

Lecture 5: 7 February, 2022

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

Data Mining and Machine Learning
January–May 2022

Predicting numerical values

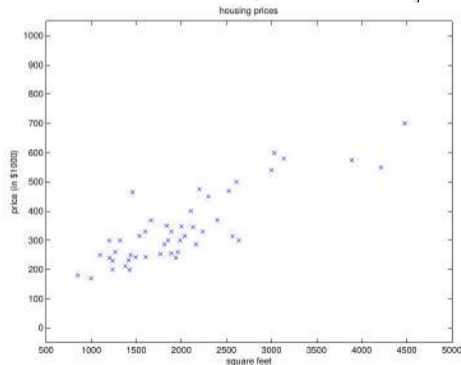
- Data about housing prices
- Predict house price from living area

Living area (feet ²)	Price (1000\$)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Predicting numerical values

- Data about housing prices
- Predict house price from living area
- Scatterplot corresponding to the data
- Fit a function to the points

Living area (feet ²)	Price (1000\$)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Linear predictors

- A richer set of input data

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Linear predictors

- A richer set of input data
- Simplest case: fit a linear function with parameters

$$\theta = (\theta_0, \theta_1, \theta_2)$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Living area (feet ²)	#bedrooms	Price (1000\$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Linear predictors

- A richer set of input data
- Simplest case: fit a linear function with parameters $\theta = (\theta_0, \theta_1, \theta_2)$
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
- Input x may have k features (x_1, x_2, \dots, x_k)

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Linear predictors

- A richer set of input data
- Simplest case: fit a linear function with parameters $\theta = (\theta_0, \theta_1, \theta_2)$
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
- Input x may have k features (x_1, x_2, \dots, x_k)
- By convention, add a dummy feature $x_0 = 1$

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Linear predictors

- A richer set of input data

- Simplest case: fit a linear function with parameters

$$\theta = (\theta_0, \theta_1, \theta_2)$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

- Input x may have k features

$$(x_1, x_2, \dots, x_k) \mapsto (1, x_1, x_2, \dots, x_k)$$

- By convention, add a dummy feature $x_0 = 1$

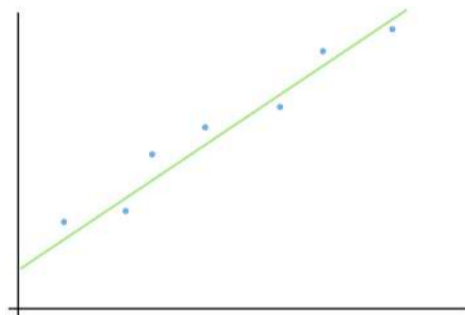
- For k input features

$$h_{\theta}(x) = \sum_{i=0}^k \theta_i x_i$$

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

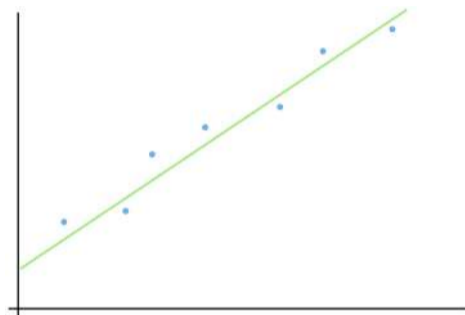
Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output



Finding the best fit line

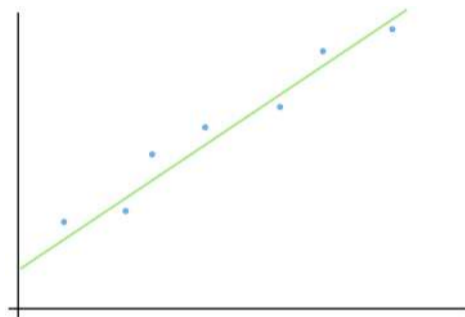
- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output
- How far away is our prediction $h_\theta(x_i)$ from the true answer y_i ?



Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output
- How far away is our prediction $h_\theta(x_i)$ from the true answer y_i ?
- Define a cost (loss) function

$$J(\theta) = \frac{1}{2} \sum_{i=0}^n (h_\theta(x_i) - y_i)^2$$



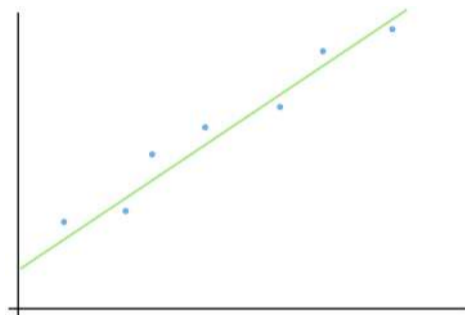
Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output
- How far away is our prediction $h_\theta(x_i)$ from the true answer y_i ?

- Define a cost (loss) function

$$J(\theta) = \frac{1}{2} \sum_{i=0}^n (h_\theta(x_i) - y_i)^2$$

- Essentially, the sum squared error (SSE)



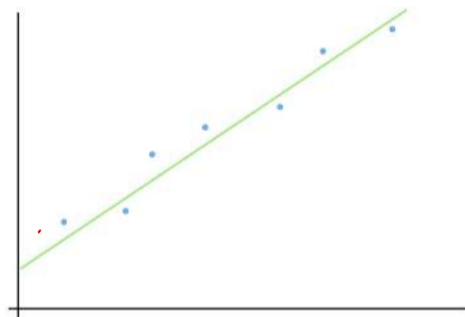
Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
 - Each input x_i is a vector (x_i^1, \dots, x_i^k)
 - Add $x_i^0 = 1$ by convention
 - y_i is actual output
- How far away is our prediction $h_\theta(x_i)$ from the true answer y_i ?

- Define a cost (loss) function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^n (h_\theta(x_i) - y_i)^2$$

- Essentially, the sum squared error (SSE)
- Divide by n , mean squared error (MSE)



Minimizing SSE

- Write x_i as row vector $[1 \quad x_i^1 \quad \cdots \quad x_i^k]$

Minimizing SSE

- Write x_i as row vector $[1 \ x_i^1 \ \dots \ x_i^k]$

- $$X = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^k \\ 1 & x_2^1 & \dots & x_2^k \\ & & \dots & \\ 1 & x_i^1 & \dots & x_i^k \\ & & \dots & \\ 1 & x_n^1 & \dots & x_n^k \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_i \\ \dots \\ y_n \end{bmatrix}$$

- Write θ as column vector, $\theta^T = [\theta_0 \ \theta_1 \ \dots \ \theta_k]$

Minimizing SSE

- Write x_i as row vector $[1 \ x_i^1 \ \dots \ x_i^k]$

- $$X = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^k \\ 1 & x_2^1 & \dots & x_2^k \\ & \dots & & \dots \\ 1 & x_i^1 & \dots & x_i^k \\ & \dots & & \dots \\ 1 & x_n^1 & \dots & x_n^k \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_i \\ \dots \\ y_n \end{bmatrix}$$

- Write θ as column vector, $\theta^T = [\theta_0 \ \theta_1 \ \dots \ \theta_k]$

- $$J(\theta) = \frac{1}{2} \sum_{i=1}^n \underbrace{(h_{\theta}(x_i) - y_i)^2} = \frac{1}{2} \underbrace{(X\theta - y)^T (X\theta - y)}$$

$$\begin{bmatrix} 1 & x_1 & \dots & x_k \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_k \end{bmatrix}$$

$$2\theta_0 \cdot 1 + \theta_1 x_1 + \dots - \theta_k x_k$$

$$\begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} \begin{bmatrix} \Delta y_1 & \Delta y_2 \end{bmatrix}^T \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix}$$

Minimizing SSE

- Write x_i as row vector $[1 \ x_i^1 \ \dots \ x_i^k]$

- $$X = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^k \\ 1 & x_2^1 & \dots & x_2^k \\ & & \dots & \\ 1 & x_i^1 & \dots & x_i^k \\ & & \dots & \\ 1 & x_n^1 & \dots & x_n^k \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_i \\ \dots \\ y_n \end{bmatrix}$$

- Write θ as column vector, $\theta^T = [\theta_0 \ \theta_1 \ \dots \ \theta_k]$

- $$J(\theta) = \frac{1}{2} \sum_{i=1}^n (h_{\theta}(x_i) - y_i)^2 = \frac{1}{2} (X\theta - y)^T (X\theta - y)$$

- Minimize $J(\theta)$ — set $\nabla_{\theta} J(\theta) = 0$

=

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$
- Expand, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y) = 0$

$$(XY)^T = Y^T X^T$$

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$
- Expand, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - \underbrace{y^T X\theta}_{\text{red}} - \underbrace{\theta^T X^T y}_{\text{red}} + y^T y) = 0$
 - Check that $y^T X\theta = \theta^T X^T y = \sum_{i=1}^n h_{\theta}(x_i) \cdot y_i$

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$
- Expand, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - \underbrace{y^T X\theta}_{\text{red}} - \underbrace{\theta^T X^T y}_{\text{red}} + y^T y) = 0$
 - Check that $y^T X\theta = \theta^T X^T y = \sum_{i=1}^n h_{\theta}(x_i) \cdot y_i$
- Combining terms, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - \underbrace{2\theta^T X^T y}_{\text{red}} + y^T y) = 0$

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$
- Expand, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y) = 0$
 - Check that $y^T X\theta = \theta^T X^T y = \sum_{i=1}^n h_{\theta}(x_i) \cdot y_i$
- Combining terms, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - 2\theta^T X^T y + y^T y) = 0$
- After differentiating, $X^T X\theta - X^T y = 0$

Minimizing SSE

- $J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_{\theta} J(\theta) = \nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- To minimize, set $\nabla_{\theta} \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$
- Expand, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y) = 0$
 - Check that $y^T X\theta = \theta^T X^T y = \sum_{i=1}^n h_{\theta}(x_i) \cdot y_i$
- Combining terms, $\frac{1}{2}\nabla_{\theta} (\theta^T X^T X\theta - 2\theta^T X^T y + y^T y) = 0$
- After differentiating, $X^T X\theta - X^T y = 0$
- Solve to get **normal equation**, $\theta = \underline{(X^T X)^{-1} X^T y}$

Minimizing SSE iteratively

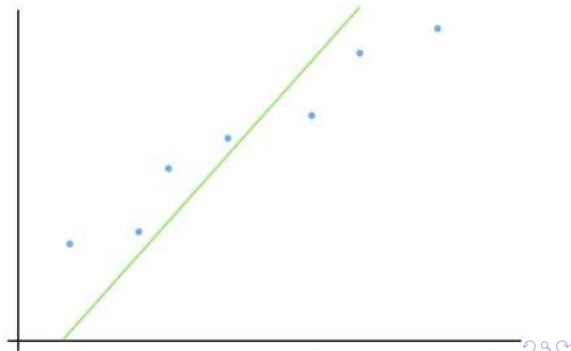
- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution

Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility

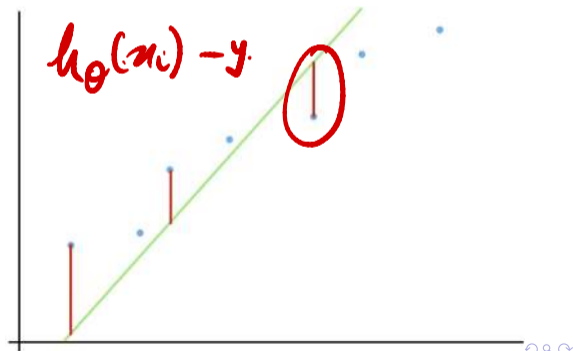
Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess



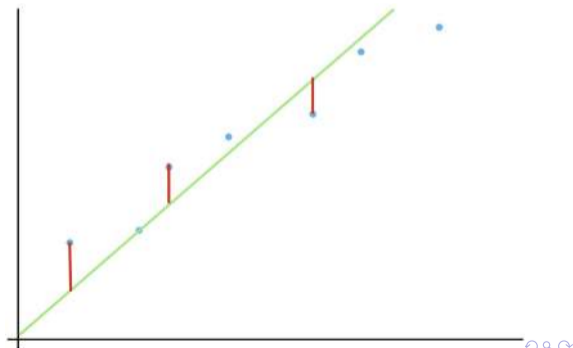
Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess



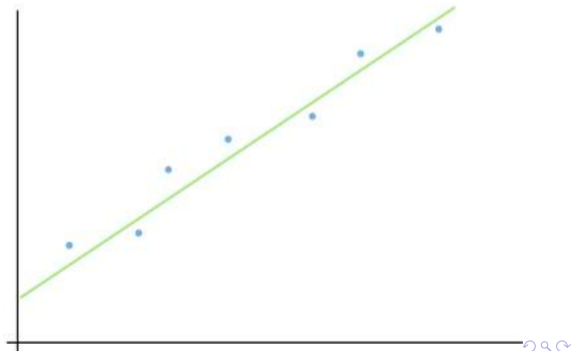
Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess
- Keep adjusting the line to reduce SSE



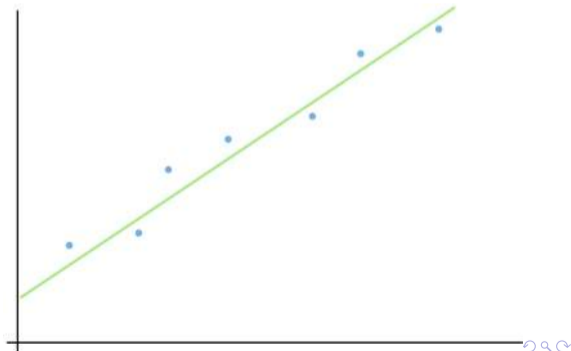
Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess
- Keep adjusting the line to reduce SSE
- Stop when we find the best fit line



Minimizing SSE iteratively

- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
 - Slow if n large, say $n > 10^4$
 - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess
- Keep adjusting the line to reduce SSE
- Stop when we find the best fit line
- How do we adjust the line?



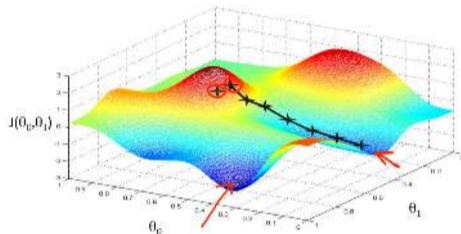
Gradient descent

- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

$$\frac{\Delta y}{\Delta x}$$



Gradient descent

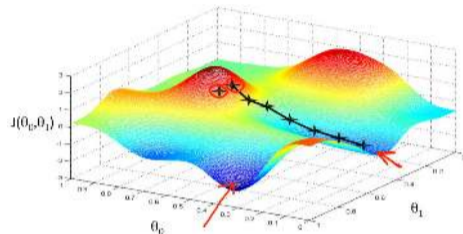
- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

- Adjust each parameter against gradient

- $\theta_i = \theta_i - \underbrace{\alpha}_{\text{learning rate}} \frac{\partial}{\partial \theta_i} J(\theta)$



Gradient descent

- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

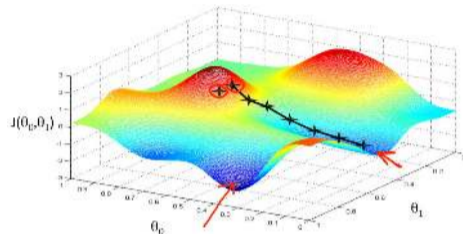
- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

- Adjust each parameter against gradient

- $\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$

- For a single training sample (x, y)

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_{\theta}(x) - y)^2$$



Gradient descent

- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

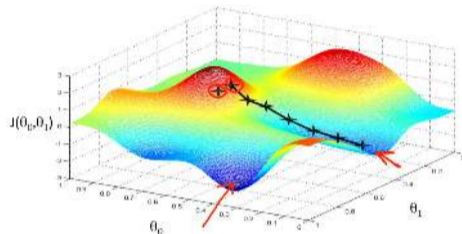
- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

- Adjust each parameter against gradient

- $\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$

- For a single training sample (x, y)

$$\begin{aligned} \frac{\partial}{\partial \theta_i} J(\theta) &= \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_i} (h_{\theta}(x) - y) \end{aligned}$$



Gradient descent

- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

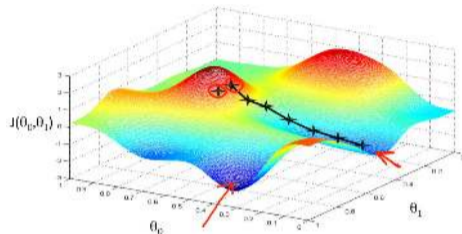
- Adjust each parameter against gradient

- $\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$

- For a single training sample (x, y)

$$\begin{aligned} \frac{\partial}{\partial \theta_i} J(\theta) &= \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_i} (h_{\theta}(x) - y) \end{aligned}$$

$$= (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_i} \left[\left(\sum_{j=1}^k \theta_j(x) \right) - y \right]$$



$$\frac{d}{d\theta} (f(\theta))^2 = 2 f(\theta) \frac{d}{d\theta} f(\theta)$$

Gradient descent

- How does cost vary with parameters

$$\theta = (\theta_0, \theta_1, \dots, \theta_k)?$$

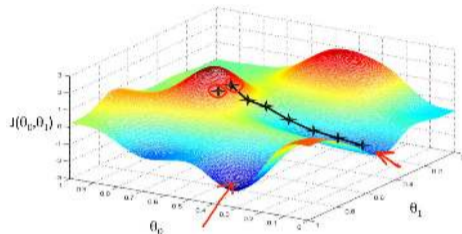
- Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$

- Adjust each parameter against gradient

- $\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$

- For a single training sample (x, y)

$$\begin{aligned} \frac{\partial}{\partial \theta_i} J(\theta) &= \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_{\theta}(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_i} (h_{\theta}(x) - y) \\ &= (h_{\theta}(x) - y) \frac{\partial}{\partial \theta_i} \left[\left(\sum_{j=1}^k \theta_j(x) \right) - y \right] \end{aligned}$$



$$\frac{d}{d\theta_i} (\theta_i x_i - y) = x_i$$

$$= (h_{\theta}(x) - y) \cdot x_i$$

Gradient descent

- For a single training sample (x, y) , $\frac{\partial}{\partial \theta_i} J(\theta) = (h_{\theta}(x) - y) \cdot x_i$

Gradient descent

- For a single training sample (x, y) , $\frac{\partial}{\partial \theta_i} J(\theta) = (h_\theta(x) - y) \cdot x_i$
- Over the entire training set, $\frac{\partial}{\partial \theta_i} J(\theta) = \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i$

Gradient descent

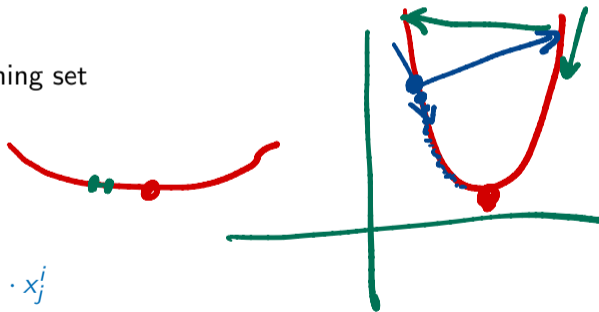
- For a single training sample (x, y) , $\frac{\partial}{\partial \theta_i} J(\theta) = (h_\theta(x) - y) \cdot x_i$
- Over the entire training set, $\frac{\partial}{\partial \theta_i} J(\theta) = \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i$

Batch gradient descent

- Compute $h_\theta(x_j)$ for entire training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$
- Adjust each parameter

$$\begin{aligned}\theta_i &= \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta) \\ &= \theta_i - \alpha \cdot \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i\end{aligned}$$

- Repeat until convergence



Gradient descent

- For a single training sample (x, y) , $\frac{\partial}{\partial \theta_i} J(\theta) = (h_\theta(x) - y) \cdot x_i$
- Over the entire training set, $\frac{\partial}{\partial \theta_i} J(\theta) = \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i$

Batch gradient descent

- Compute $h_\theta(x_j)$ for entire training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$
- Adjust each parameter

$$\begin{aligned}\theta_i &= \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta) \\ &= \theta_i - \alpha \cdot \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i\end{aligned}$$

- Repeat until convergence

Stochastic gradient descent

- For each input x_j , compute $h_\theta(x_j)$
- Adjust each parameter —
 $\theta_i = \theta_i - \alpha \cdot (h_\theta(x_j) - y) \cdot x_j^i$

Small batches

Gradient descent

- For a single training sample (x, y) , $\frac{\partial}{\partial \theta_i} J(\theta) = (h_\theta(x) - y) \cdot x_i$
- Over the entire training set, $\frac{\partial}{\partial \theta_i} J(\theta) = \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i$

Batch gradient descent

- Compute $h_\theta(x_j)$ for entire training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$

- Adjust each parameter

$$\begin{aligned}\theta_i &= \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta) \\ &= \theta_i - \alpha \cdot \sum_{j=1}^n (h_\theta(x_j) - y_j) \cdot x_j^i\end{aligned}$$

- Repeat until convergence

Stochastic gradient descent

- For each input x_j , compute $h_\theta(x_j)$
- Adjust each parameter —
 $\theta_i = \theta_i - \alpha \cdot (h_\theta(x_j) - y) \cdot x_j^i$

Pros and cons

- Faster progress for large batch size
- May oscillate indefinitely