Revision

COST FUNCTION, ERROR, LOSS

Cost Function

$$\boldsymbol{Squared\ Error = (h(x) - y)^2}$$

$$\boldsymbol{Sum\ of\ Squared\ Errors = \sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2}$$

Note: superscript is not power, but representing ith instance/record.

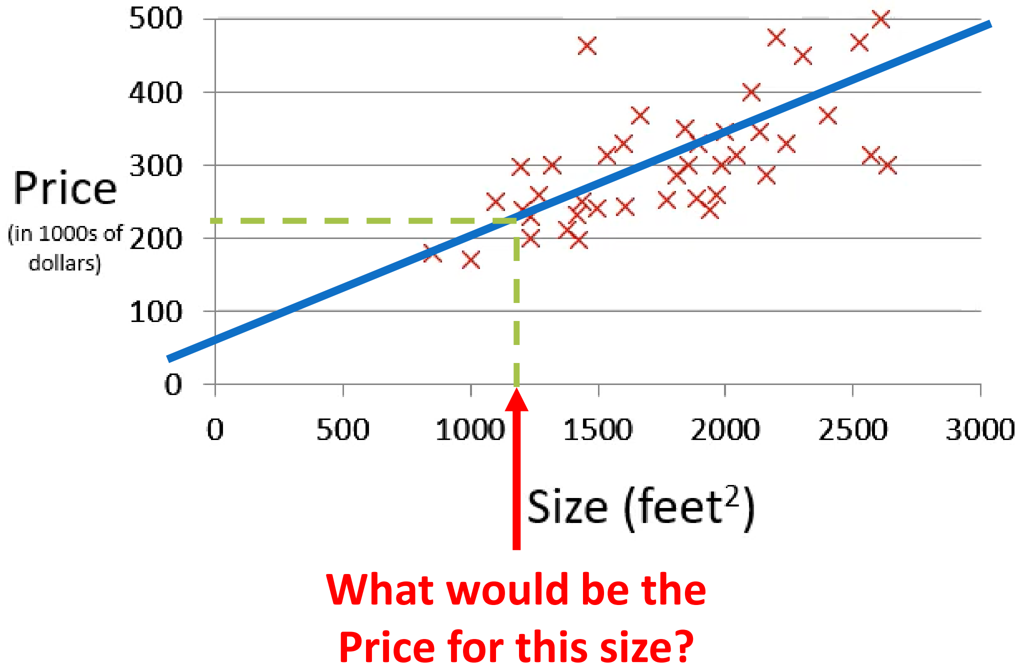
$$\boldsymbol{MSE = \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2}$$

Or equally…

$$\boldsymbol{MSE = \frac{\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2}{m}}$$

LR with One Variable: Alternative Perspective

Size $(Feet^2)$	Price $\$(\times 1000)$
1500	190
2250	285
2740	420
2318	300
2500	350
1250	180
…	…



Notations:

$$m = Total\ Number\ of\ Training\ Samples$$
$$x = Feature$$
$$y = Label$$

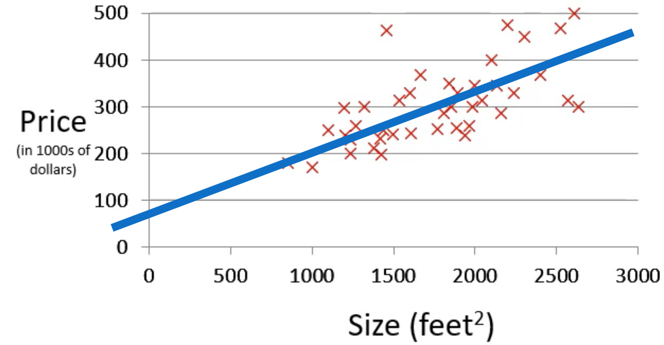
$(x^{(i)}, y^{(i)})$: the ith sample in the dataset
i.e., when $i = 1, x^{(1)} = 2250, y^{(1)} = 285$

Univariate Linear Regression

- Also called, Linear Regression with one variable or simple linear regression

$$y = mx + b$$

$$h(x) = w_0 + w_1 x$$



Parameters:

$$w_0, w_1$$

Goal: Choose w_0, w_1 such that $h(x) \approx y$ for training examples (x, y)

Or…

Goal: Choose w_0, w_1 such that $h(x) - y \approx 0$

Cost Function:

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2$$

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right)^2$$

Will be helpful later when differentiating. This only changes scale of minimization.

A Simplified Case

Restrict to those lines which pass through origin.

Assume $w_0 = 0$

Hypothesis

$$h(x) = w_0 + w_1 x$$

Parameters

$$w_0, w_1$$

Cost Function

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Goal

$$Minimize_{w_0, w_1}\, J(w_0, w_1)$$

Hypothesis

$$h(x) = w_1 x$$

Parameters

$$w_1$$

Cost Function

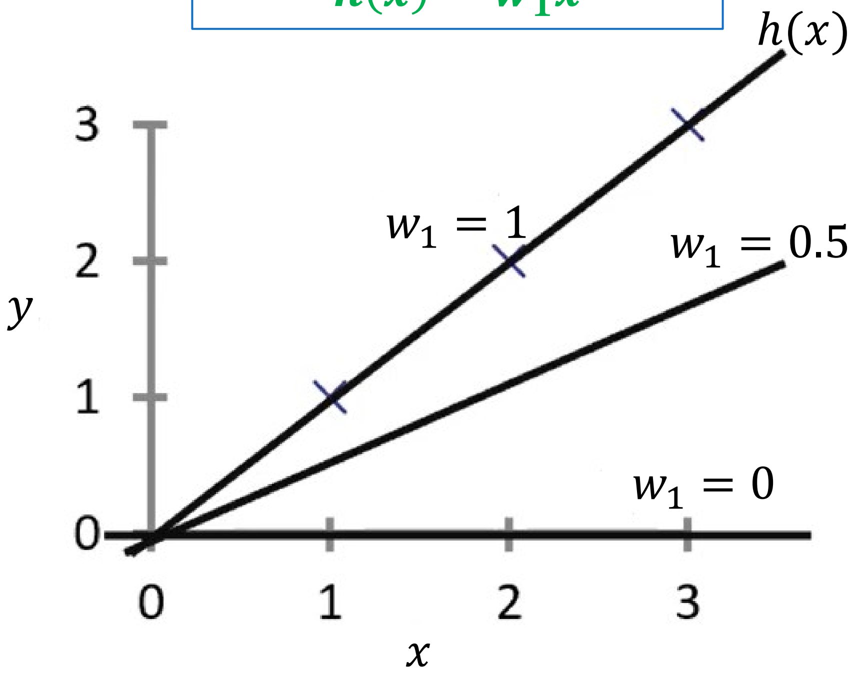
$$J(w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Goal

$$Minimize_{w_1}\, J(w_1)$$

Visualizing Hypothesis and Cost Function

$$h(x) = w_1 x$$



$$J(w_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

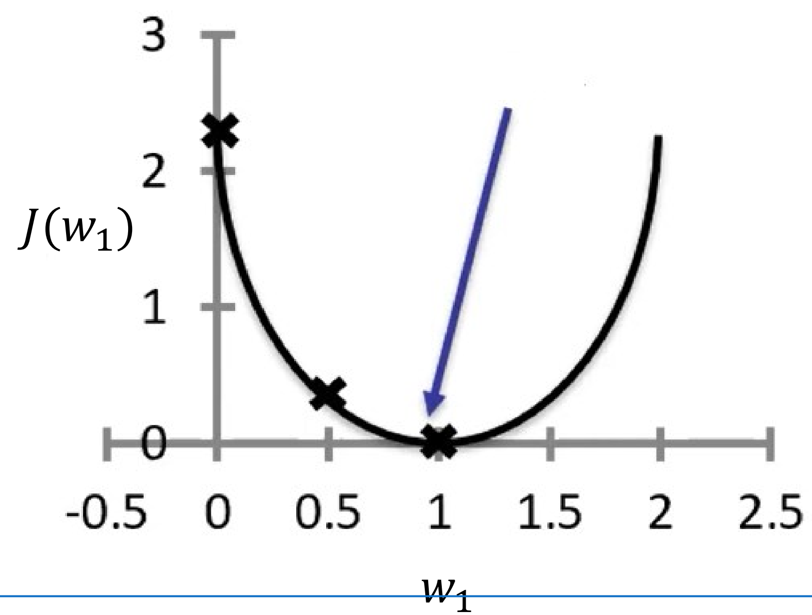
$$J(w_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(w_1 x^{(i)} - y^{(i)}\right)^2$$

$$J(w_1)$$

$$J(1) = \frac{1}{2(3)}\left((1-1)^2 + (2-2)^2 + (3-3)^2\right) = 0$$

$$J(0.5) = \frac{1}{6}\left((0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right) = 0.58$$

$$J(0) = \frac{1}{6}\left((0-1)^2 + (0-2)^2 + (0-3)^2\right) = 2.3$$



In actual, the cost functions are not that nice!

Using Both Parameters

- **Hypothesis**

$$\boldsymbol{h(x) = w_0 + w_1 x}$$

- **Parameters**

$$\boldsymbol{w_0, w_1}$$

- **Cost Function**

$$\boldsymbol{J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2}$$

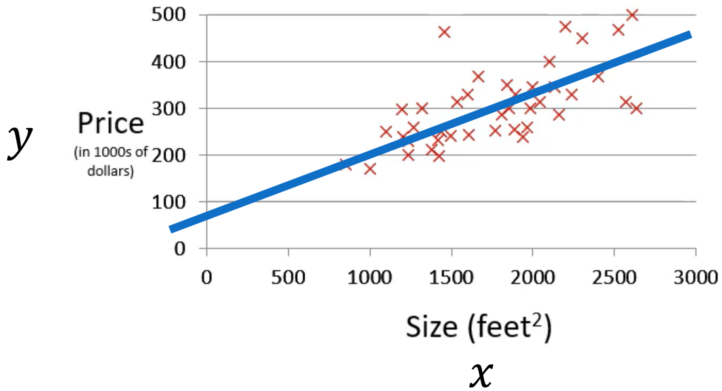
- **Goal**

$$\boldsymbol{Minimize_{w_0, w_1}\, J(w_0, w_1)}$$

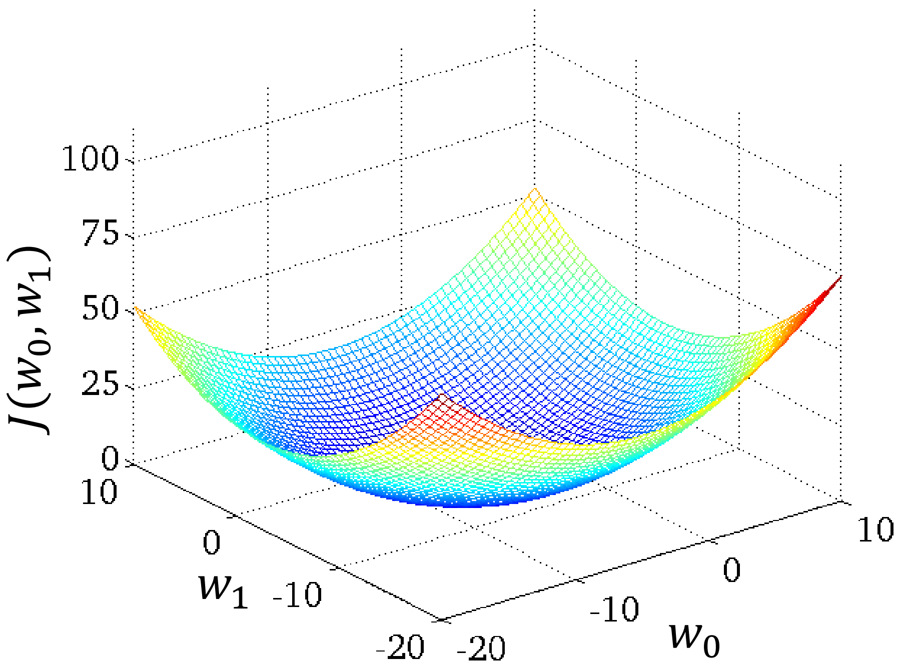
How would the "error surface" look like now?

Using Both Parameters

$$h(x) = w_0 + w_1 x$$



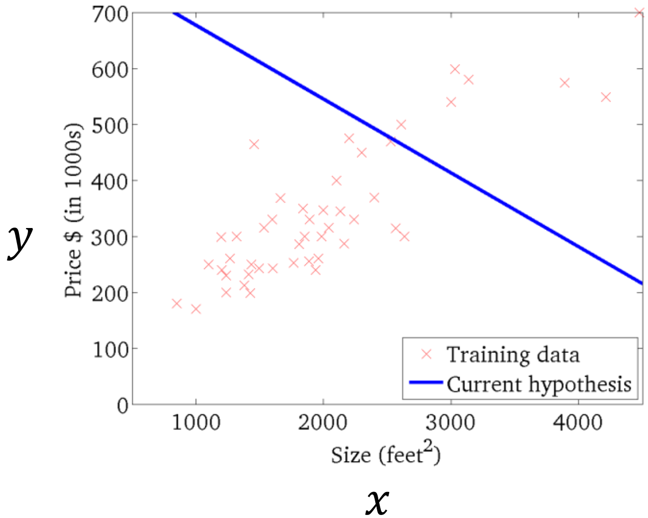
$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$



But plotting in 3D is not convenient!

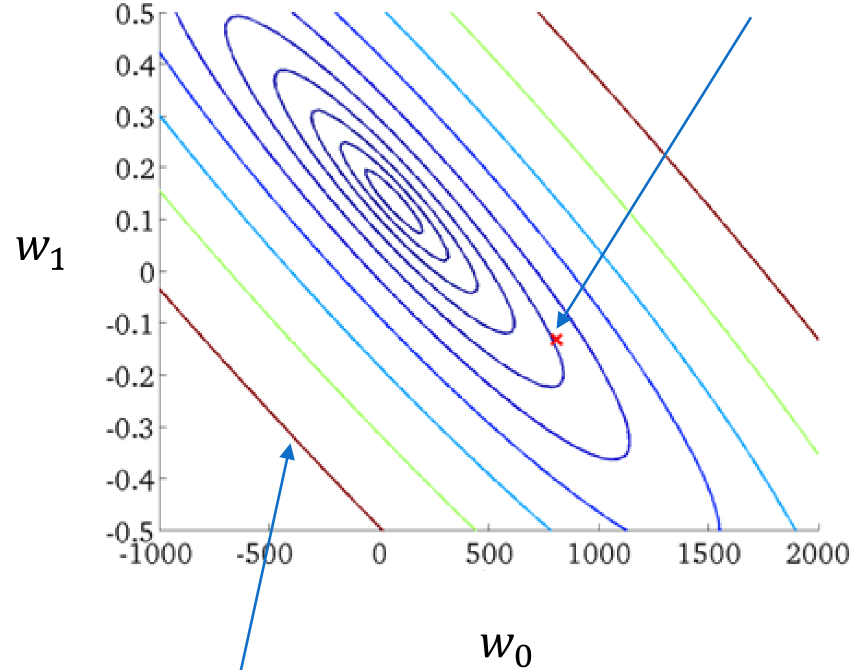
Contour Plots

$$h(x) = w_0 + w_1 x$$



$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

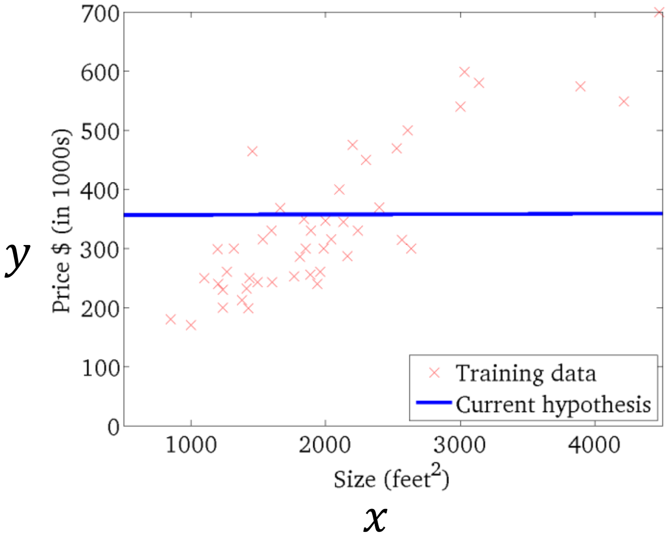
Error for current w_0, w_1



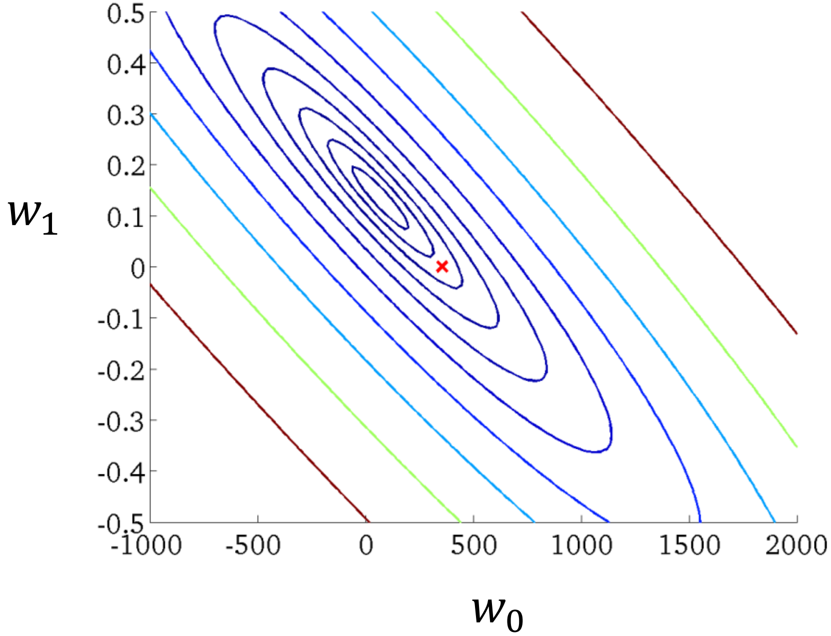
Highest error at edges. Lowest error as we move closer to the center.

Contour Plots

$$h(x) = w_0 + w_1 x$$

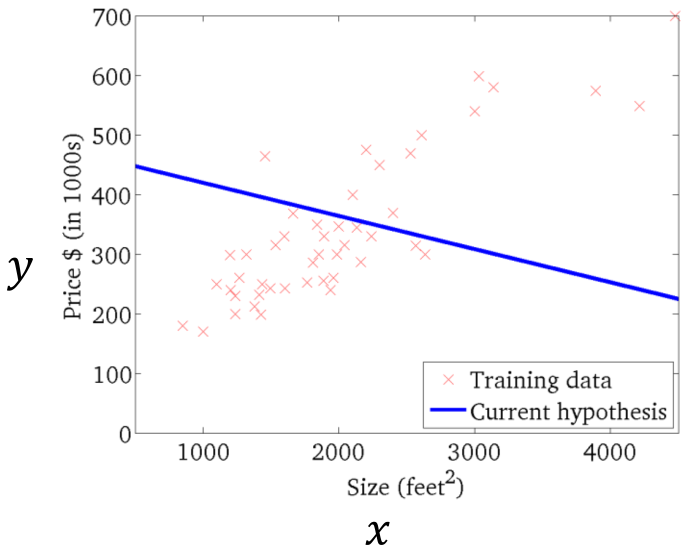


$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

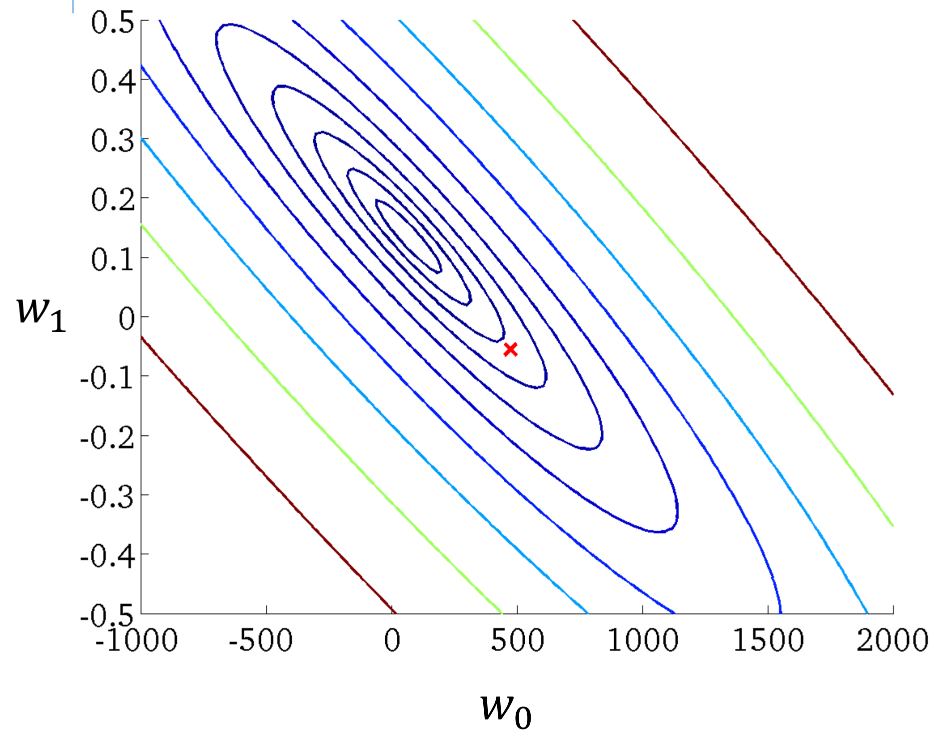


Contour Plots

$$h(x) = w_0 + w_1 x$$

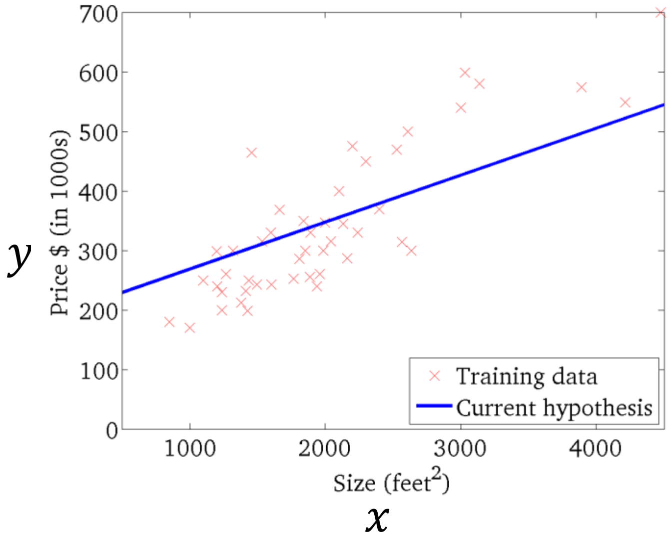


$$J(w_0, w_1) = \frac{1}{2m} \sum^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

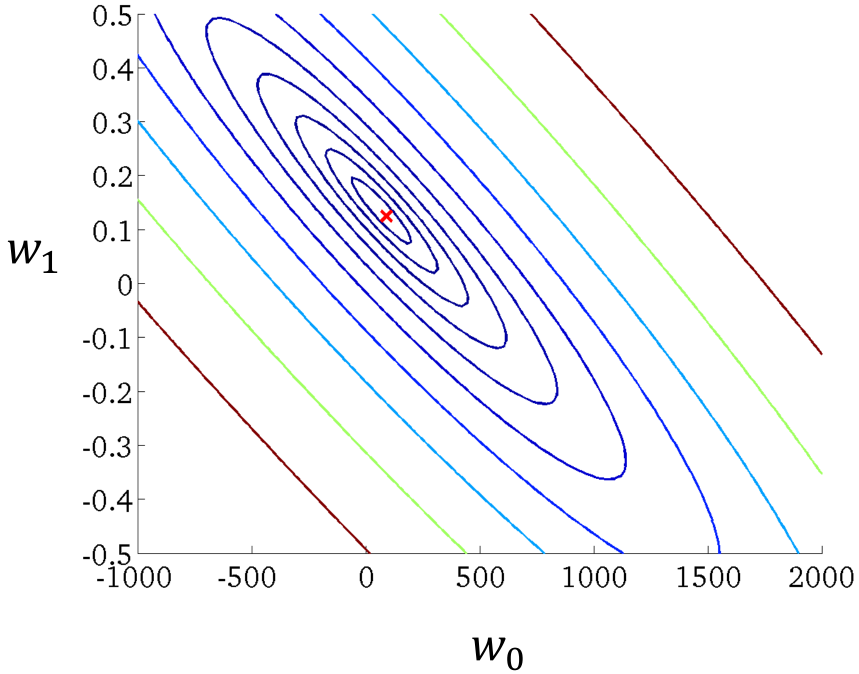


Contour Plots

$$h(x) = w_0 + w_1 x$$



$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$



How to find optimal values of w_0, w_1?

Gradient Descent Algorithm

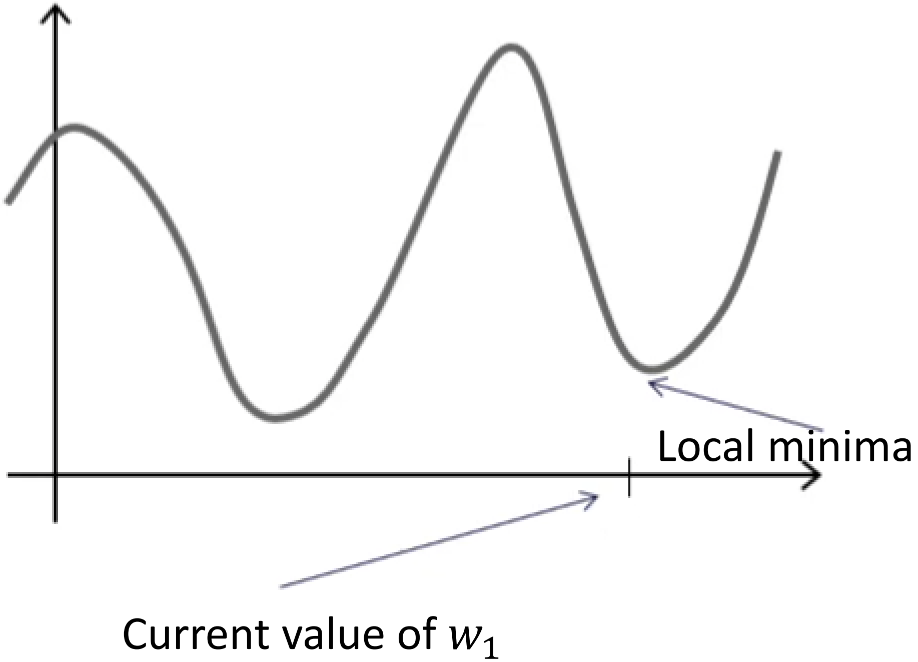
The Gradient Descent Algorithm

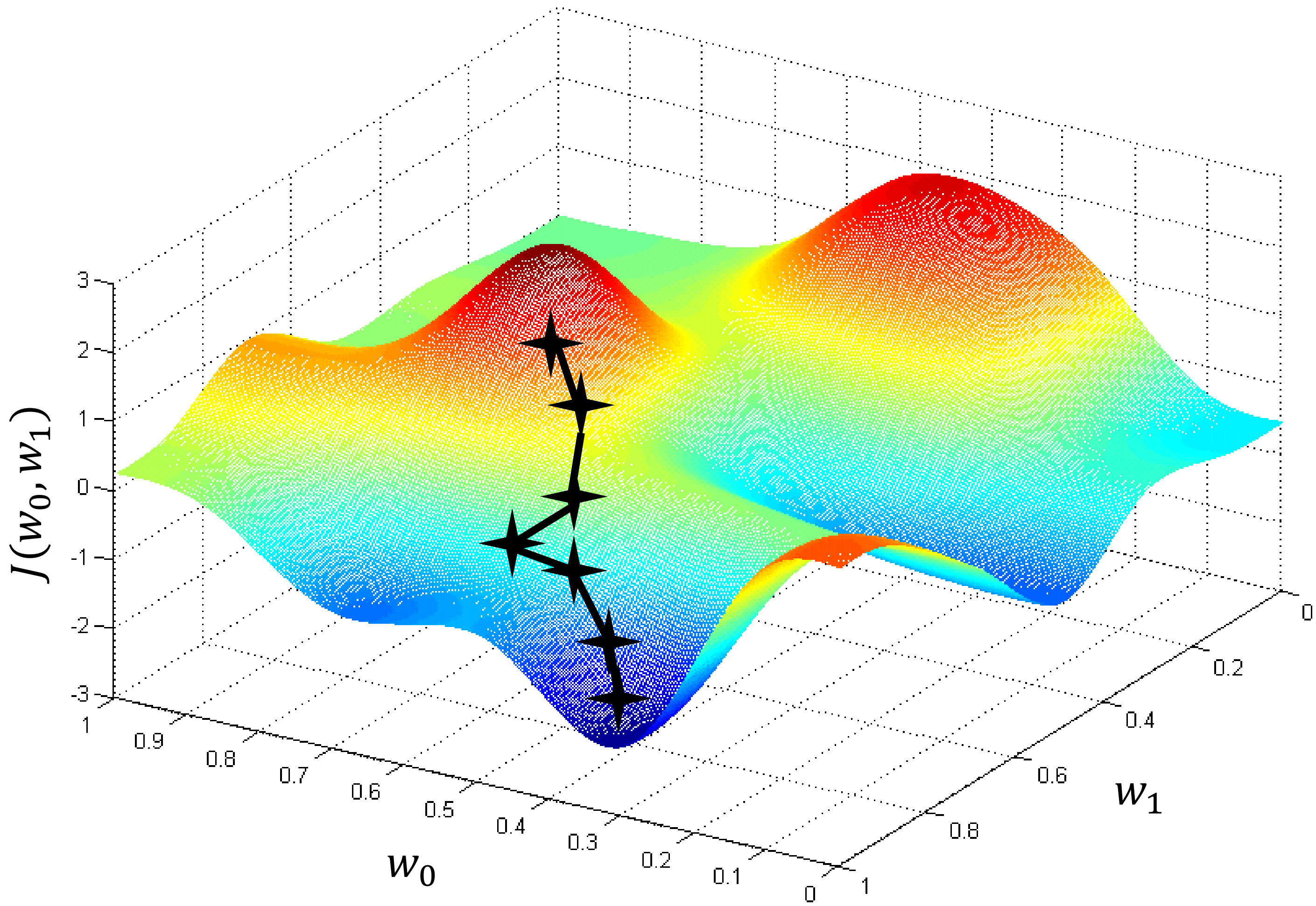
- **Goal:** Minimize $J(w_0, w_1)$
- **Outline:**
  - Start with some (w_0, w_1)
  - Keep updating (w_0, w_1) to reduce $J(w_0, w_1)$
    - Until, we **hopefully** reach **a** minimum.

Global Minimum vs Local Minimum

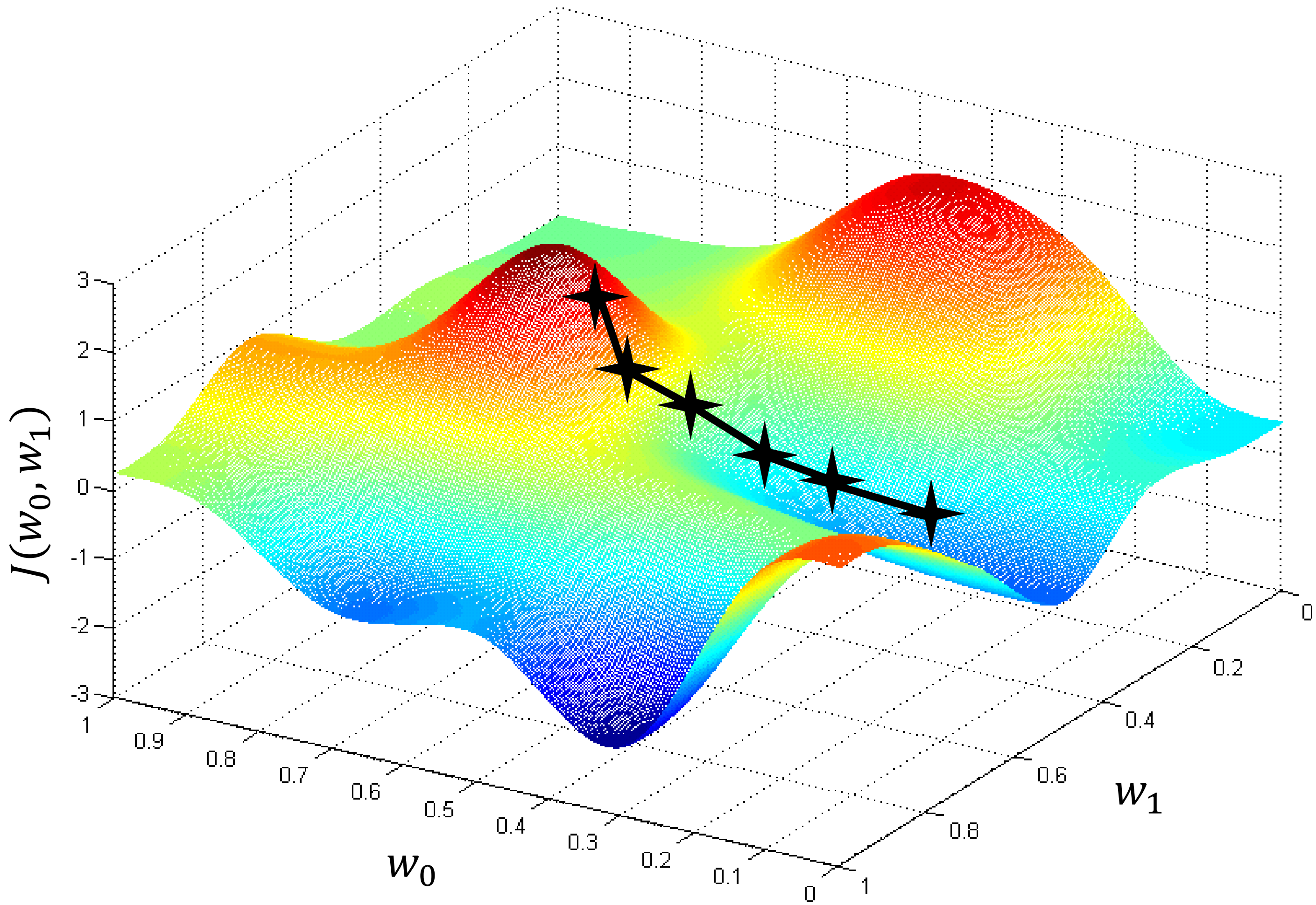
Do we really want the Global Minimum?

We must consider test data …





Initial values of our weights determine in which direction the algorithm would move!



Credit: Andrew Ng

A Simplified Version of Gradient Descent

- Assume again that we set $w_0 = 0$ and our hypothesis and cost function practically have only one coefficient, w_1

$$\boldsymbol{h(x) = w_1 x}$$

Step size (aka learning rate)

Repeat until convergence {

$$\boldsymbol{w_1 := w_1 - \alpha \boxed{\frac{\partial}{\partial w_1}\big(J(w_1)\big)}}$$

Direction to move

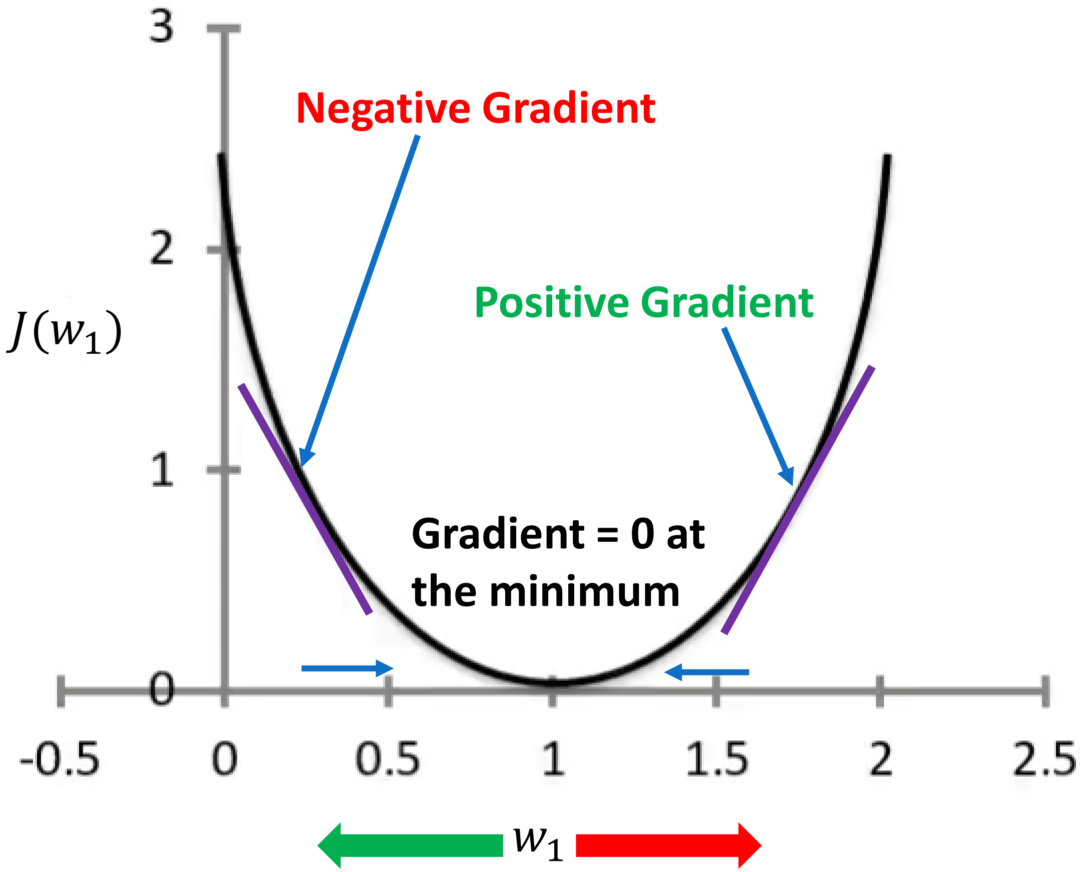
}

Where $\boldsymbol{J(w_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(w_1 x^{(i)} - y^{(i)}\right)^2}$

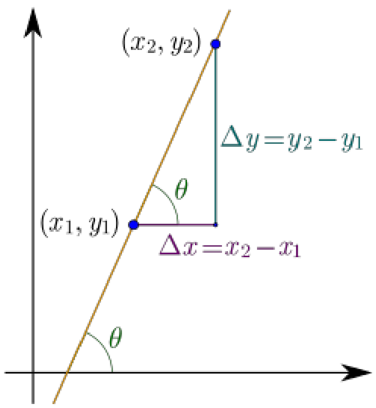
Derivate of the cost function with respect to w_1

We just want to find the rate of change from current point (instantaneous gradient)

Direction of Step



$$\boldsymbol{w_1 := w_1 - \alpha \frac{\partial}{\partial w_1}\left(J(w_1)\right)}$$



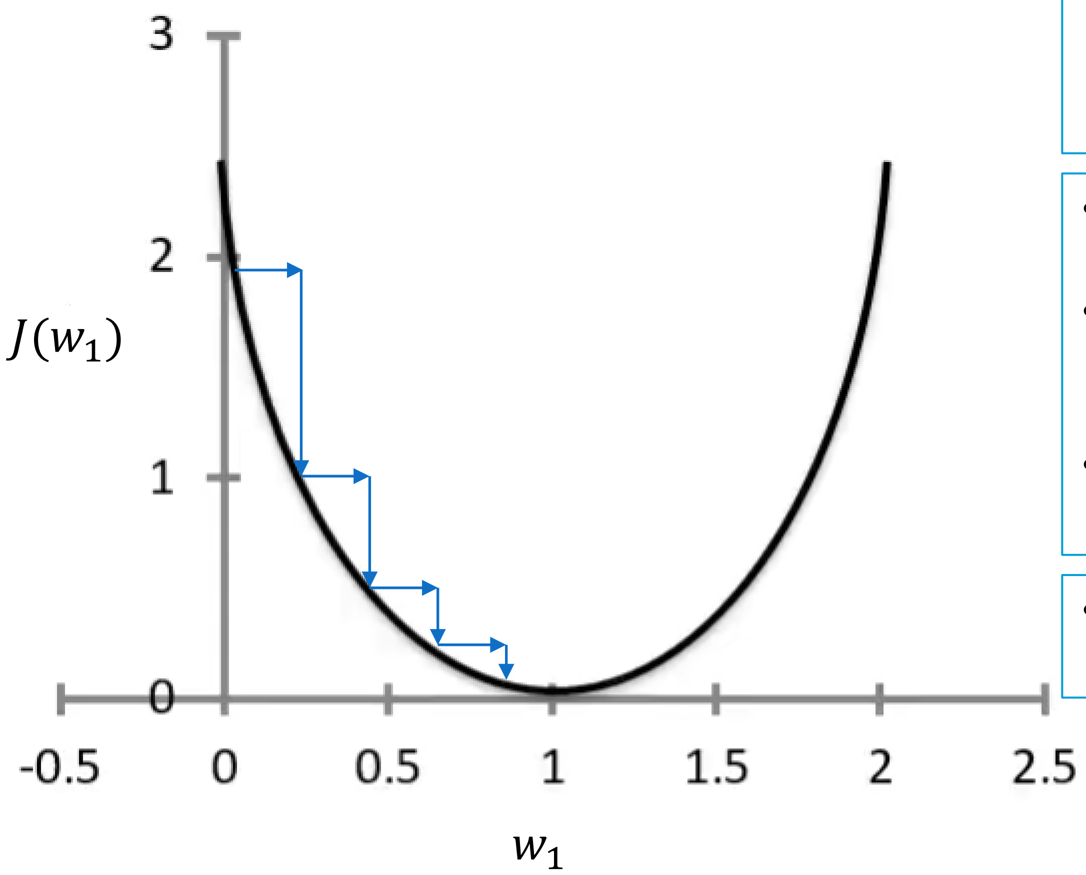
Gradient: $\boldsymbol{\frac{y_2 - y_1}{x_2 - x_1}}$

$$\boldsymbol{w_1 := w_1 - \alpha \frac{\partial}{\partial w_1}\left(J(w_1)\right)}$$

$$\boldsymbol{w_1 := w_1 - \alpha(negative): Increase\ w_1}$$

$$\boldsymbol{w_1 := w_1 - \alpha(positive): Decrease\ w_1}$$

Step Size



$$w_1 := w_1 - \alpha \frac{\partial}{\partial w_1}\left(J(w_1)\right)$$

$$w_1 := w_1 - \alpha \frac{\delta}{\delta w_1}\left(J(w_1)\right)$$

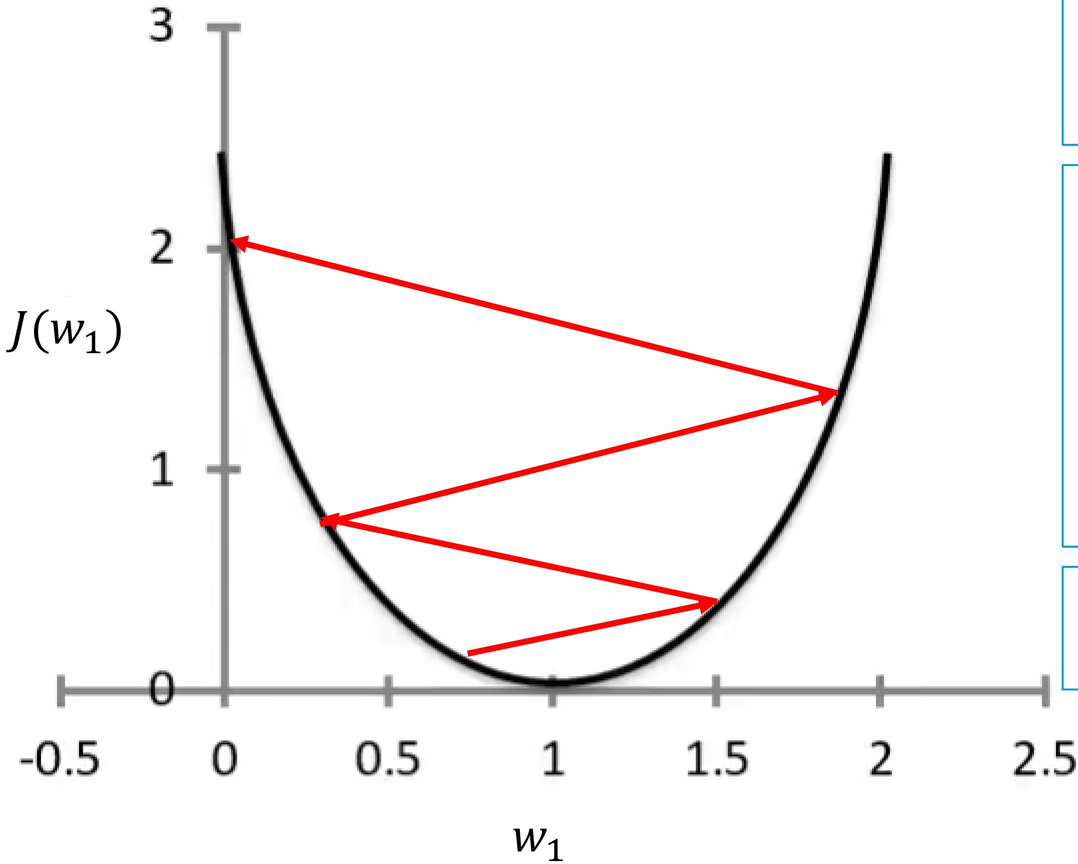
$$w_1 := w_1 - \alpha(negative): Increase\ w_1$$

$$w_1 := w_1 - \alpha(positive): Decrease\ w_1$$

- **In addition to α, the steepness of the gradient also control the step size.**
- **The step sizes become smaller as we get closer to the minimum, even with a fixed α**
- **With α too small, it takes a long time to reach the minimum.**

- **With α too small, it takes a long time to reach the minimum.**

Step Size



$$\boldsymbol{w_1 := w_1 - \alpha \frac{\partial}{\partial w_1}\left(J(w_1)\right)}$$

$$\boldsymbol{w_1 := w_1 - \alpha \frac{\delta}{\delta w_1}\left(J(w_1)\right)}$$

$$\boldsymbol{w_1 := w_1 - \alpha(negative): Increase\ w_1}$$

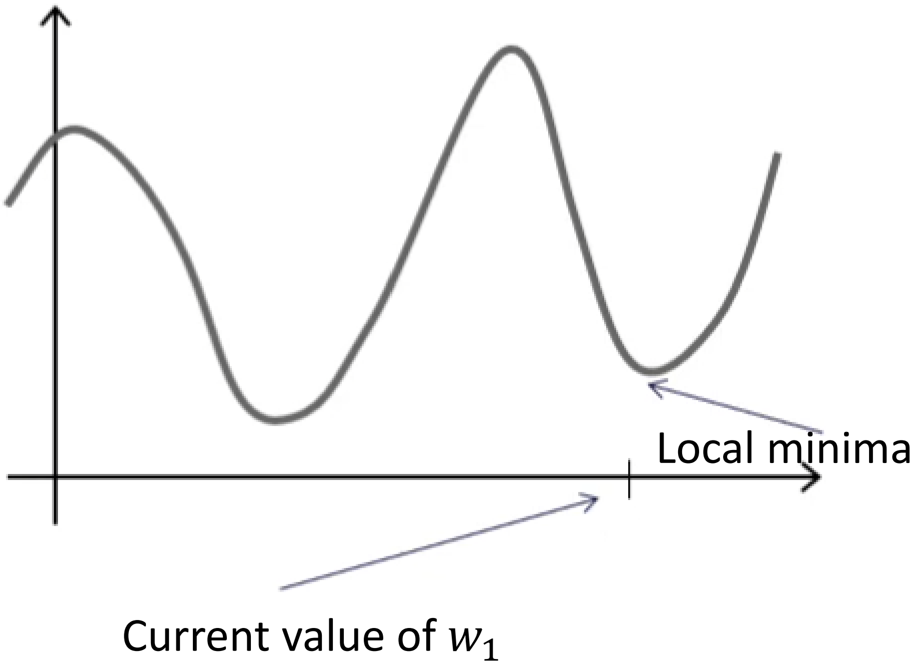
$$\boldsymbol{w_1 := w_1 - \alpha(positive): Decrease\ w_1}$$

- **In addition to α, the steepness of the gradient also control the step size.**
- **The step sizes become smaller as we get closer to the minimum, even with a fixed α**
- **With α too small, it takes a long time to reach the minimum.**

- **With α too small, it takes a long time to reach the minimum.**

- **With α too large, we can miss the minimum, and may fail to converge.**

The Problem of Local Optima



Assignment 2 – Task 2

- Use your images dataset and age label (Same as Task 1 of Assignment 2)
  - Resize images to 32x32, just like before.
- Train SGD Linear Regressor using scikit-learn using training split
- Compute R^2 and MSE on test split.
- Compare the metrics with simple linear regression model trained with OLS that you trained in Task 1. (To see which one is better i.e., OLS vs SGD regression)

Book Reading

- Murphy – Chapter 1, Chapter 14
- Tom Mitchel (TM) – Chapter 4