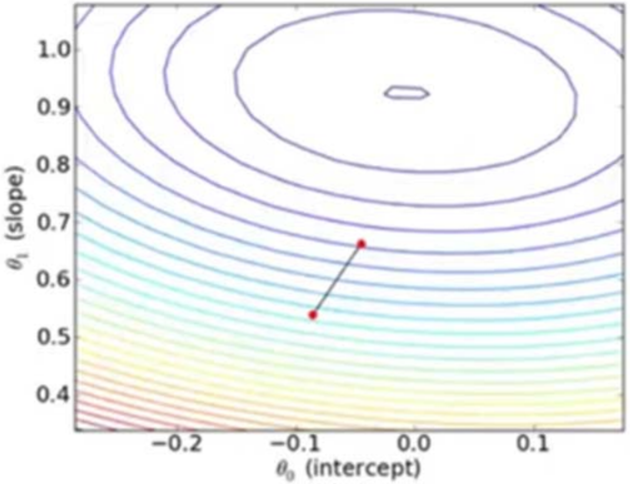
- Express your learning problem in terms of a cost function that should be minimized
- Starting at initial point, step “downhill” until you reach a minimum
- Some situations offer a guarantee that the minimum is the global minimum; others don’t

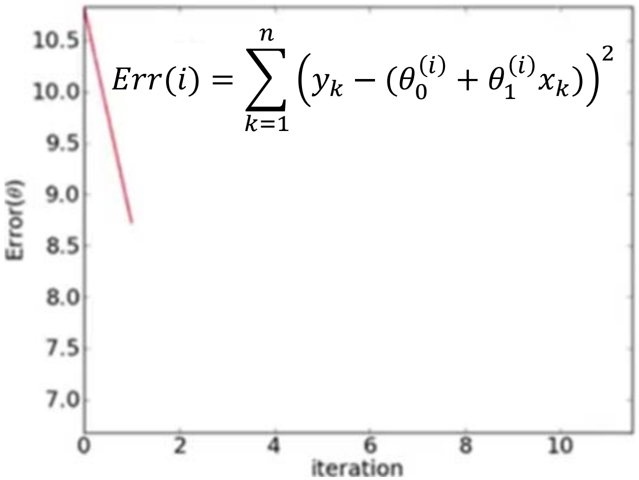
Gradient descent



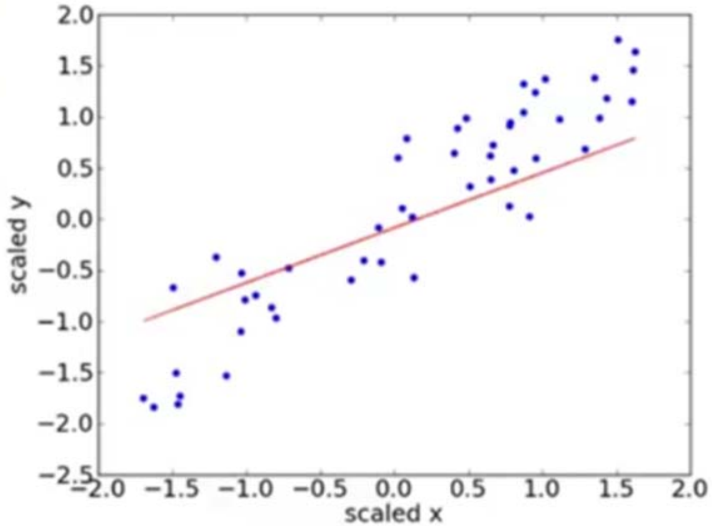
input variable	response variable
X	**y**
3.1	84.2
19.6	175.8
45.9	448.3
6.8	50.4
3.5	81.9
…	…

Gradient descent



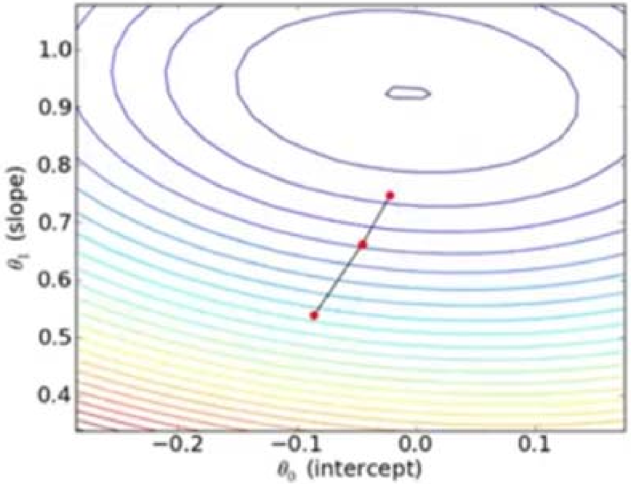


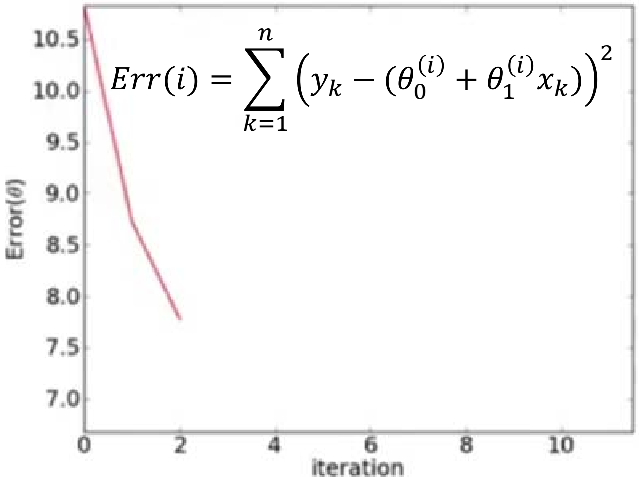
$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k) \right)^2$$



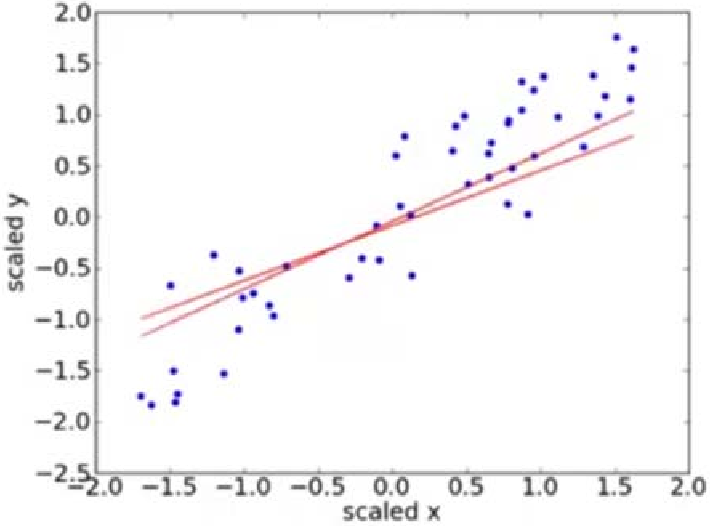
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

Gradient descent





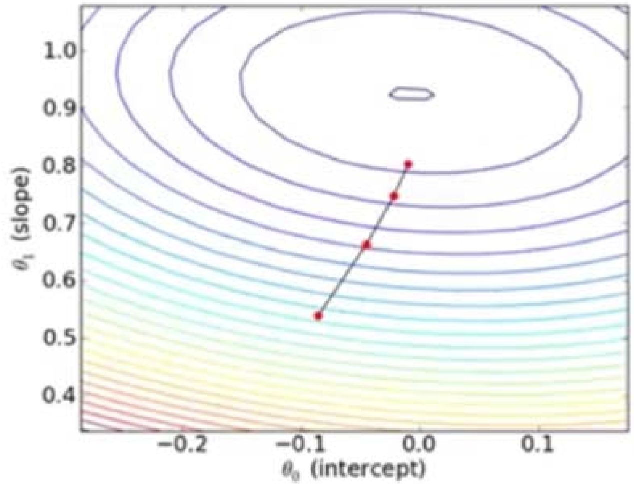
$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k)\right)^2$$

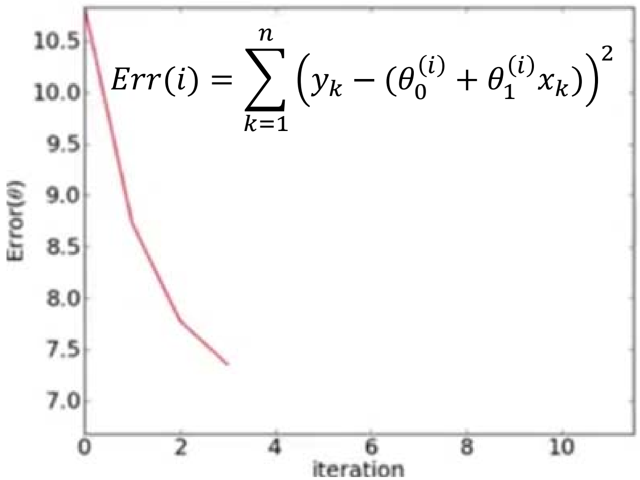


$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

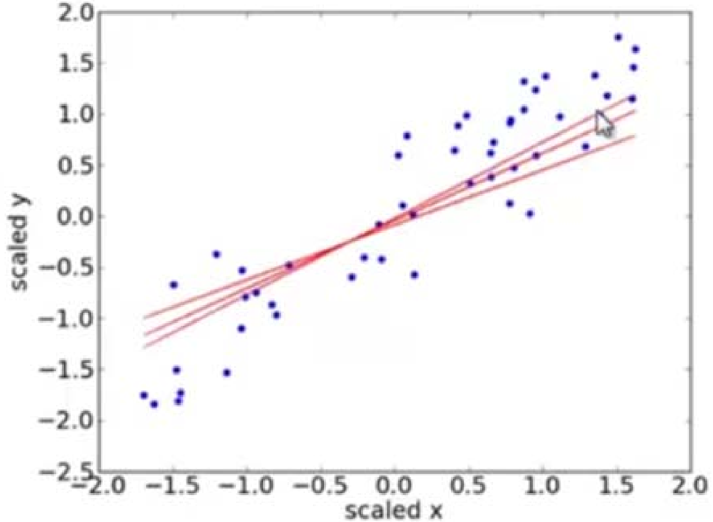
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Gradient descent





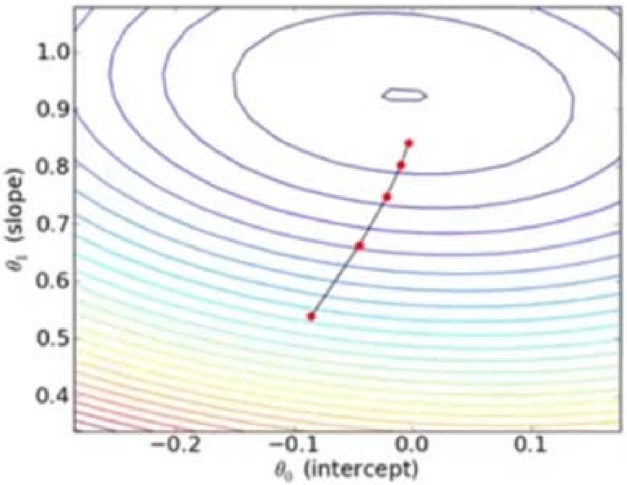
$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k)\right)^2$$

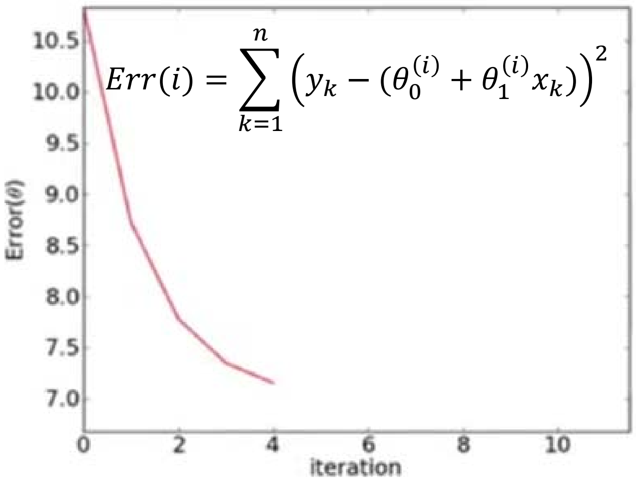


$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

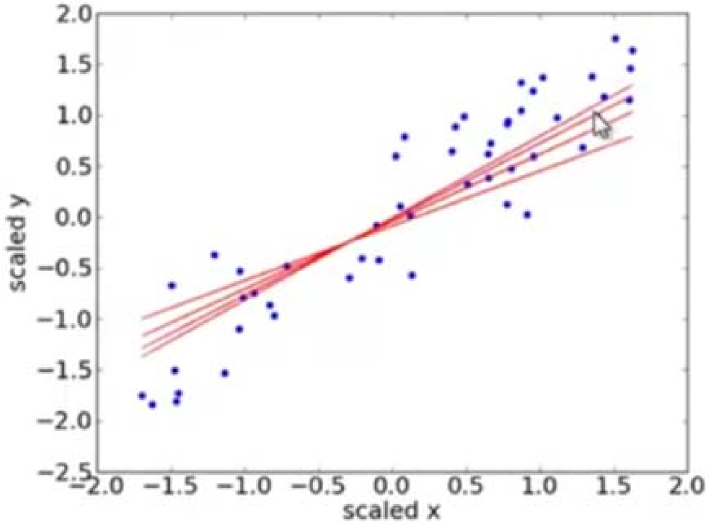
POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Gradient descent



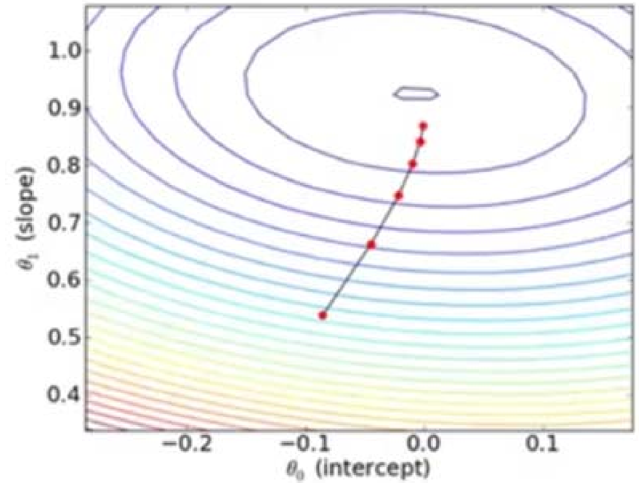


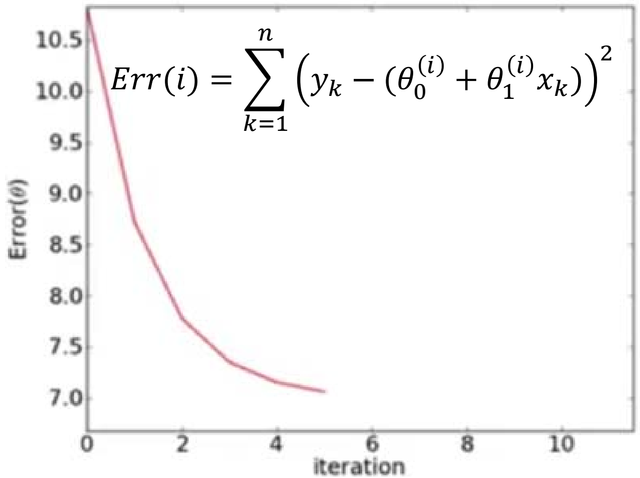
$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k) \right)^2$$



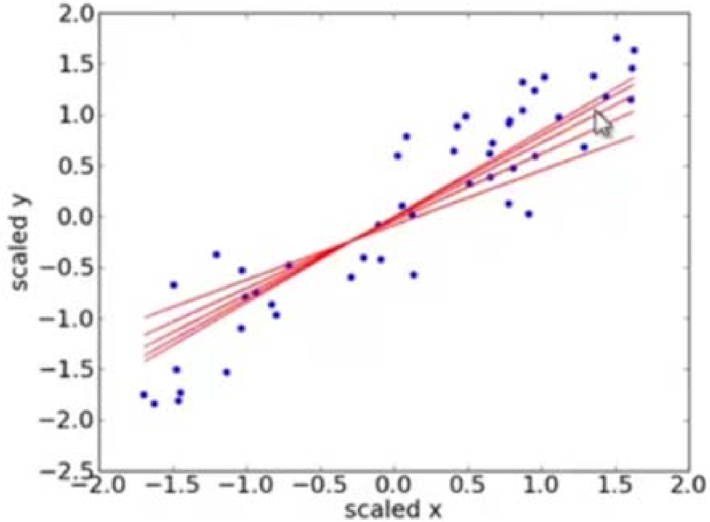
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

Gradient descent



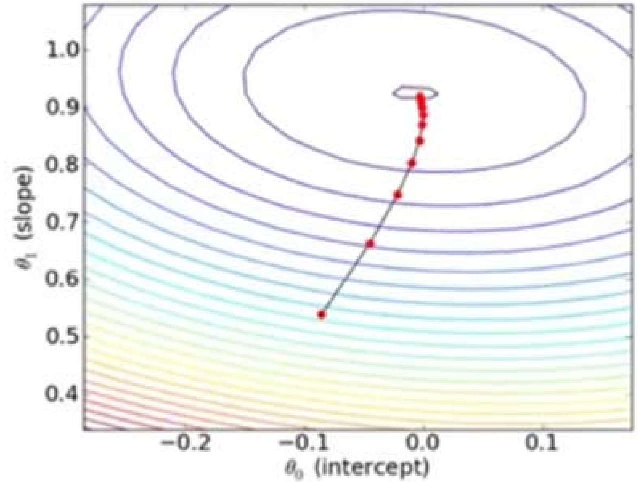


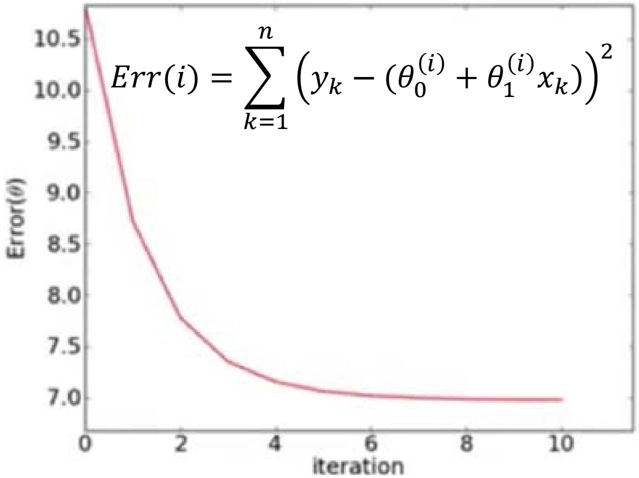
$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k) \right)^2$$



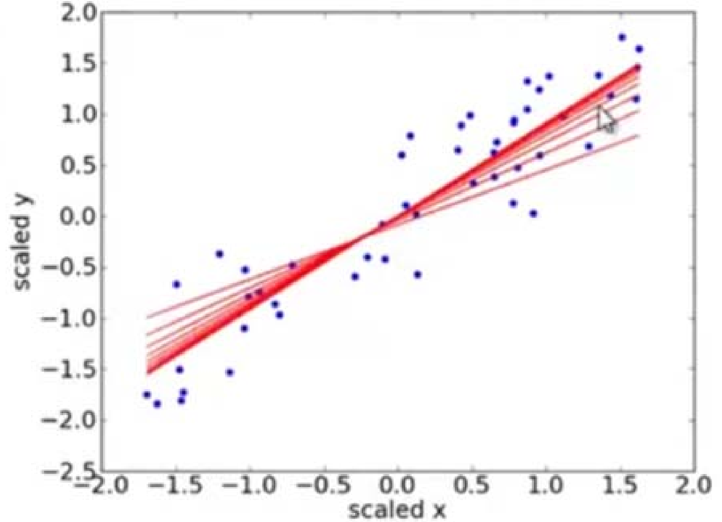
$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

Gradient descent



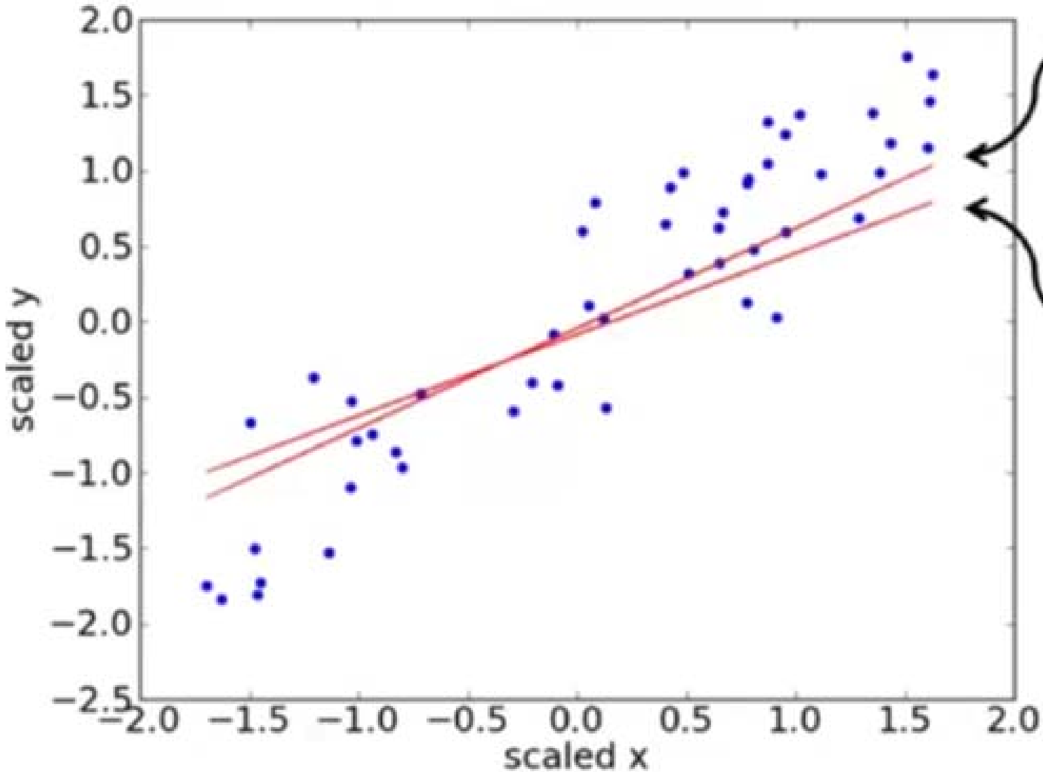


$$Err(i) = \sum_{k=1}^{n} \left(y_k - (\theta_0^{(i)} + \theta_1^{(i)} x_k) \right)^2$$



$$y = \theta_0^{(i)} + \theta_1^{(i)} x$$

Gradient descent



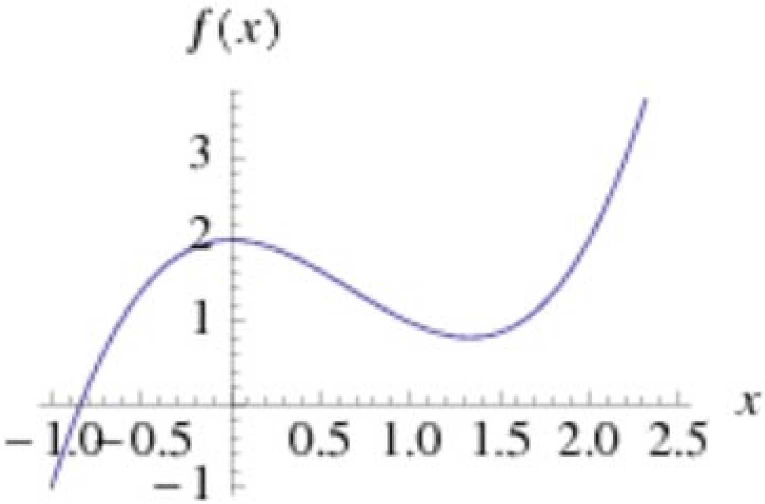
$$y = \theta_0^{(1)} + \theta_1^{(1)} x$$

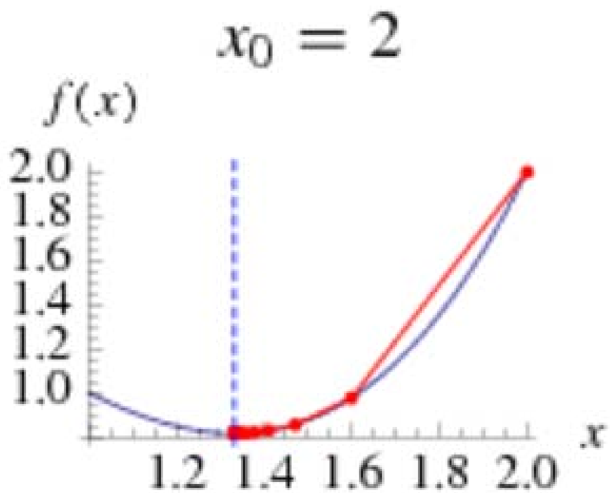
$$y = \theta_0^{(2)} + \theta_1^{(2)} x$$

Gradient descent

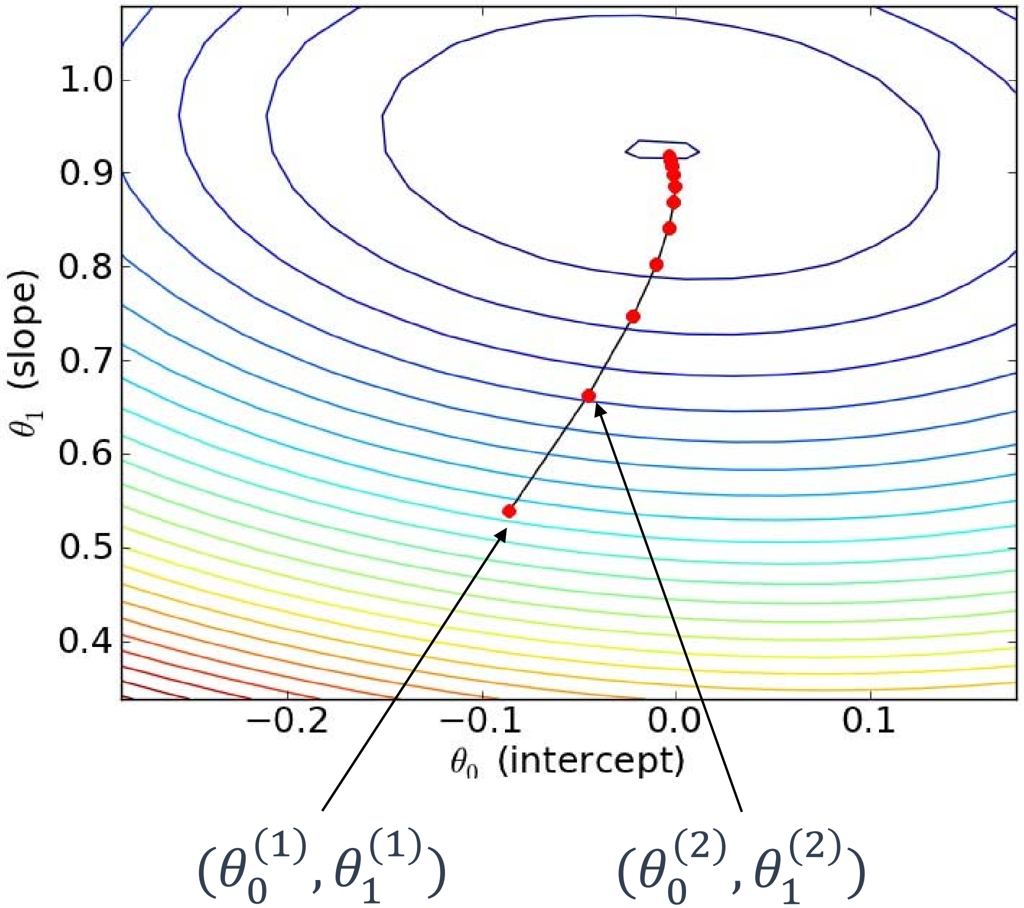
- $y = x^3 - 2x^2 + 2$

- To find a local minimum, $x_{i+1} = x_i - \alpha \frac{\partial y}{\partial x}$





Gradient descent



Learning Rate

Cost Function

$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} - \alpha \frac{\partial}{\partial \theta_0} J(\theta^{(i)})$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} - \alpha \frac{\partial}{\partial \theta_1} J(\theta^{(i)})$$

Model at *i*

where,

$$J(\theta^{(i)}) = \sum_{k=1}^{n} \left(h^{(i)}(x_k) - y_k\right)^2$$

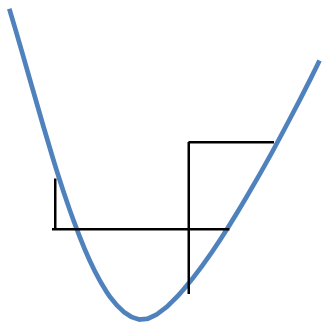
$$J(\theta^{(i)}) = \sum_{k=1}^{n} \left(\left(\theta_0^{(i)} + \theta_1^{(i)} x_k\right) - y_k\right)^2$$

$$\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} - \alpha \sum_{k=1}^{n} \left(\theta_0^{(i)} + \theta_1^{(i)} x_k - y_k\right)$$

$$\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} - \alpha \sum_{k=1}^{n} \left(\theta_0^{(i)} + \theta_1^{(i)} x_k - y_k\right) x_k$$

Gradient descent

- Initialization
  - Where do you drop the ball? "small random values"
- Step size
  - We don't really "roll", we "jump" in the direction of steepest descent
  - How far should we jump? α
  - Too far => you might hop over the minimum and raise the function value
  - Too small => slow convergence
- Momentum
  - $v_{i+1} \leftarrow \alpha \frac{\partial}{\partial \theta} J\left(\theta^{(i)}\right) - \beta v_i$
  - $\theta^{(i+1)} \leftarrow \theta^{(i)} - v_{i+1}$



What's the point?

$$\theta_j^{(i+1)} \leftarrow \theta_j^{(i)} - \alpha \frac{\partial}{\partial \theta_j} J\left(\theta^{(i)}\right)$$

Model parameters can be anything

Cost function can be anything*

*needs to be differentiable

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Other cost functions

- Logistic Regression

vector of weights

vector of instance data

Regularization term

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} log_2(1 + \exp(-y_i(\theta \cdot x_i))) + \frac{\lambda}{2}||\theta||^2$$

- Support Vector Machines

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} max(1 - y_i(\theta \cdot x_i), 0) + \frac{\lambda}{2}||\theta||^2$$

Quick intuition for regularization

- As one weight goes up, another goes down to compensate.
- And so weights may explode – overfitting again
- Need to enforce some condition on the weights to prefer simple models.
- The regularization term provides this balance

Aside on norms

- Norm: any function that assigns a strictly positive number to every non-zero vector

$$L^p - norm = \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$$

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Aside on cost functions

- Not a norm : $\sum_i H_i - H_i'$
  - errors cancel out
- L1-norm : $\sum_i |H_i - H_i'|$
  - assert that 1 error of 7 units is as bad as 7 errors of 1 unit each
- L2-norm : $\sqrt{\sum_i \left(H_i - H_i'\right)^2}$
  - assert that 1 error of 7 units is as bad as 49 errors of 1 unit each

Back to regularization

- **"LASSO"** : Regularized Least Squares with **L1-norm**
  - $J(\theta) = \sum_{k=1}^{n} (h(x_k) - y_k)^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} ||\boldsymbol{\theta}||_{\mathbf{1}}$
- **"Ridge Regression"** : Regularized Least Squares with **L2-norm**
  - $J(\theta) = \sum_{k=1}^{n} (h(x_k) - y_k)^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} ||\boldsymbol{\theta}||_{\mathbf{2}}^{\mathbf{2}}$

Back to gradient descent

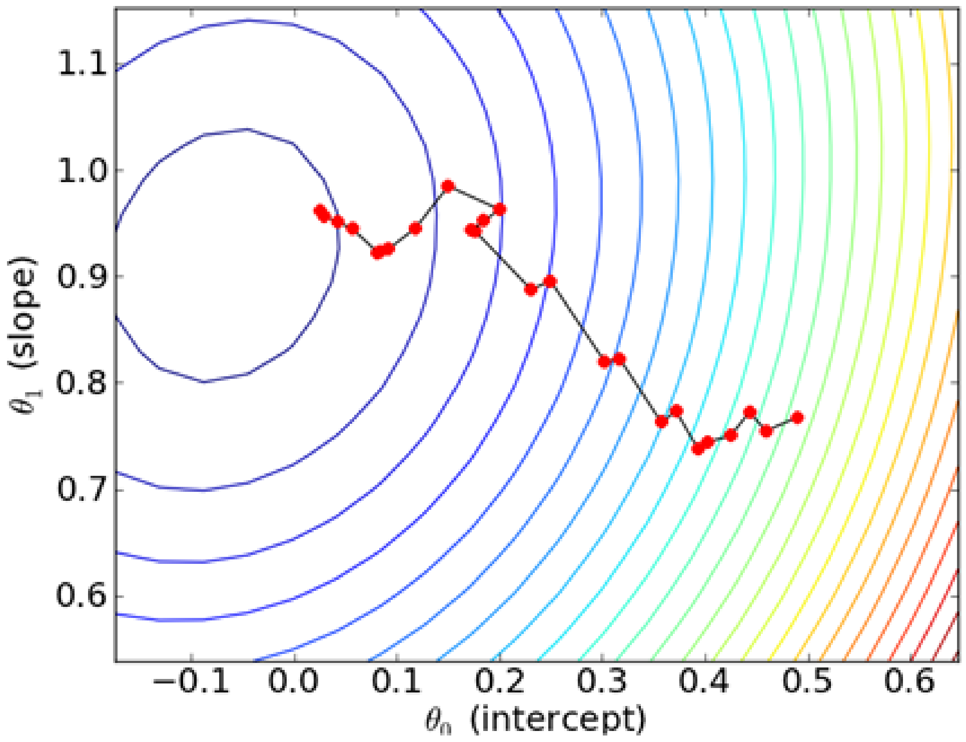
- Process the entire dataset on every iteration
  - $\theta_0^{(i+1)} \leftarrow \theta_0^{(i)} - \alpha \sum_{k=1}^{n} \left(\theta_0^{(i)} + \theta_1^{(i)} x_k - y_k\right)$
  - $\theta_1^{(i+1)} \leftarrow \theta_1^{(i)} - \alpha \sum_{k=1}^{n} \left(\theta_0^{(i)} + \theta_1^{(i)} x_k - y_k\right) x_k$

- Stochastic Gradient Descent (SGD)
  - At each step, pick **one random data point**
  - Continue as if your entire dataset was just the one point

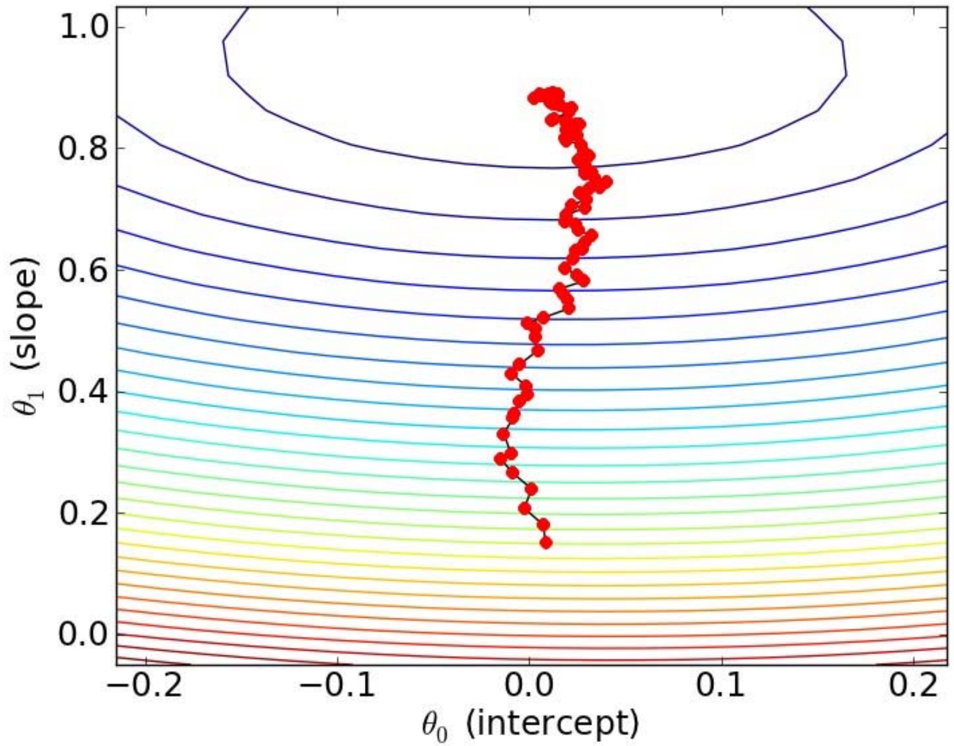
- Minibatch Gradient Descent
  - At each step, pick **a small subset of data points**
  - Continue as if your entire dataset just this subset

Back to gradient descent

Example using Stochastic Gradient Descent



Example using Minibatches



Parallel stochastic gradient descent

- Stochastic Gradient Descent (SGD)
  - At each step, pick **one random data point**
  - Continue as if your entire dataset was just the one point
- Parallel Stochastic Gradient Descent
  - In each of k threads, pick a random data point
  - Compute the gradient and update the weights
  - Weights will be "mixed"
    => converging zig-zag
  - How about forcing "strong consistency" by locking?

Thread 1	Thread 2
$\theta_0^{(1)} \leftarrow \theta_0^{(0)} - \left(\theta_0^{(0)} + \theta_1^{(0)} x_3 - y_3\right)$	
	$\theta_0^{(2)} \leftarrow \theta_1^{(1)} - \left(\theta_0^{(1)} + \theta_1^{(0)} x_8 - y_8\right)$
$\theta_1^{(1)} \leftarrow \theta_1^{(0)} - \left(\theta_0^{(2)} + \theta_1^{(0)} x_3 - y_3\right) x_3$	
	$\theta_1^{(2)} \leftarrow \theta_1^{(1)} - \left(\theta_0^{(2)} + \theta_1^{(1)} x_8 - y_8\right) x_8$
$\theta_0^{(3)} \leftarrow \theta_0^{(2)} - \left(\theta_0^{(2)} + \theta_1^{(2)} x_5 - y_5\right)$	
	$\theta_0^{(4)} \leftarrow \theta_0^{(3)} - \left(\theta_0^{(3)} + \theta_1^{(2)} x_9 - y_9\right)$
$\theta_1^{(3)} \leftarrow \theta_1^{(2)} - \left(\theta_0^{(4)} + \theta_1^{(2)} x_5 - y_5\right) x_5$	
	$\theta_1^{(4)} \leftarrow \theta_1^{(3)} - \left(\theta_0^{(4)} + \theta_1^{(3)} x_9 - y_9\right) x_9$