



Linear Regression

Qinliang Su (苏勤亮)

Sun Yat-sen University

suqliang@mail.sysu.edu.cn

Outline


- Introduction
- Single Feature Case
- Multiple Features Case
- Numerical Optimization

Introduction

- What is regression?

Based on the given features, predict the values of interested variables

- Example: House price prediction



The diagram shows a blue bracket labeled "Features" spanning the first four columns of the table. A red arrow labeled "Interest variable" points down to the fifth column.

Size (feet) x_1	# bedrooms x_2	# floors x_3	# years (Ages) x_4	Price (\$ 1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
....

Features: 1) size, 2) # bedrooms, 3) # floors, 4) # years

- Mathematically, regression aims to learn a function $f(\cdot)$ that can model the relation between input data x and output value y

$$\hat{y} = f(x_1, x_2, x_3, x_4)$$

- Linear** regression

Restricting the function family $f(\cdot)$ to be **linear-form**, *i.e.*,

$$f(x_1, x_2 \cdots x_m) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_mx_m$$

- w_k : model parameters
- m : number of features

- Objective

Find a set of parameters $\{w_k\}_{k=1}^m$ so that the prediction

$$\hat{y} = f(x_1, x_2, \dots, x_m)$$

is **as close as possible** to the ground-truth values y *for all data samples* in the training dataset

	Size (feet) x_1	# bedrooms x_2	# floors x_3	# years (Ages) x_4	Price (\$ 1000) y
Sample 1 →	2104	5	1	45	460
Sample 2 →	1416	3	2	40	232
Sample 3 →	1534	3	2	30	315
Sample 4 →	852	2	1	36	178

Outline

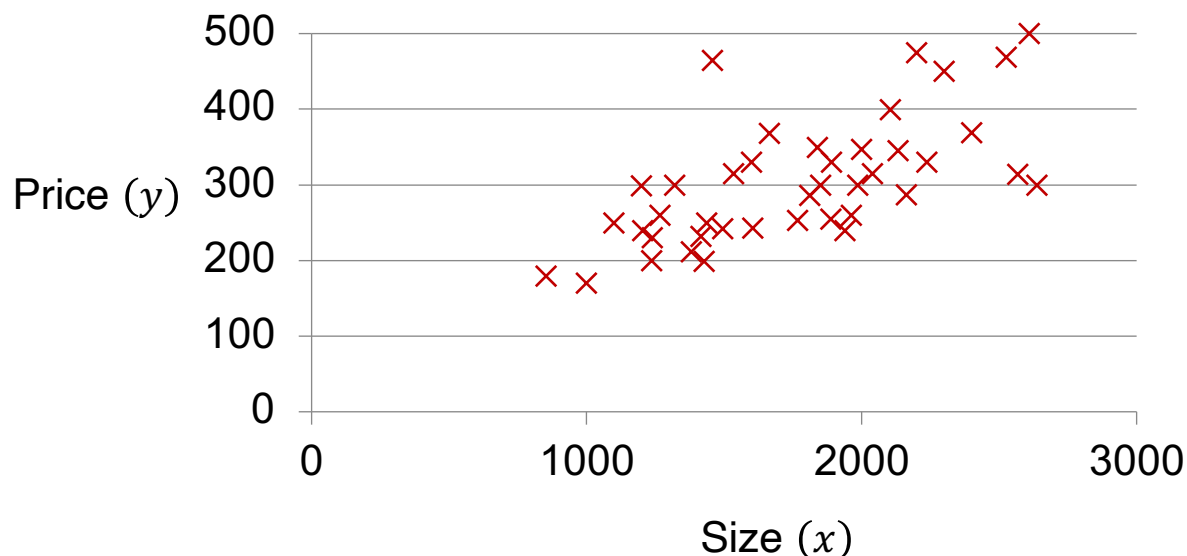
- Introduction
- **Single Feature Case**
- Multiple Features Case
- Numerical Optimization

Model

- For simplicity, first consider only one feature, e.g., *house size*

Size (feet) x	Price (\$ 1000) y
2104	460
1416	232
852	178
....

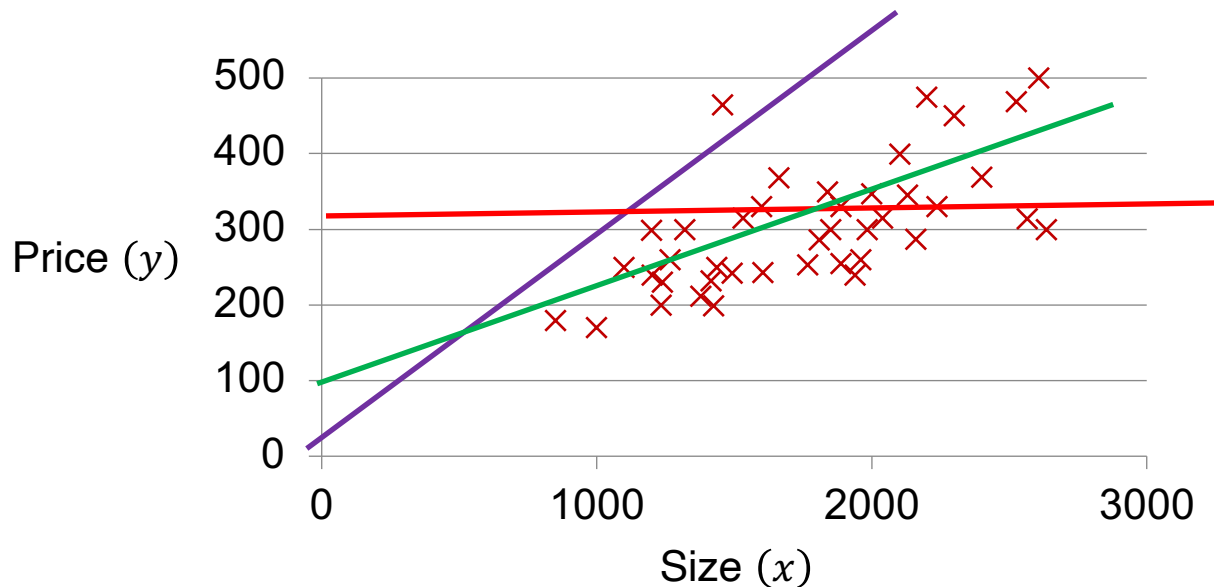
- Plot the (x, y) pairs on a plane



- The prediction function is reduced to

$$f(x) = w_0 + w_1x$$

- For different w_0 and w_1 , the function $f(x)$ represents different straight lines



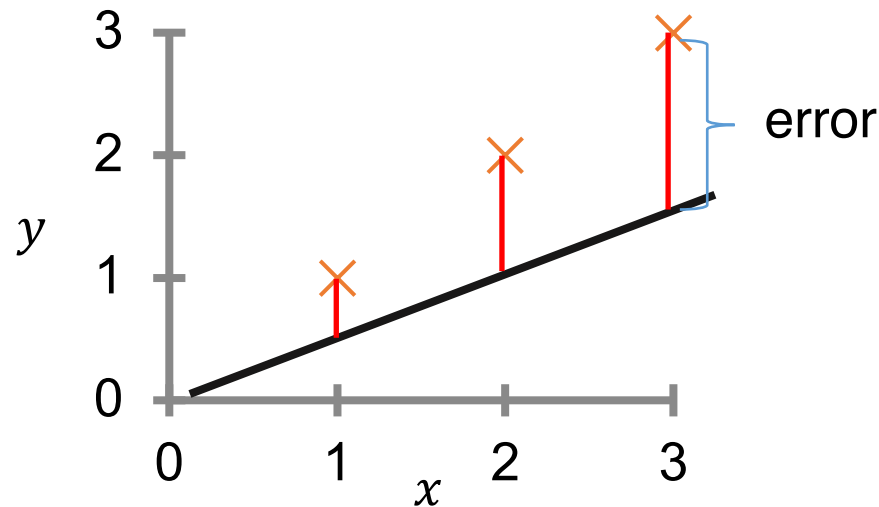
- The goal is to find an appropriate w_0 and w_1 such that **the line is as fit as possible** to the true y 's for all given x

Cost / Loss Function

- Mathematically, the goal can be formulated as minimizing the cost (loss) function

$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (f(x^{(i)}) - y^{(i)})^2$$

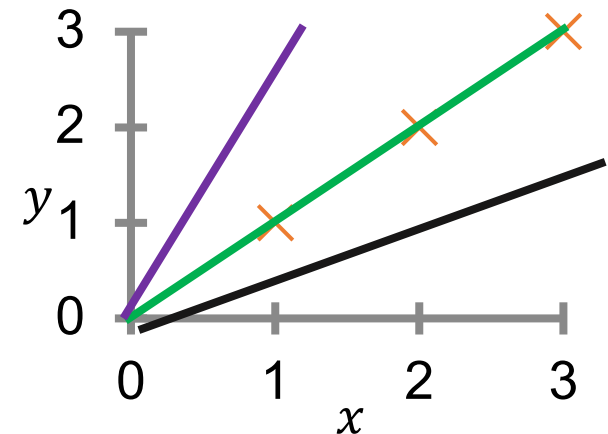
where $x^{(i)}$ and $y^{(i)}$ means the i -th feature and target values;
 n is the number of training examples



- Substituting $f(x^{(i)}) = w_0 + w_1 x^{(i)}$ into $L(w_0, w_1)$ gives

$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)})^2$$

Remark: To better understand this cost function, we simplify it by **setting** $w_0 = 0$

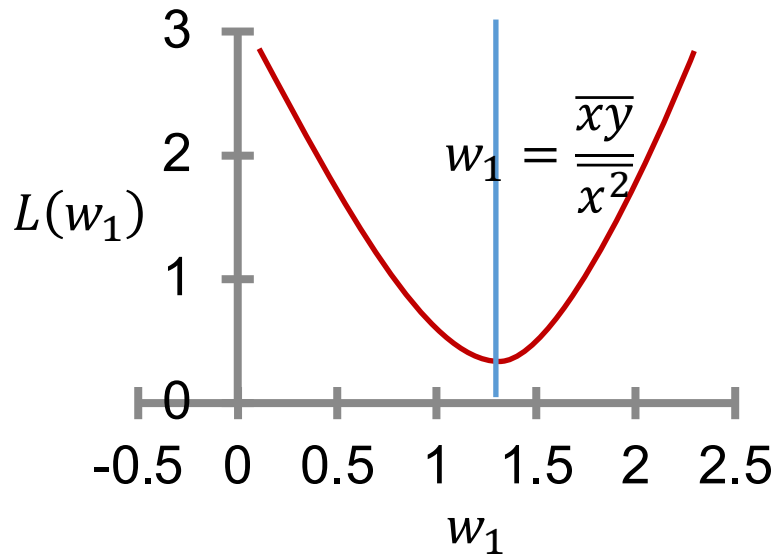


- Then, the cost function becomes

$$L(w_1) = \overline{x^2} w_1^2 - 2\overline{xy} w_1 + \overline{y^2}$$

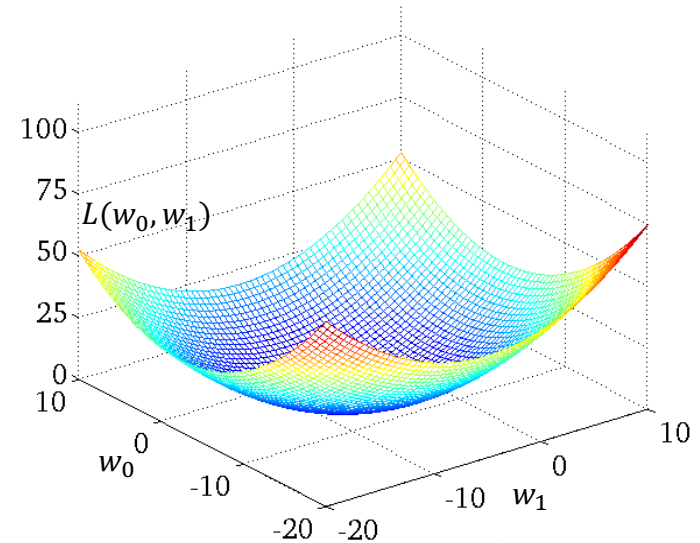
where $\overline{x^2} = \frac{\sum_{i=1}^n (x^{(i)})^2}{n}$, $\overline{xy} = \frac{\sum_{i=1}^n x^{(i)} y^{(i)}}{n}$ and $\overline{y^2} = \frac{\sum_{i=1}^n (y^{(i)})^2}{n}$

- The cost function is a **quadratic function** *w.r.t.* w_1



$$L(w_1) = \overline{x^2}w_1^2 - 2\overline{xy}w_1 + \overline{y^2}$$

- If w_0 is taken into account, the cost function $L(w_0, w_1)$ is still a quadratic function, but is **two-dimensional**



$$L(w_0, w_1) = \frac{1}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)})^2$$

- The best w_0 and w_1 can be found by setting the derivatives to zero

$$\frac{\partial L}{\partial w_0} = \frac{2}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)}) = 0$$

$$\frac{\partial L}{\partial w_1} = \frac{2}{n} \sum_{i=1}^n (w_0 + w_1 x^{(i)} - y^{(i)}) x^{(i)} = 0$$

- By solving the linear equation system, the optimal w_0 and w_1 can be derived as

$$w_0 = \frac{\overline{xy\bar{x}} - \overline{x^2}\bar{y}}{\bar{x}^2 - \overline{x^2}}$$

$$w_1 = \frac{\bar{x}\bar{y} - \overline{xy}}{\bar{x}^2 - \overline{x^2}}$$

Outline

- Introduction
- Single Feature Case
- **Multiple Features Case**
- Numerical Optimization

- Training data from single-feature to multiple-feature case

Size (feet)	Price (\$ 1000)
x	y
2104	460
1416	232
1534	315
852	178
....



Size (feet)	# bedrooms	# floors	# years (Ages)	Price (\$ 1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
....

- The function of a general linear regression is

$$f(x_1, x_2 \cdots x_m) = \mathbf{w}_0 + w_1x_1 + w_2x_2 + \cdots + w_mx_m$$

- x_i is the i -th feature
- Working with the scalar form is cumbersome. Reformulating it into a matrix-form gives

$$f(\mathbf{x}) = \mathbf{x}\mathbf{w}$$

- $\mathbf{x} = [\mathbf{1}, x_1, x_2, \cdots, x_m]$ is the feature row vector
 - $\mathbf{w} = [\mathbf{w}_0, w_1, w_2, \cdots, w_m]^T$ is the parameter column vector

By setting the first element in \mathbf{x} to be 1, \mathbf{w}_0 can be treated in the same way as the other parameters w_k

Cost Function

- The goal is still to find a \mathbf{w} such that the prediction

$$f(\mathbf{x}^{(i)}) = \mathbf{x}^{(i)} \mathbf{w}$$

is close to the true value $y^{(i)}$, where $\mathbf{x}^{(i)}$ and $y^{(i)}$ is the i -th feature vector and target value

Size (feet)	# bedrooms	# floors	# years (Ages)	Price (\$ 1000)
x_1	x_2	x_3	x_4	y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
....

- Thus, the cost function can be represented as

$$L(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} \mathbf{w} - y^{(i)})^2$$

- The cost function can be further written as

$$L(\mathbf{w}) = \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

where \mathbf{X} is the feature matrix defined as

$$\mathbf{X} \triangleq \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(n)} \end{bmatrix}$$

	Size (feet)	# bedrooms	# floors	# years (Ages)	Price (\$ 1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1

\mathbf{X} \mathbf{y}

- The gradient of the cost function w.r.t. \mathbf{w} is

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{2}{n} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y})$$

- Since $L(\mathbf{w})$ is a convex function, its optima can be found by setting

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{2}{n} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y}) = 0$$

- Solving the equation gives

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

- It can be verified that when the number of feature is 1, the result reduces to

$$w_0 = \frac{\overline{xy\bar{x}} - \overline{x^2}\bar{y}}{\bar{x}^2 - \overline{x^2}}, \quad w_1 = \frac{\bar{x}\bar{y} - \overline{xy}}{\bar{x}^2 - \overline{x^2}}$$

Supplement: gradient of a function w.r.t. a vector or matrix

- The meaning of gradient w.r.t a vector or matrix
 - $L(\cdot)$ is a scalar function

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \triangleq \begin{bmatrix} \frac{\partial L(\mathbf{w})}{\partial w_1} \\ \vdots \\ \frac{\partial L(\mathbf{w})}{\partial w_m} \end{bmatrix} \quad \frac{\partial L(\mathbf{X})}{\partial \mathbf{X}} \triangleq \begin{bmatrix} \frac{\partial L(\mathbf{X})}{\partial x_{11}} & \cdots & \frac{\partial L(\mathbf{X})}{\partial x_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L(\mathbf{X})}{\partial x_{m1}} & \cdots & \frac{\partial L(\mathbf{X})}{\partial x_{mm}} \end{bmatrix}$$

The gradient *w.r.t.* a vector or matrix is **just the compact notation** of gradients of function *w.r.t.* every element

- $L(\cdot)$ could be any function, *e.g.*, norm $\|\cdot\|^2$, sum $\sum_{i=1}^m w_i$, matrix trace $trace(\cdot)$, matrix determinant $\det(\cdot)$ etc.

$$\frac{\partial \|\mathbf{w}\|^2}{\partial \mathbf{w}} = 2\mathbf{w} \quad \frac{\partial \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2}{\partial (\mathbf{X}\mathbf{w} - \mathbf{y})} = 2(\mathbf{X}\mathbf{w} - \mathbf{y})$$

- When $\mathbf{L}(\cdot) = [L_1(\mathbf{w}), \dots, L_p(\mathbf{w})]$ is a **row vector function**

$$\frac{\partial \mathbf{L}(\mathbf{w})}{\partial \mathbf{w}} \triangleq \begin{bmatrix} \frac{\partial L_1(\mathbf{w})}{\partial \mathbf{w}} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial \mathbf{w}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L_1(\mathbf{w})}{\partial w_1} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial w_1} \\ \vdots & \dots & \vdots \\ \frac{\partial L_1(\mathbf{w})}{\partial w_m} & \dots & \frac{\partial L_p(\mathbf{w})}{\partial w_m} \end{bmatrix}$$

It is still a compact notation

- Then, we can see that

$$\frac{\partial \|X\mathbf{w} - \mathbf{y}\|^2}{\partial \mathbf{w}} = \frac{\partial (X\mathbf{w} - \mathbf{y})^T}{\partial \mathbf{w}} \frac{\partial \|X\mathbf{w} - \mathbf{y}\|^2}{\partial (X\mathbf{w} - \mathbf{y})} = 2X^T(X\mathbf{w} - \mathbf{y})$$

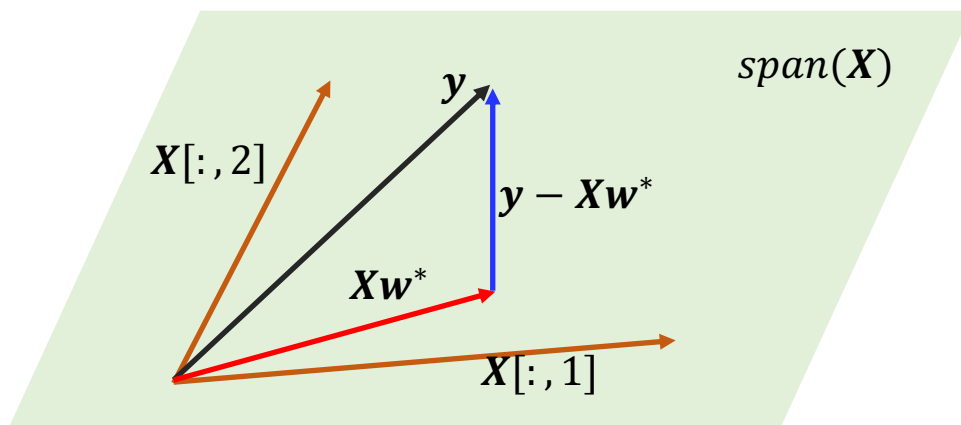
chain rule

Geometric Interpretation

- From the requirement of $\mathbf{X}^T(\mathbf{X}\mathbf{w}^* - \mathbf{y}) = \mathbf{0}$, we can see that

$$\mathbf{y} - \mathbf{X}\mathbf{w}^* \perp \text{span}(\mathbf{X})$$

- The result suggests that $\mathbf{X}\mathbf{w}^*$ can be understood as *the projection of \mathbf{y} onto the space spanned by \mathbf{X}*



Outline

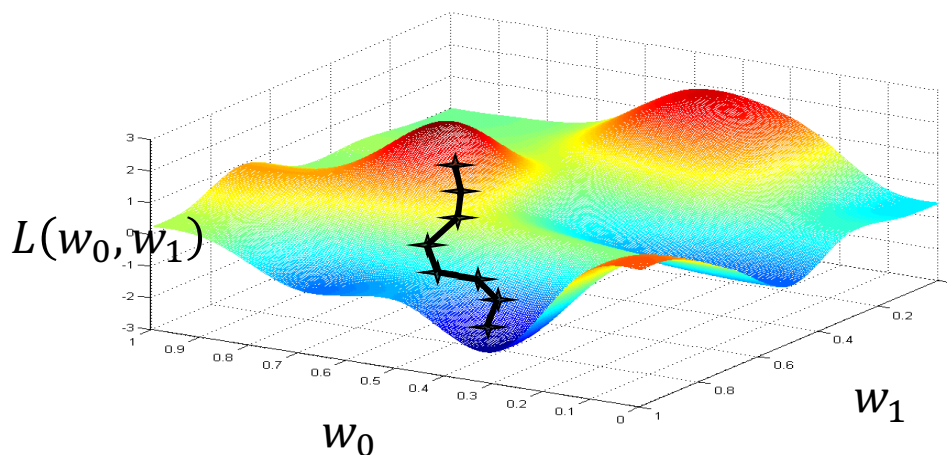
- Introduction
- Single Feature Case
- Multiple Features Case
- Numerical Optimization

Gradient Descent

- Close-form solutions do *not always exist*, or computing the close-form expression is *too expensive*
- Under such circumstances, we can resort to numerical methods, *e.g.*, the gradient descent

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - r \cdot \left. \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}=\mathbf{w}^{(t)}}$$

- r : the learning rate

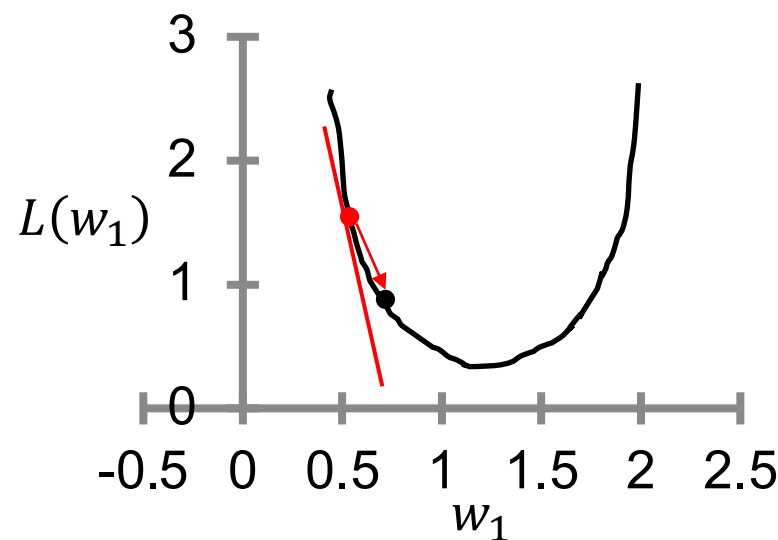
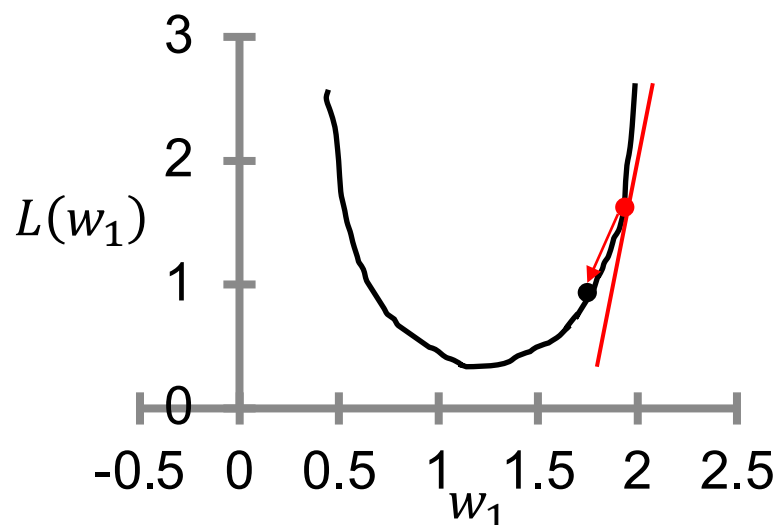


- Let us take **the single-feature case** and set $w_0 = 0$ as an example, under which the loss function becomes

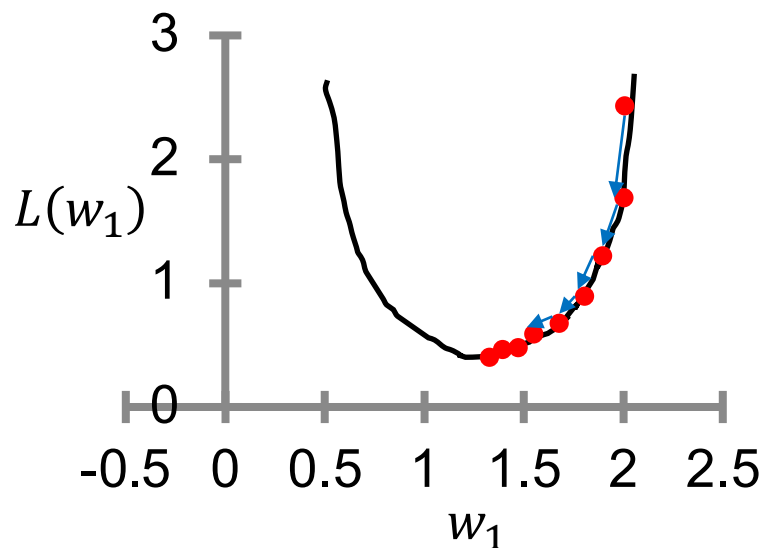
$$L(w_1) = \frac{1}{n} \sum_{i=1}^n (w_1 x^{(i)} - y^{(i)})^2$$

- The parameter w_1 can be updated as

$$w_1^{(t+1)} = w_1^{(t)} - r \cdot \left. \frac{\partial L(w_1)}{\partial w_1} \right|_{w_1 = w_1^{(t)}}$$

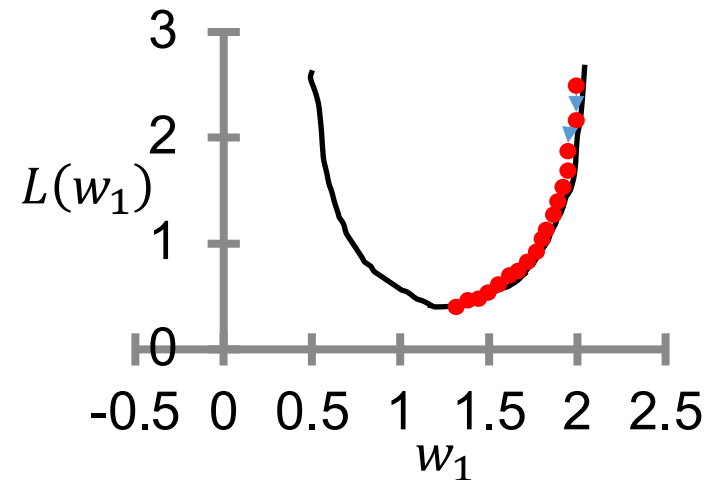


- With **appropriate learning rate**, the model parameter is iteratively updated, and will eventually converge to the optima

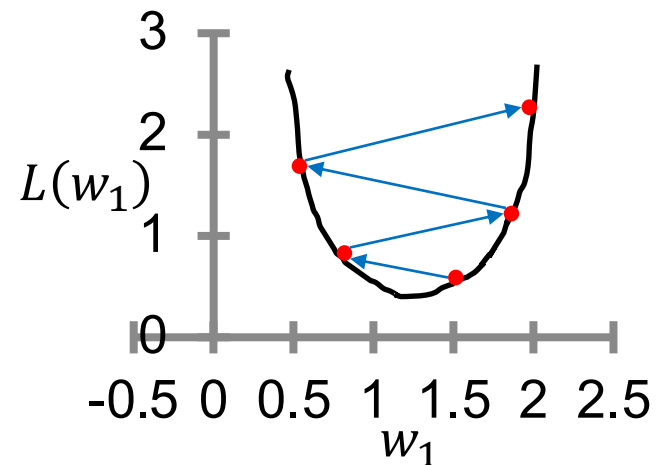


- As approaching the optimal value, the gradient becomes smaller and smaller. Thus, *even if the learning rate is fixed*, the updating intervals also **approach 0** as the iteration proceeds, **as long as the rate is set appropriately**

- If the learning rate is **too small**, the convergence speed will be **very slow**



- If the learning rate is **too large**, the iteration may **diverge**

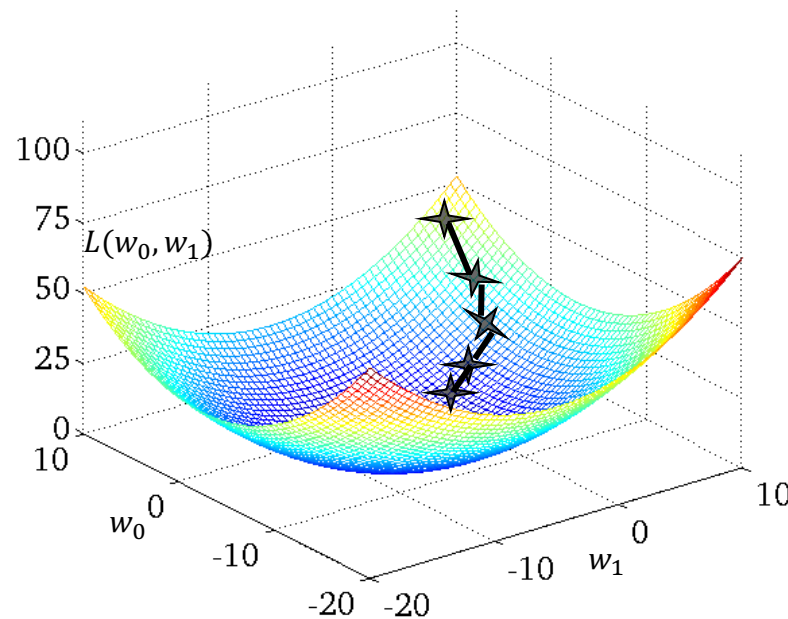


- So, setting appropriate learning rate is important

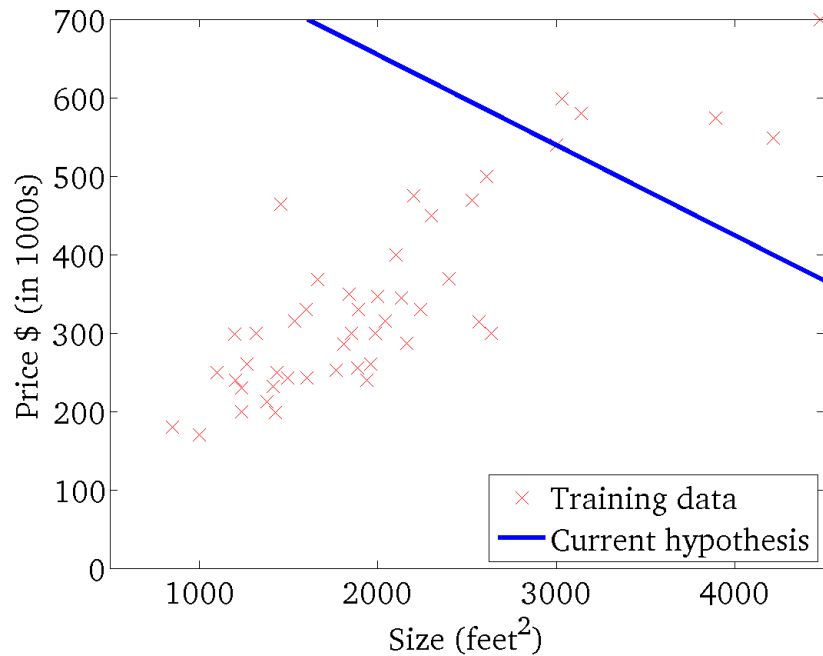
- Now consider the case with both w_0 and w_1

$$w_0^{(t+1)} = w_0^{(t)} - r \cdot \left. \frac{\partial L(w_0, w_1)}{\partial w_0} \right|_{w_0=w_0^{(t)}, w_1=w_1^{(t)}}$$

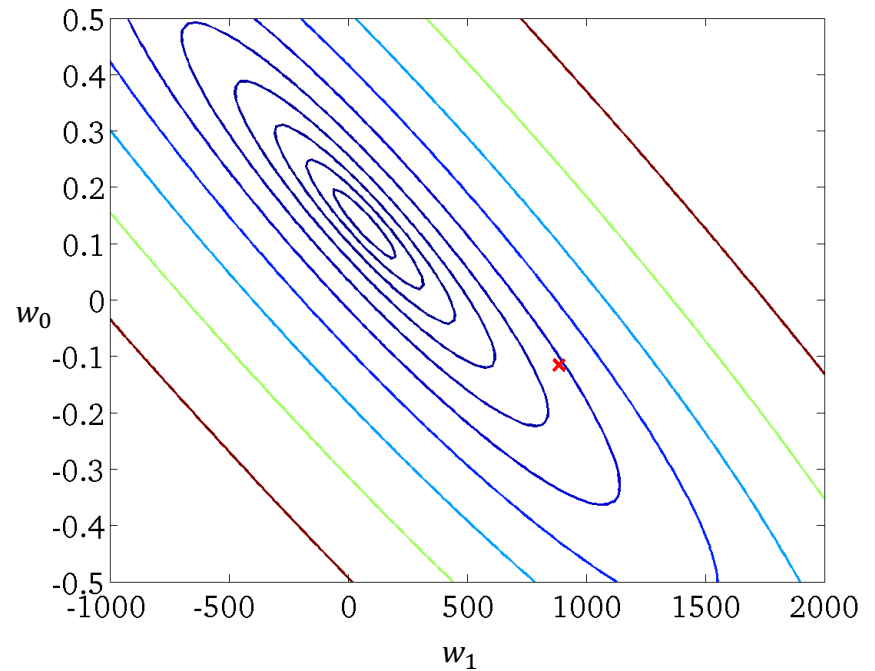
$$w_1^{(t+1)} = w_1^{(t)} - r \cdot \left. \frac{\partial L(w_0, w_1)}{\partial w_1} \right|_{w_0=w_0^{(t)}, w_1=w_1^{(t)}}$$



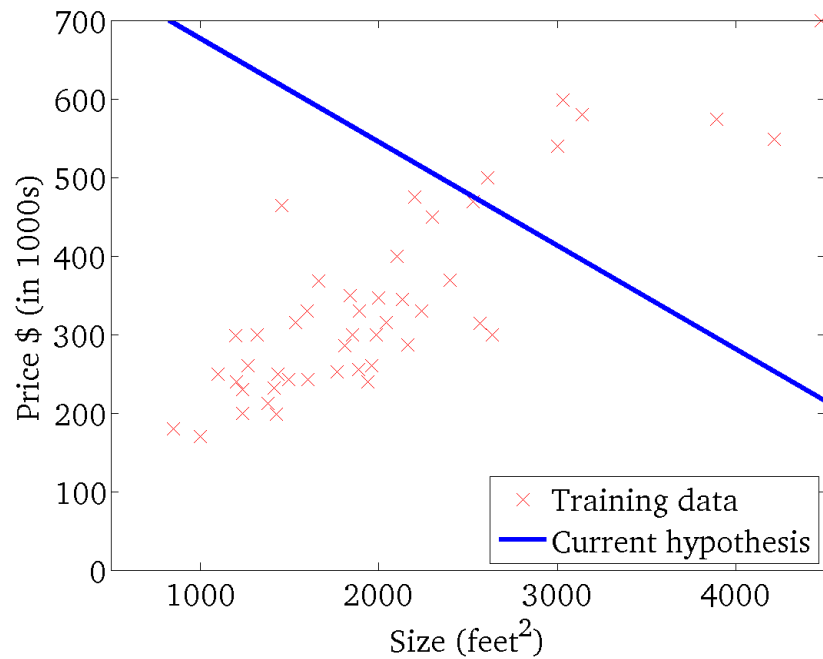
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



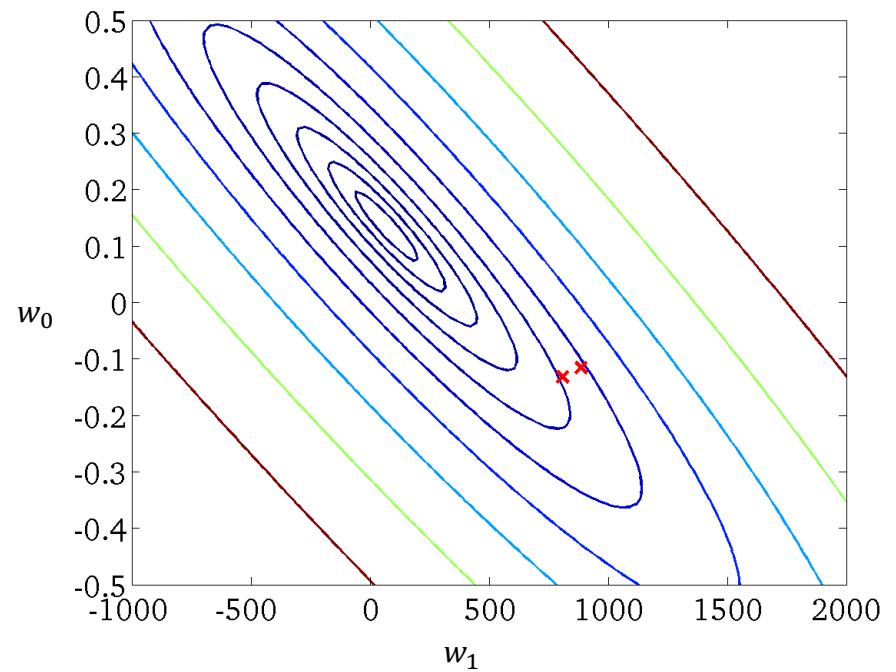
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



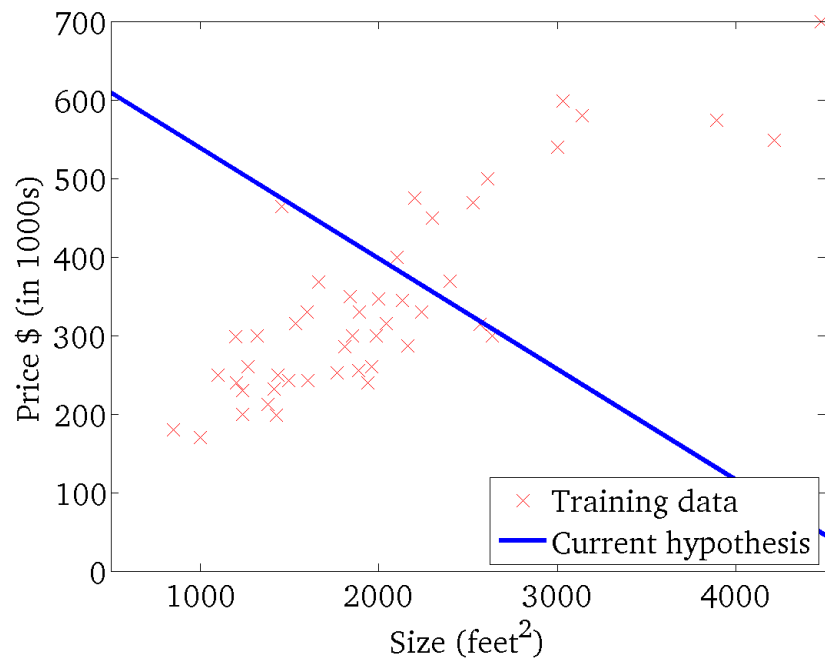
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



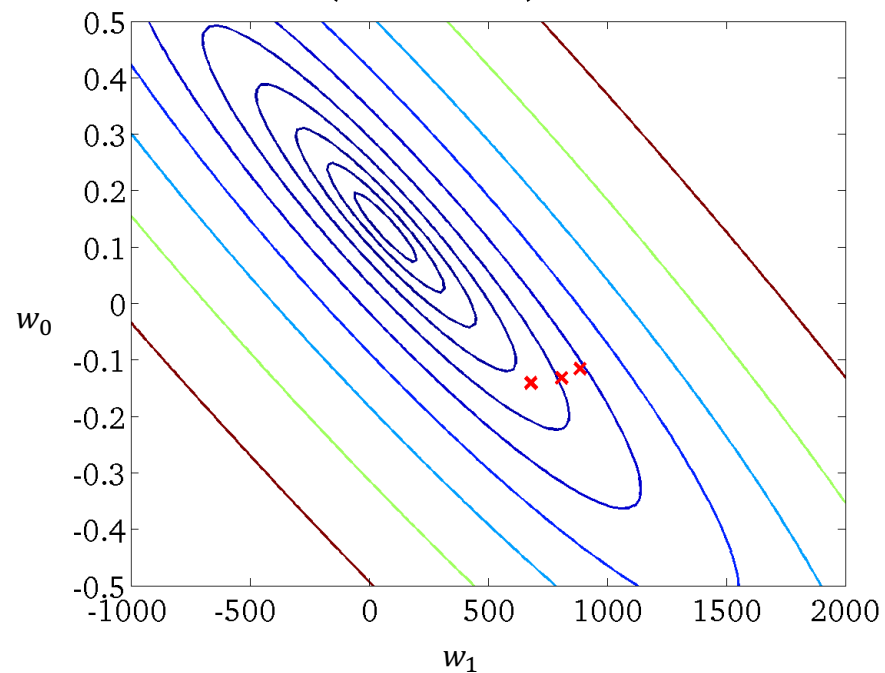
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



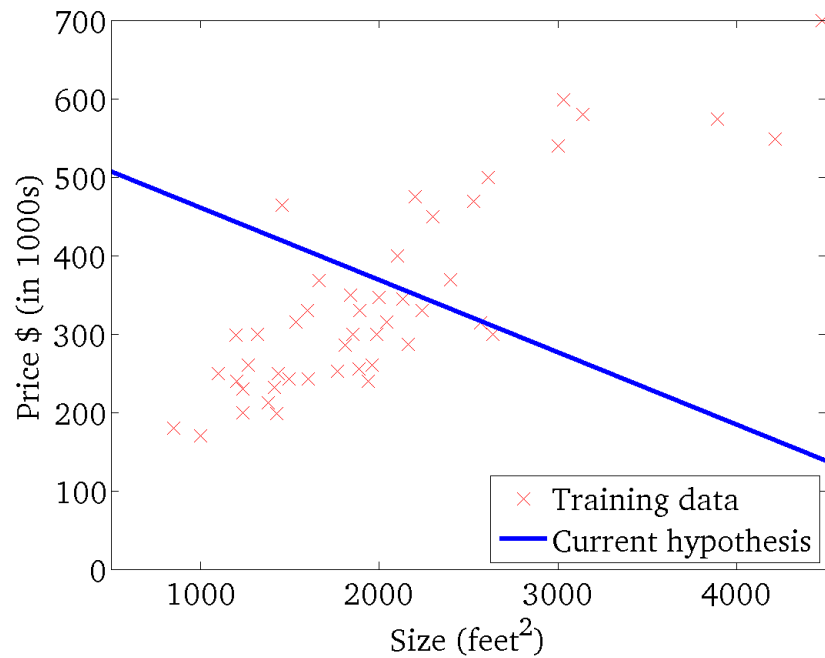
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



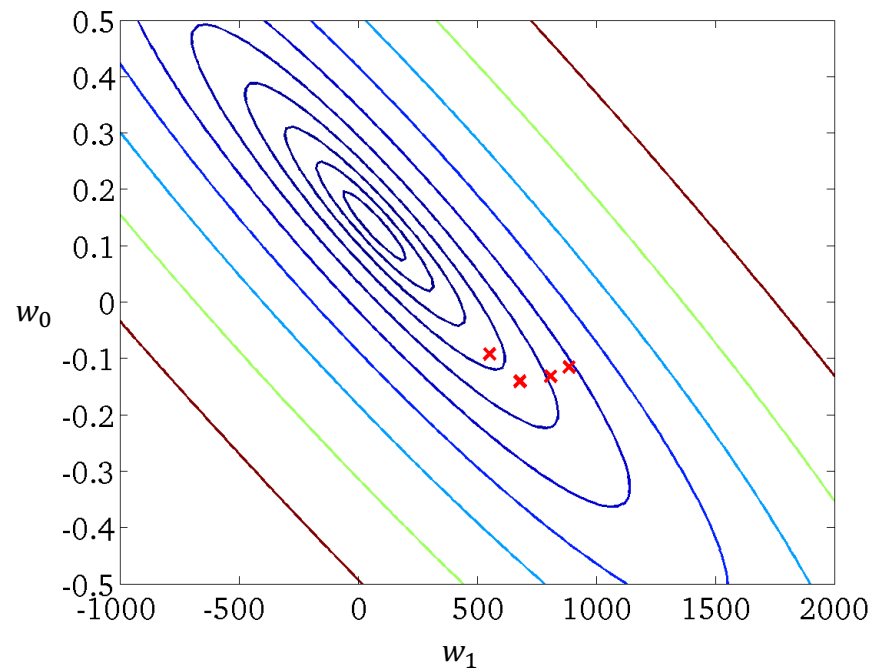
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



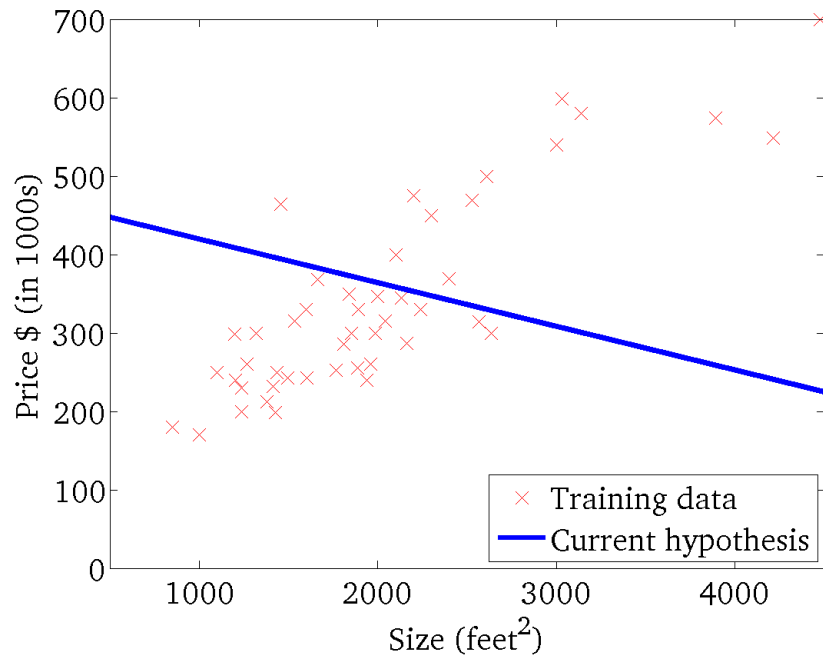
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



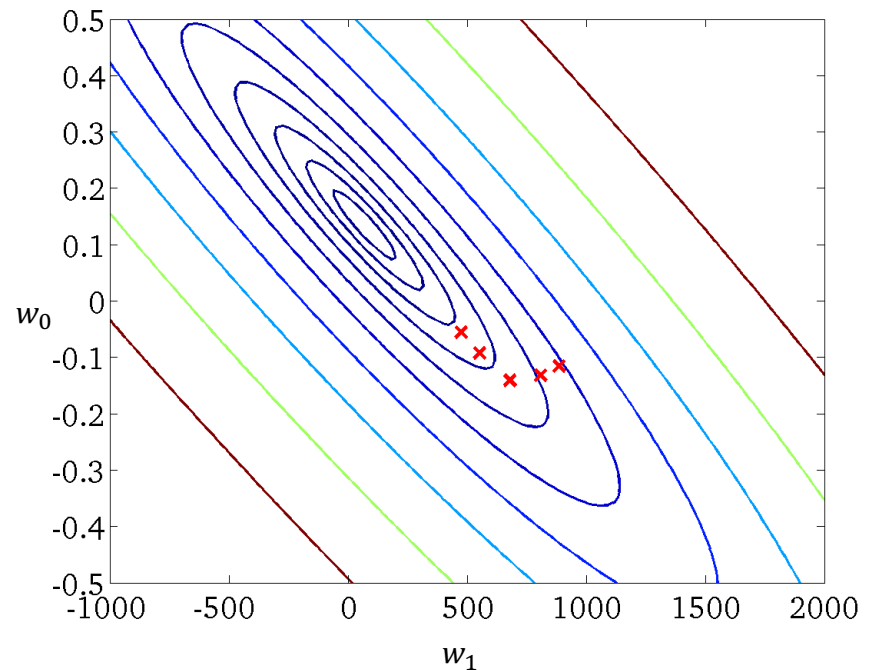
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



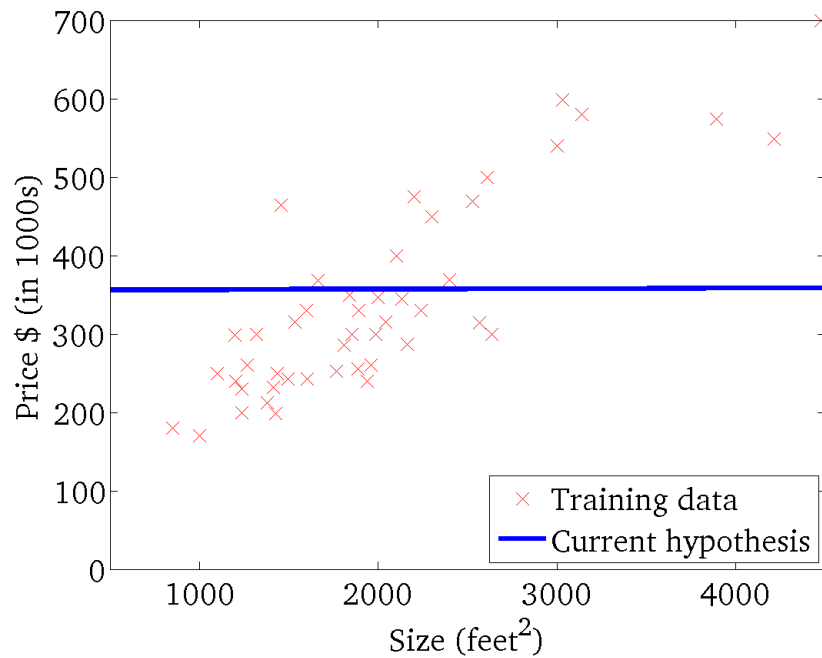
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



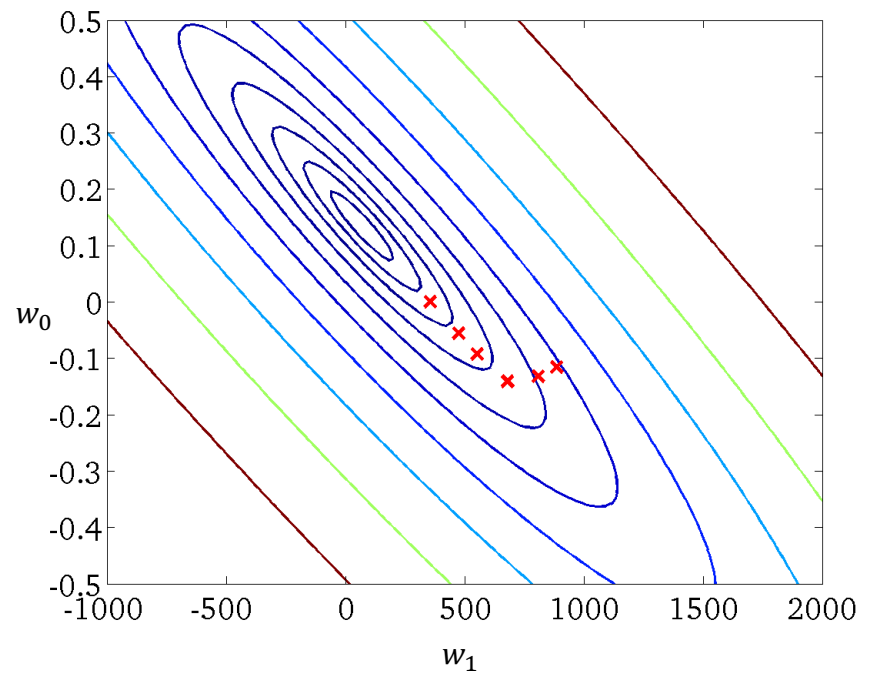
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



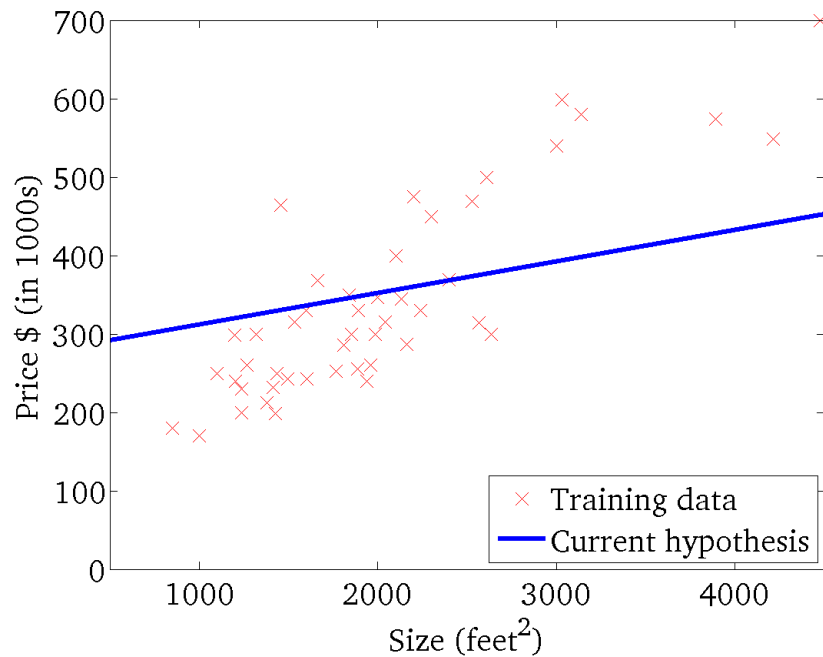
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



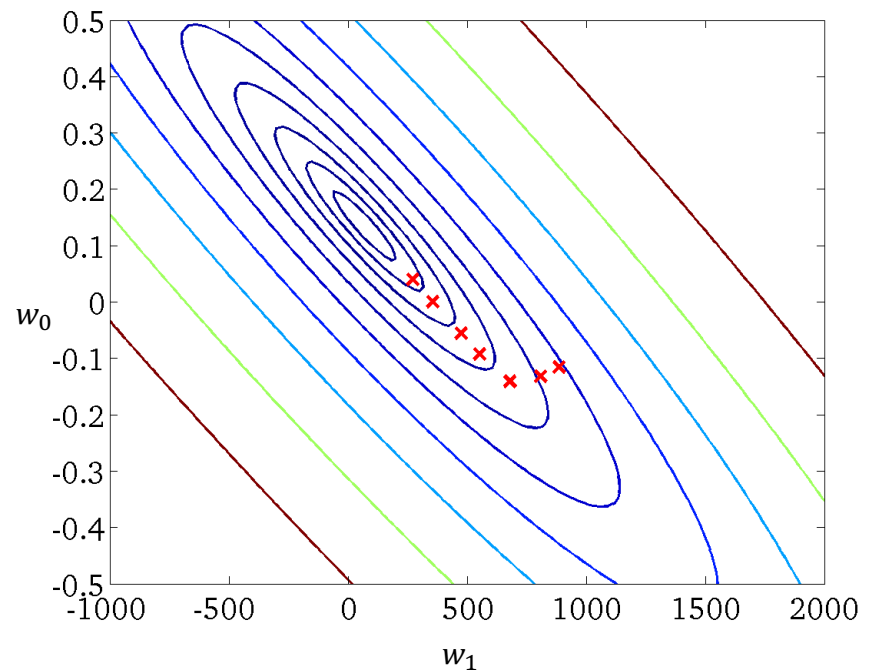
The contours of $L(w_0, w_1)$ and the track of $(w_0^{(t)}, w_1^{(t)})$



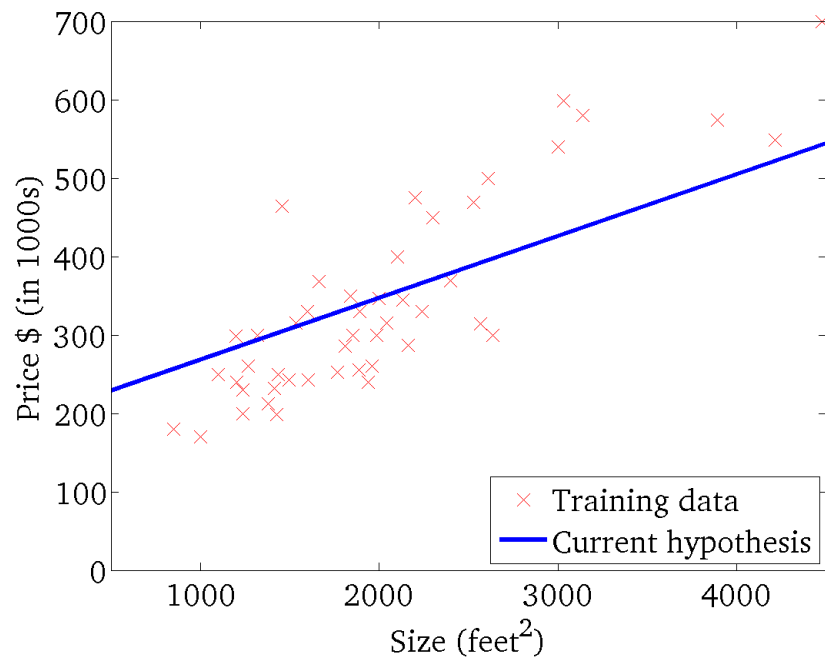
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



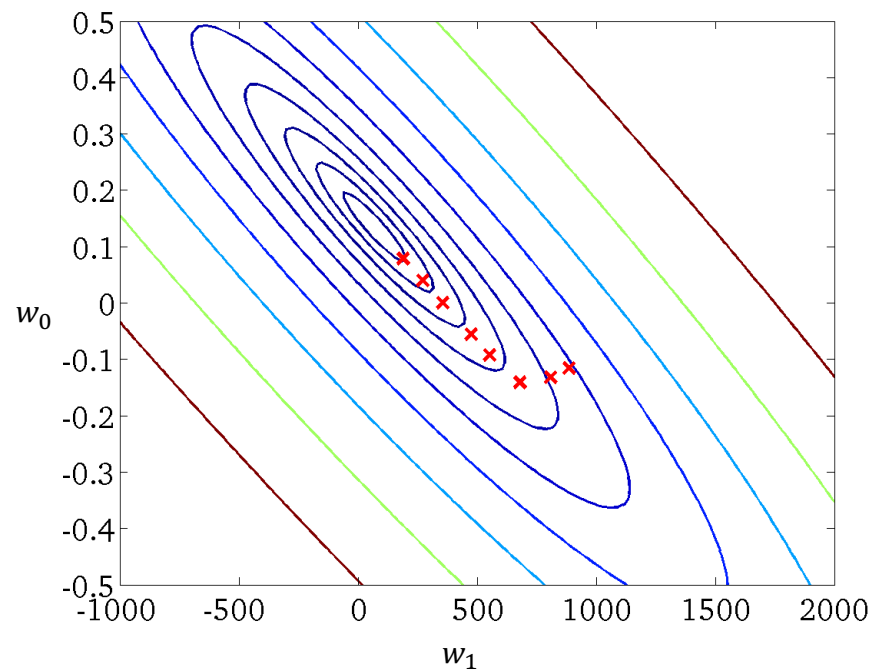
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



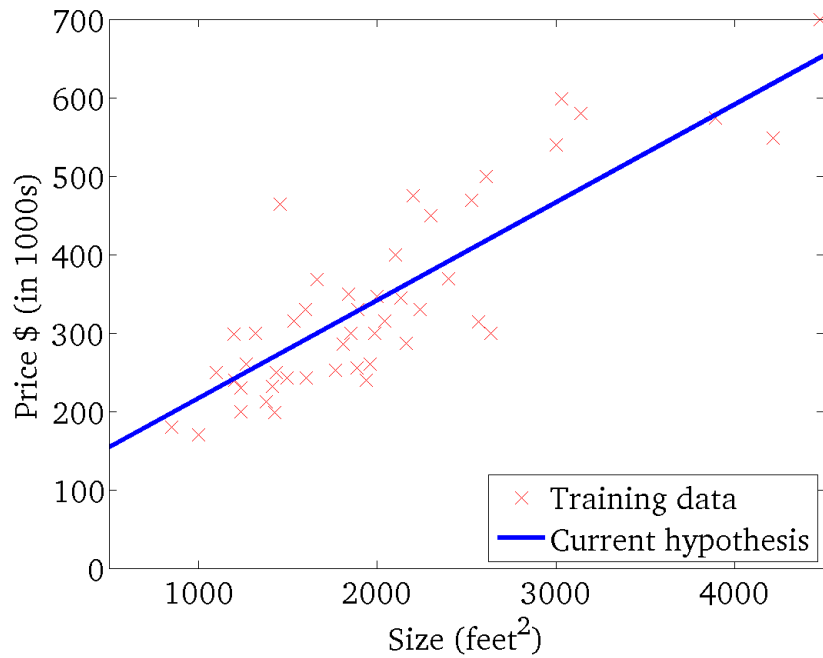
The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



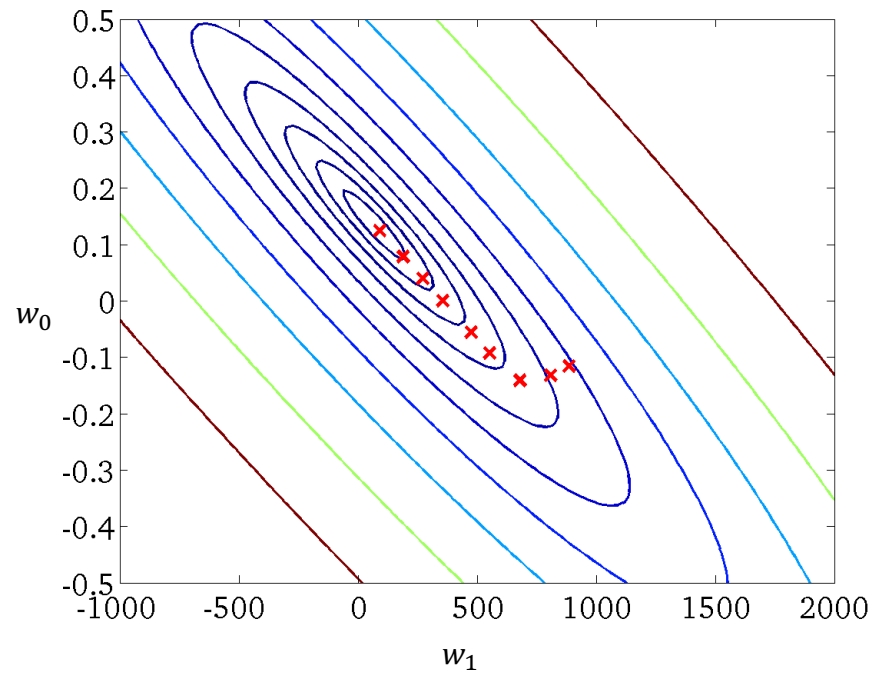
The contours of $L(w_0, w_1)$ and the track of $((w_0^{(t)}, w_1^{(t)}))$



The function $f(x) = w_0^{(t)} + w_1^{(t)}x$



The contours of $L(w_0, w_1)$ and the track of $(w_0^{(t)}, w_1^{(t)})$



Stochastic Gradient Descent

- The GD algorithm need to evaluate the gradient of loss w.r.t. model parameters \mathbf{w} *at every iteration*
- Generally, the gradient takes the form

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

- Every iteration requires computing the gradient on *all data samples in the training dataset*

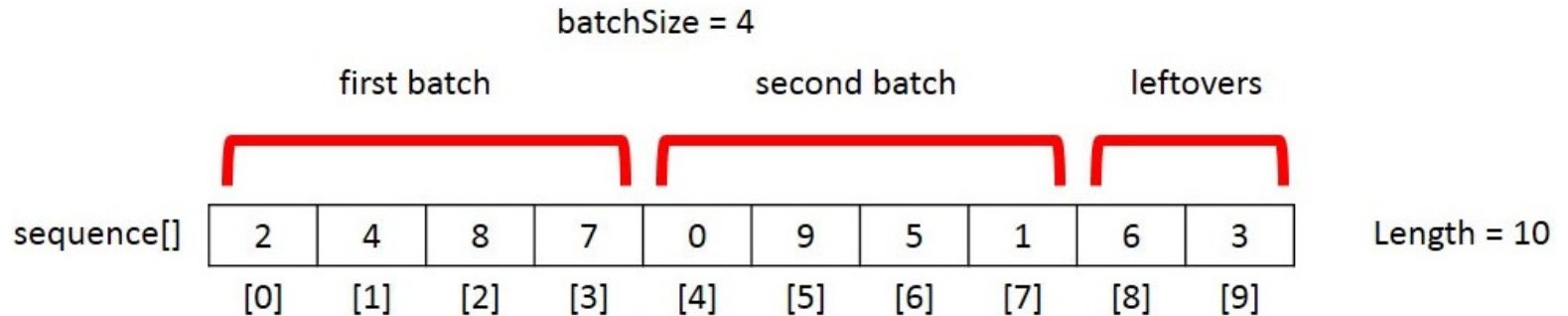
The complexity would be extremely high for large datasets

- To reduce the complexity, we can estimate the gradient $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$ using a small portion of the dataset, *i.e. mini-batch*

- How to obtain the mini-batches?

- Reshuffling

- Segmenting



- Update:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + r \cdot \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

A noisy estimate to the true gradient

where \mathcal{B}_t is a mini-batch of the dataset at the t -th iteration

- **Question:** What's the relation between the stochastic gradient

$$\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

and the ground-truth gradient below?

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$$

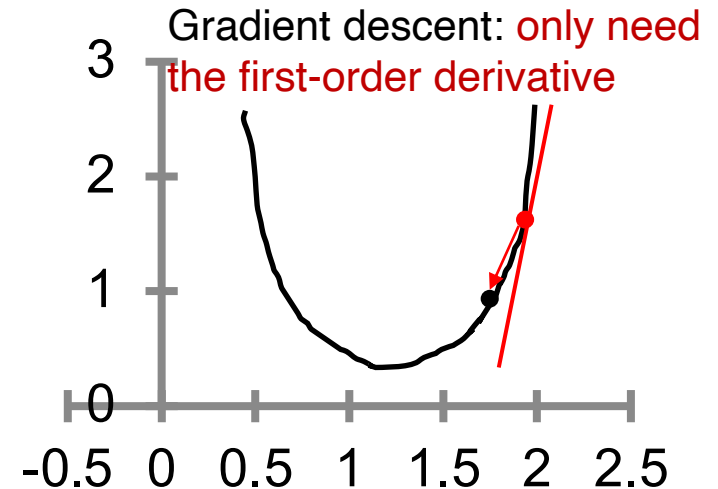
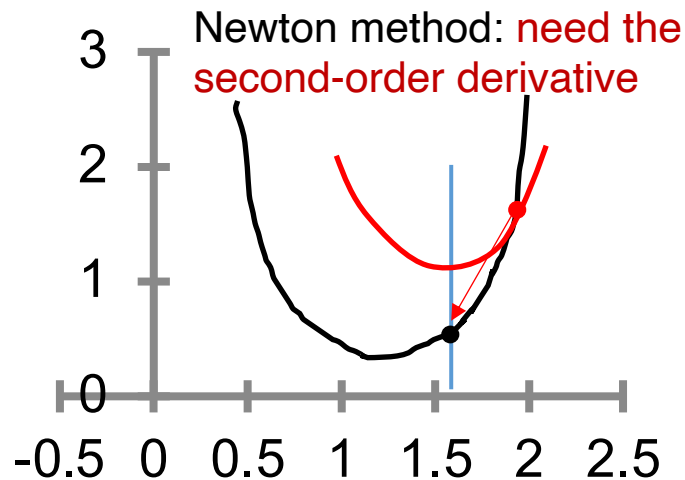
- $\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}}$ is an unbiased estimate to the true $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$, *i.e.*,

$$\mathbb{E}_{\mathcal{B}_t} \left[\frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} \frac{\partial \ell(\mathbf{w}, \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{w}} \right] = \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$$

Other Optimization Methods

- There also exist many other optimization methods

1) Newton method



Advantages

- No need to manually choose the learning rate
- Faster convergence rate

Disadvantages

- More expensive

- 2) Quasi-Newton methods
- 3) Conjugate gradient method
- 4) Coordinated descent method
- ⋮

These methods generally converge faster than gradient descent methods, but are more expensive computationally