Gradient Descent for Multiple Linear Regression

Summary and Motivation

- We have seen the ordinary least squares solution to find the regression parameters.

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- When we want a model that is adaptable (we may accumulate more data over time etc), a common approach is to use gradient descent.

Gradient Descent

- Gradient descent (also called Steepest Descent) is an important optimization technique that is used in many machine learning algorithms.

- **Iterative** technique:
  - Start with a random vector **w**
  - Iteratively improve **w** to minimize the error.

- In general, it can be used to minimize/maximize a performance metric (here the error) with respect to the parameters (here the weights)

Multiple Linear Regression

- Function y() is a linear combination of input features

$$y(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p$$

- In vector notation with extended input vector **x** (x_0=1):

$$y(\mathbf{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + \ldots + w_p x_p = w^{\mathbf{T}} \mathbf{x}$$

- **Data**: $D = \{(\mathbf{x}_i, y_i)\}$

- **Hypothesis**: $\mathbf{x}_i \to f(\mathbf{x}_i)$

  We would like to have $y_i \approx f(\mathbf{x}_i)$ for all $i = 1 \ldots n$

- **Error function:**

  Mean Squared Error $J = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2$

- **Learning:** Weights that minimizes the error

Gradient Descent

- Start with some (random) weights
- Adjust weights in the direction that most reduces the error
- The gradient points in the direction of maximum change
  - Go in the opposite direction of the gradient

$$\mathbf{w} \leftarrow \mathbf{w} - \lambda \nabla_{\mathbf{w}} Error$$

  - λ is called the learning rate (a small positive number, like 0.01) that determines the step size in the chosen direction (reverse of the gradient)

Gradient Vector

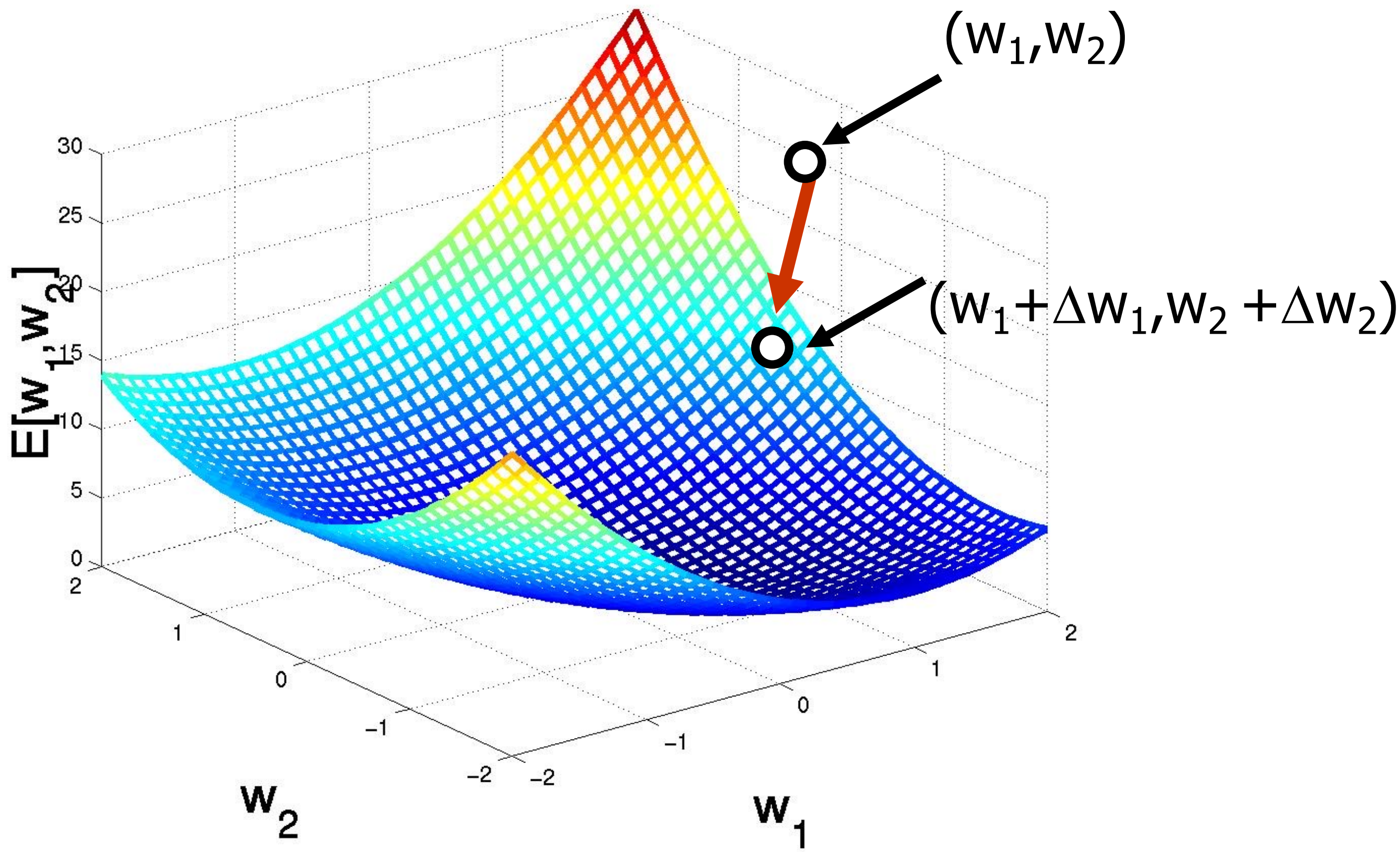
Each dimension of the gradient vector is the vector of partial derivatives of the function with respect to each of the dimensions.

$$\nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial}{\partial w_0} E(\mathbf{w}) \\ \frac{\partial}{\partial w_1} E(\mathbf{w}) \\ \vdots \\ \frac{\partial}{\partial w_n} E(\mathbf{w}) \end{bmatrix}$$

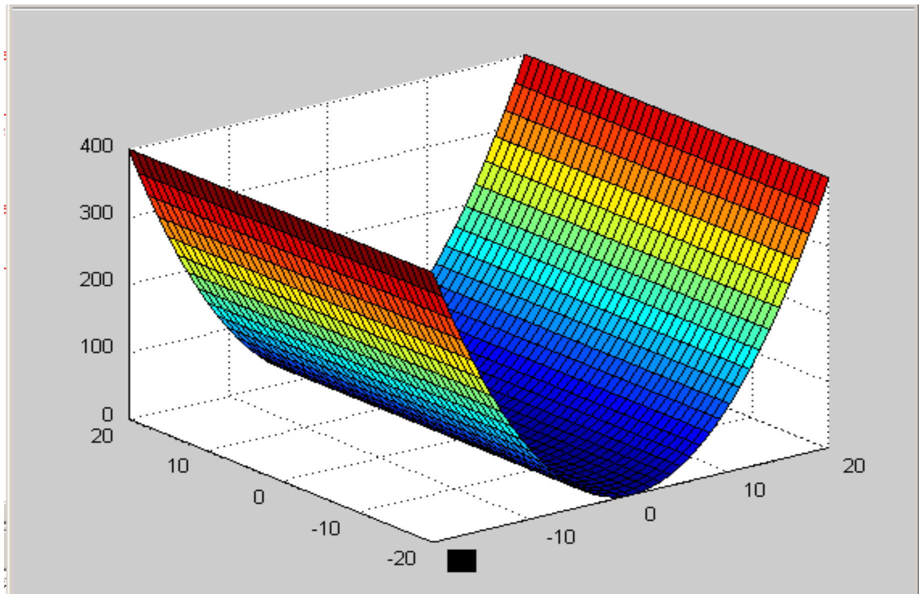
Gradient Descent with Multiple Weights

Gradient:
$\nabla E[w]=[\partial E/\partial w_0,\ldots\ \partial E/\partial w_n]$

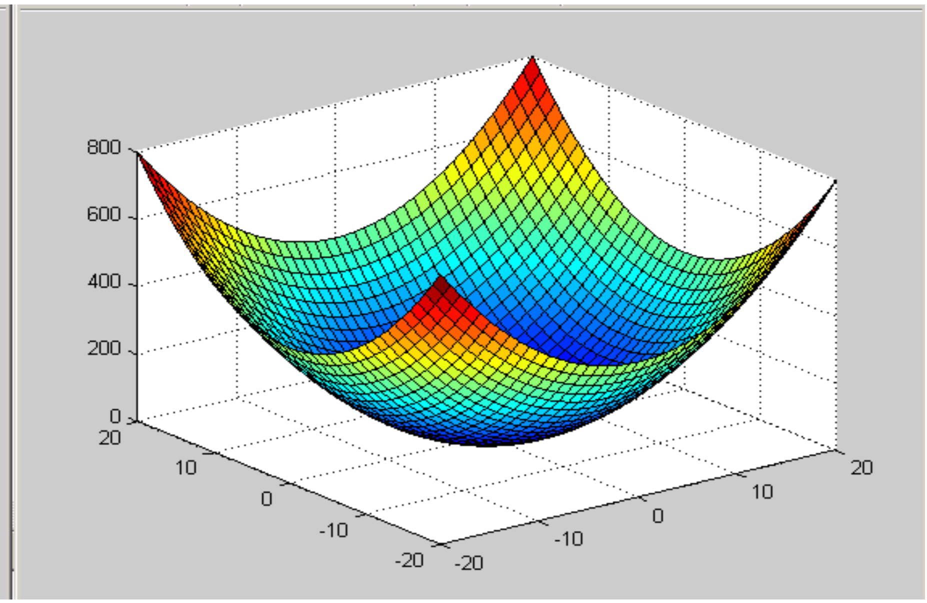
$\Delta w=-\eta\ \nabla E[w]$



Two simple error surfaces (for 2 weights)



a)



b)

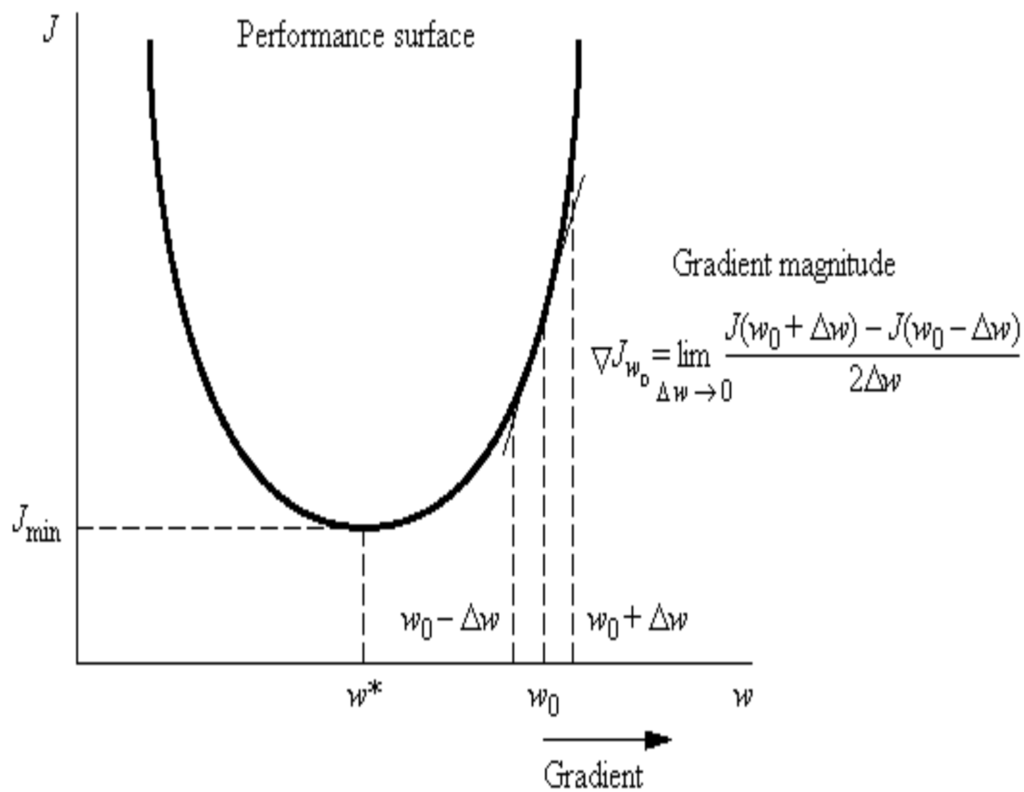
In a) as you move parallel to one of the axis, there is no change in error

In b) moving diagonally, rather than parallel to the axes, brings the biggest change.

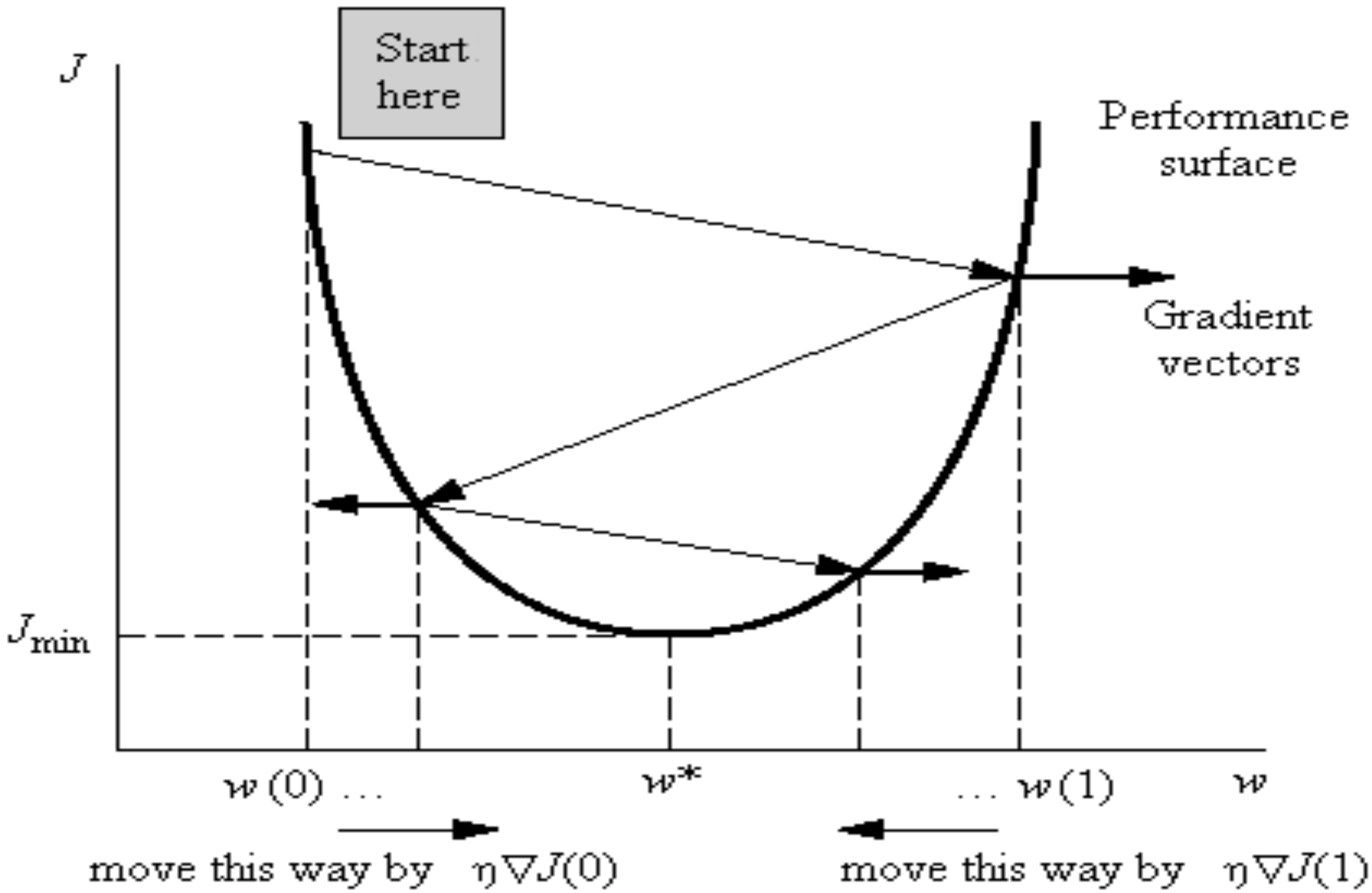
Gradient Descent

- The **gradient** of the performance surface is a **vector**
- (with the dimension of w) that:
  - **points toward the direction of maximum change,**
  - **with a magnitude equal to the slope of the tangent of the performance surface.**

- A ball rolling down the hill will always attempt to roll in the direction **opposite to the gradient** (steepest descent).
  - The slope at the bottom is zero, so the gradient is also zero (that is the reason the ball stops there).



Steepest Descent



Gradient descent

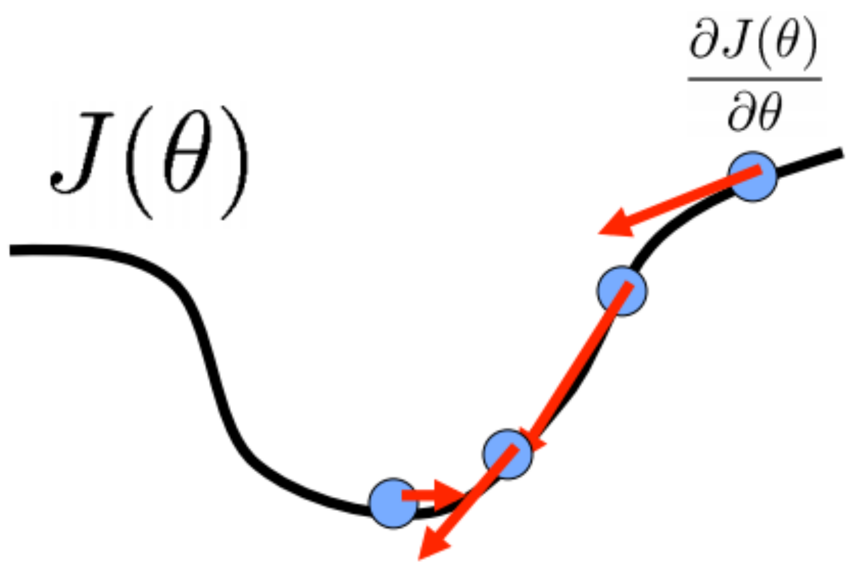
- Initialization
- Step size
  - Can change as a function of iteration
- Gradient direction
- Stopping condition

Initialize θ

Do {

$\theta \leftarrow \theta - \alpha \nabla_{\theta} J(\theta)$

} while ($\alpha \|\nabla J\| > \epsilon$)



Gradient descent for linear regression

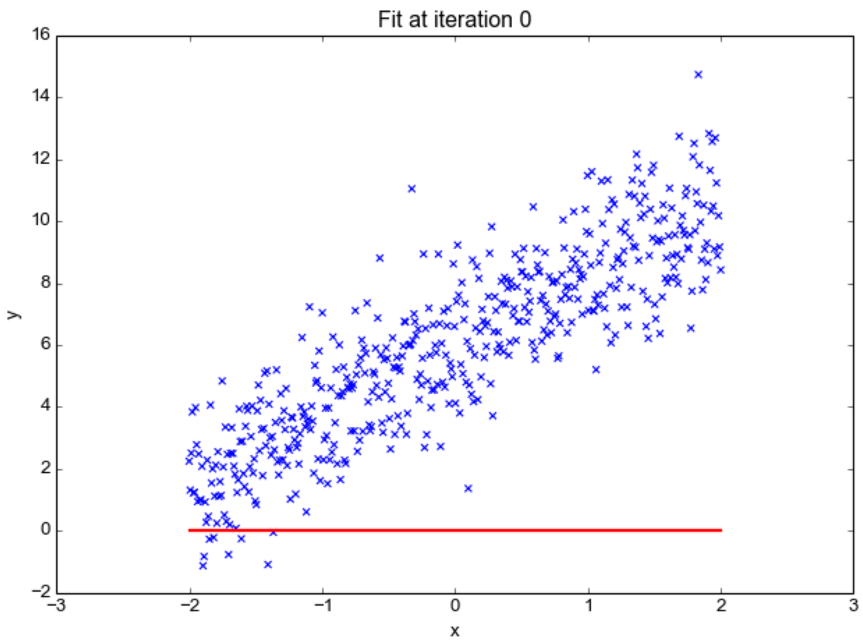
Repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x^{(i)}$$

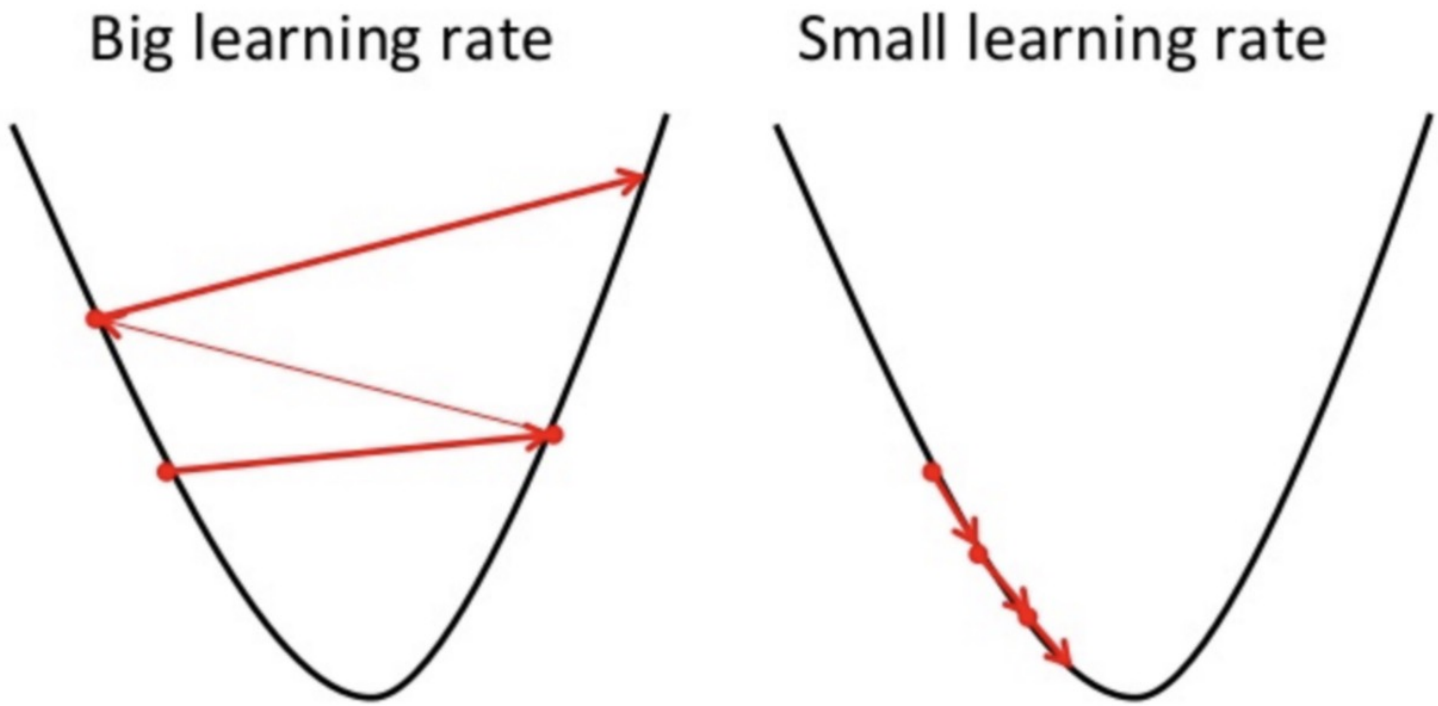
}

Update θ_0 *and* θ_1 *simultaneously*



Slide credit: Andrew Ng

Learning rate



Gradient descent in practice: Learning rate

- Automatic convergence test
- α too small: slow convergence
- α too large: may not converge

- To choose α, try

0.001, … 0.01, …, 0.1, … , 1

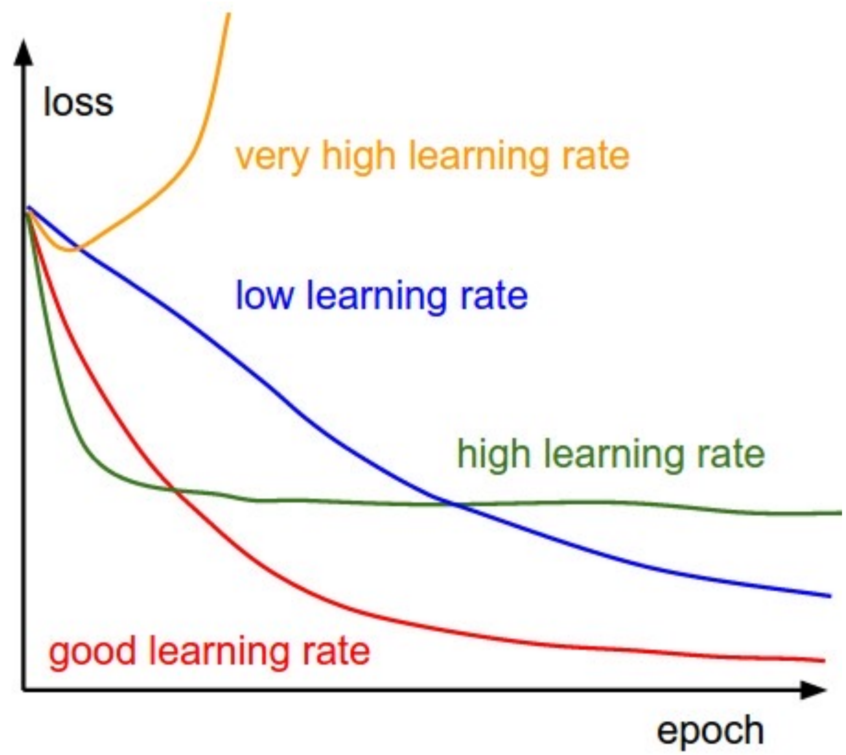
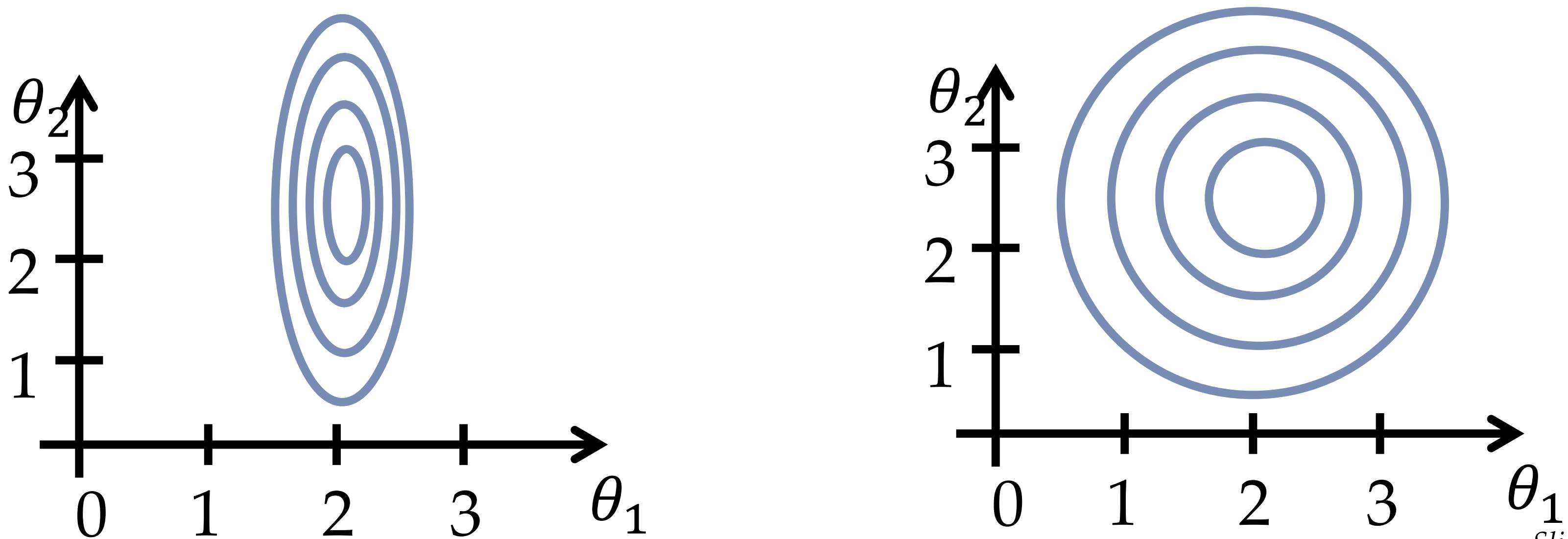


Image credit: CS231n@Stanford

Feature scaling

- Idea: Make sure features are on a similar scale (e.g,. $-1 \leq x_i \leq 1$)
- Not same scale! : x_1 = size (0-2000 sq.ft)
  x_2 = number of bedrooms (1-5)



Slide credit: Andrew Ng

Be careful

Correct: simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 := \text{temp1}$$

Slide credit: Andrew Ng

Gradient Descent for Multiple Linear Regression

$$\frac{\partial E}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial (y_i - f(\mathbf{x}_i))^2}{\partial w_j}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (y_i - f(\mathbf{x}_i)) \frac{\partial (y_i - f(\mathbf{x}_i))}{\partial w_j}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (y_i - f(\mathbf{x}_i)) \frac{\partial (y_i - \sum_{k=1}^{d} w_k x_{ik})}{\partial w_j}$$

$$= \frac{1}{N}\sum_{i=1}^{N} (y_i - f(\mathbf{x}_i)) \times (-x_{ij})$$

Gradient Descent for Multiple Linear Regression

- Gradient descent update:

$$w_j = w_j - \eta \frac{\partial E}{\partial w_j}$$

$$E = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$

$$w_j = w_j - \eta \frac{2}{N}\sum_{i=1}^{N}(y_i - f(x_i)) \cdot (-x_{ij})$$

where x_i is the ith input pattern and x_{ij} is its jth dimension

Self Study

Example to Show

- Let $E(x)=(x-2)^2$ (corresponding to the plot in the previous slide with w*=2)

- If we compute the gradient (one dimensional as there is only one variable),
- $dE/dx = 2x - 4$

- If we evaluate the gradient at a given point, say x=3, we find that the slope is positive and magnitude is 2. We indicate this as $dE/dx\,|_{x=3} = +2$ etc.

- If we evaluate the gradient at a given point, say x=1, we find that the slope is negative and magnitude is 2.

- Apply 2-steps of gradient descent to:

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

Example

$$F(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_1$$

$$\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \qquad \alpha = 0.1$$

$$\nabla F(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} F(\mathbf{x}) \\ \frac{\partial}{\partial x_2} F(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 + 1 \\ 2x_1 + 4x_2 \end{bmatrix} \qquad \mathbf{g}_0 = \nabla F(\mathbf{x})\Big|_{\mathbf{x} = \mathbf{x}_0} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \mathbf{g}_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - 0.1 \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{x}_2 = \mathbf{x}_1 - \alpha \mathbf{g}_1 = \begin{bmatrix} 0.2 \\ 0.2 \end{bmatrix} - 0.1 \begin{bmatrix} 1.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.02 \\ 0.08 \end{bmatrix}$$