Lecture 6: 22 April, 2021

Madhavan Mukund

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

Data Mining and Machine Learning
April–July 2021

Predicting numerical values

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

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

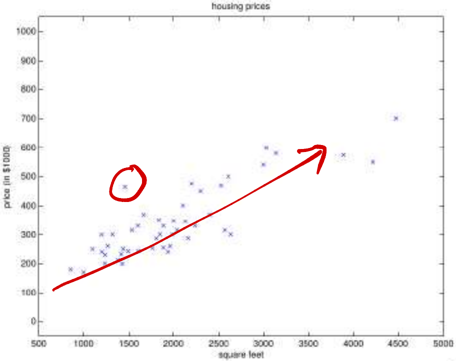
Features Outcome

Predicting numerical values

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

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

- Scatterplot corresponding to the data
- Fit a function to the points



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
⋮	⋮	⋮

2 features

Outcome

Linear predictors

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

x_1 Living area (feet^2)	x_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 $(x_1, x_2, \ldots, x_k)$

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 $(x_1, x_2, \ldots, x_k)$
- 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
⋮	⋮	⋮

$\theta_0 \cdot x_0$ $\quad x_0 = 1$ always

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 $(x_1, x_2, \ldots, x_k)$
- 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
⋮	⋮	⋮

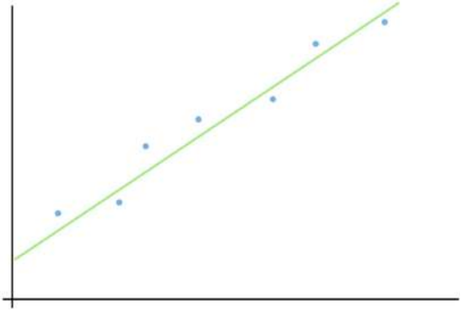
$x_0 = 1$

$\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_k x_k$

Finding the best fit line

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

y_i



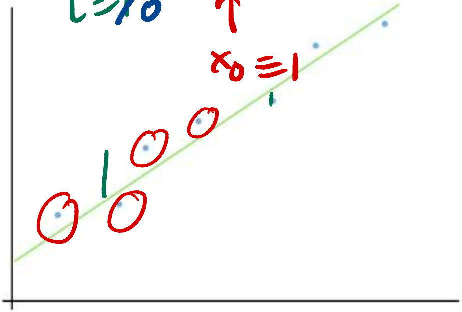
Finding the best fit line

- Training input is
  $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Each input x_i is a vector $(x_i^1, \ldots, x_i^k)$
  - 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?

$(1, x_1 \ldots x_n)$

$\sum_{i=\cancel{1}0}^{k} \theta_i x_i + \cancel{\theta_0}$

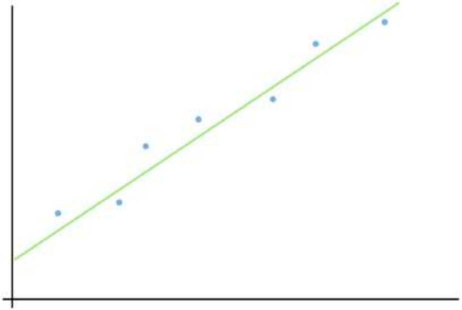
$x_0 = 1$



Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Each input x_i is a vector $(x_i^1, \ldots, x_i^k)$
  - 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

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



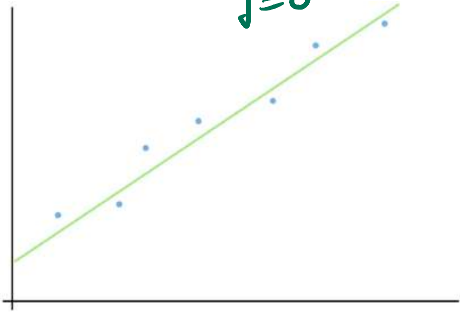
Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Each input x_i is a vector $(x_i^1, \ldots, x_i^k)$
  - 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)

$h_\theta(x_i)$ $\sum_{j=0}^{k}$



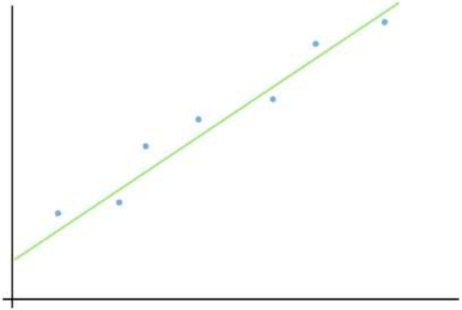
n fixed, SSE is fine

Finding the best fit line

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Each input x_i is a vector $(x_i^1, \ldots, x_i^k)$
  - 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)
- Divide by n, mean squared error (MSE)



Minimizing SSE

- Write x_i as row vector $\begin{bmatrix} 1 & x_i^1 & \cdots & x_i^k \end{bmatrix}$

Minimizing SSE

- Write x_i as row vector $\begin{bmatrix} 1 & x_i^1 & \cdots & x_i^k \end{bmatrix}$
- $X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^k \\ 1 & x_2^1 & \cdots & x_2^k \\ & & \cdots & \\ 1 & x_i^1 & \cdots & x_i^k \\ & & \cdots & \\ 1 & x_n^1 & \cdots & x_n^k \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_i \\ \cdots \\ y_n \end{bmatrix}$
- Write θ as column vector, $\underline{\theta^T} = \underline{\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_k \end{bmatrix}}$

$$\theta$$

$$X \cdot \theta \quad \begin{bmatrix} 1 & x_i^1 & \cdots & x_i^k \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \\ \theta_k \end{bmatrix}$$

Minimizing SSE

- Write x_i as row vector $\begin{bmatrix} 1 & x_i^1 & \cdots & x_i^k \end{bmatrix}$
- $X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^k \\ 1 & x_2^1 & \cdots & x_2^k \\ & & \cdots & \\ 1 & x_i^1 & \cdots & x_i^k \\ & & \cdots & \\ 1 & x_n^1 & \cdots & x_n^k \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_i \\ \cdots \\ y_n \end{bmatrix}$
- Write θ as column vector, $\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_k \end{bmatrix}$
- $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y)$

$$X\theta = \begin{bmatrix} h_\theta(x_1) \\ \vdots \\ h_\theta(x_n) \end{bmatrix}$$

$h_\theta(x_1) - y_1$
$h_\theta(x_2) - y_2$
$\vdots$
$h_\theta(x_n) - y_n$

Minimizing SSE

- Write x_i as row vector $\begin{bmatrix} 1 & x_i^1 & \cdots & x_i^k \end{bmatrix}$
- $X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^k \\ 1 & x_2^1 & \cdots & x_2^k \\ & & \cdots & \\ 1 & x_i^1 & \cdots & x_i^k \\ & & \cdots & \\ 1 & x_n^1 & \cdots & x_n^k \end{bmatrix}$, $y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_i \\ \cdots \\ y_n \end{bmatrix}$
- Write θ as column vector, $\theta^T = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_k \end{bmatrix}$
- $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x_i) - y_i)^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- Minimize $J(\theta)$ — set $\nabla_\theta\, J(\theta) = 0$

Minimizing SSE

- $J(\theta) = \dfrac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_\theta \; J(\theta) = \nabla_\theta \; \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- 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, $\frac{1}{2}\nabla_\theta\ (\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y) = 0$

$(X\theta)^T = \theta^T X^T$

Minimizing SSE

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

Minimizing SSE

- $J(\theta) = \dfrac{1}{2}(X\theta - y)^T(X\theta - y)$
- $\nabla_\theta\ J(\theta) = \nabla_\theta\ \frac{1}{2}(X\theta - y)^T(X\theta - y)$
- 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 $y^T X\theta = \theta^T X^T y = \displaystyle\sum_{i=1}^{n} h_\theta(x_i) \cdot y_i$
- Combining terms, $\frac{1}{2}\nabla_\theta\ (\theta^T X^T X\theta - \underline{2\theta^T X^T y} + y^T y) = 0$

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, $X^T X\theta - X^T y = 0$

$\theta^T\theta = \theta^2$

2

$\frac{1}{\cancel{2}} \cdot \cancel{2} \cdot X^T X\theta \qquad \cancel{2}X^T y$

$X^T X\theta = X^T y$

$\theta = (X^T X)^{-1} X^T y$

Minimizing SSE

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

Minimizing SSE iteratively

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

Minimizing SSE iteratively

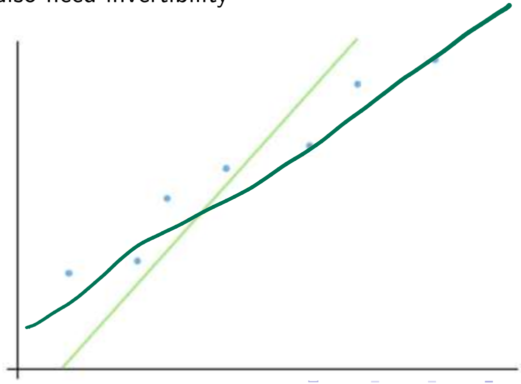
- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution

Linear $\boxed{\text{regression}}$

- Computational challenges
  - Slow if n large, say $n > 10^4$
  - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility

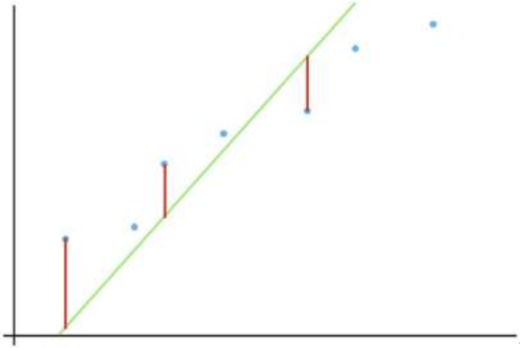
Minimizing SSE iteratively

- 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



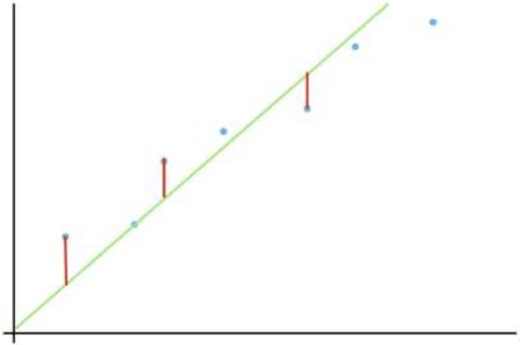
Minimizing SSE iteratively

- 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

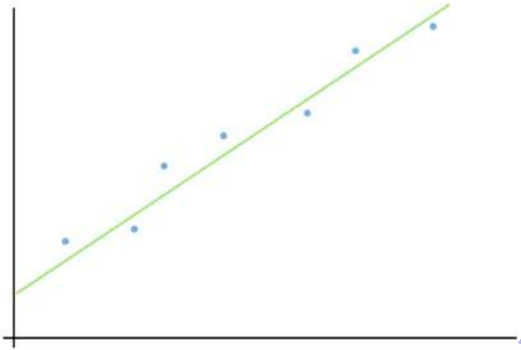
- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
  - Slow if n large, say $n > 10^4$
  - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess
- Keep adjusting the line to reduce SSE



Minimizing SSE iteratively

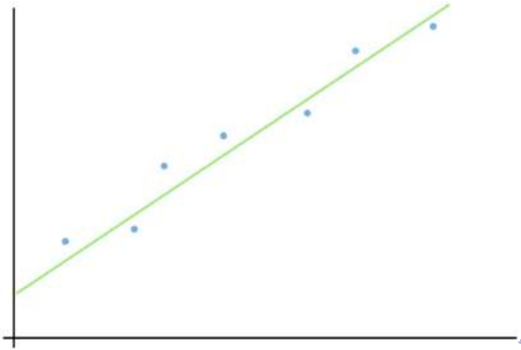
- Normal equation $\theta = (X^T X)^{-1} X^T y$ is a closed form solution
- Computational challenges
  - Slow if n large, say $n > 10^4$
  - Matrix inversion $(X^T X)^{-1}$ is expensive, also need invertibility
- Iterative approach, make an initial guess
- Keep adjusting the line to reduce SSE
- Stop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

Stop when incremental progress $< \varepsilon$



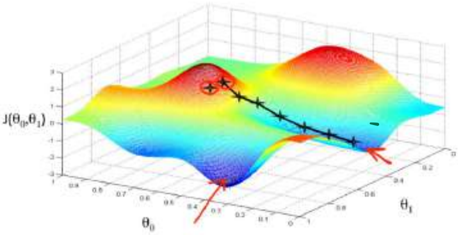
Minimizing SSE iteratively

- 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

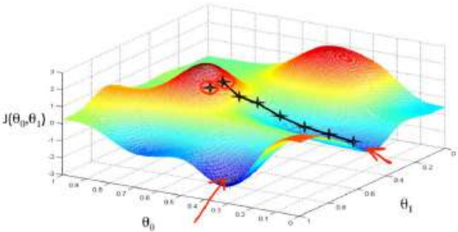
- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - Gradients $\dfrac{\partial}{\partial \theta_i} J(\theta)$



Gradient descent

- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - Gradients $\frac{\partial}{\partial \theta_i} J(\theta)$
- Adjust each parameter against gradient
  - $\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$

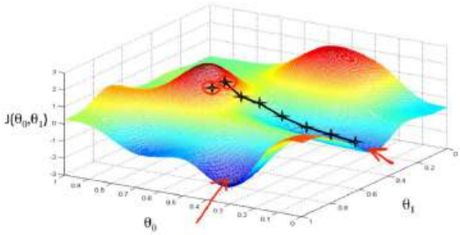
step size



Gradient descent

- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - 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)

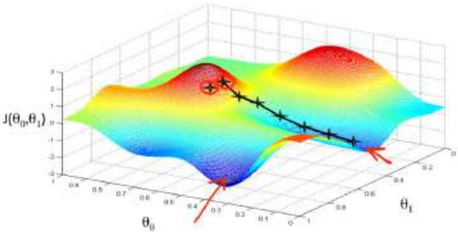
$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_\theta(x) - y)^2$$



Gradient descent

- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - 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)

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



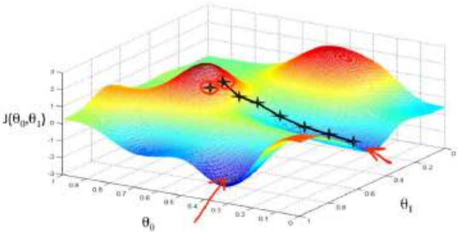
$\frac{d}{d\theta}(f(\theta))^2$ $\quad 2 \cdot f(\theta) \cdot f'(\theta)$

Gradient descent

- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - 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)

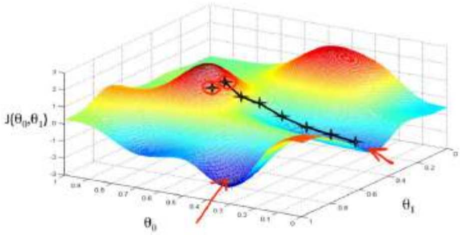
n = 1

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



Gradient descent

- How does cost vary with parameters $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$?
  - 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)

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

$\cancel{\theta_0 x_0} + \cancel{\theta_1 x_1} + \theta_i x_i + \cdots + \cancel{\theta_k x_k}$

Gradient descent

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

Gradient descent

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

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, $\frac{\partial}{\partial \theta_i} J(\theta) = \sum_{j=1}^{n} (\underline{h_\theta(x_j)} - y_j) \cdot x_j^i$

Batch gradient descent

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

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

- Repeat until convergence

Fix θ $\rightarrow$ $h_\theta(x_j)$ for all inputs

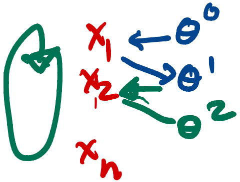
$\downarrow$

$\frac{\partial J}{\partial \theta_i}$

θ'

Gradient descent

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



Batch gradient descent

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

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

- Repeat until convergence

Stochastic gradient descent

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

Mini batch stochastic G.D.

M rows → Adj θ

Gradient descent

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

Batch gradient descent

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

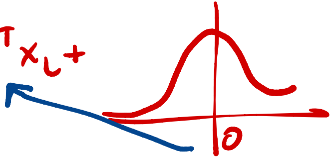
Regression and SSE loss

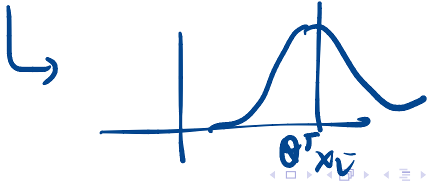
- Training input is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = \theta^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \theta^T x_i$

$y = mx + c$

$y_i = mx_i + c$ + noise

$y_i = \theta^T x_i +$





Regression and SSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = \theta^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \theta^T x_i$
- Model gives us an estimate for θ, so regression learns μ_i for each x_i

Regression and SSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = \theta^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \theta^T x_i$
- Model gives us an estimate for θ, so regression learns μ_i for each x_i
- Want Maximum Likelihood Estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} P(y_i \mid x_i; \theta)$$

100 coin tosses

outcome 63 heads 37 tails

Estimate p(heads)?

0.63

0.60 ? 0.65 ?

Prob. $\hat{p}^{63}(1-\hat{p})^{37}$

Given p(heads) = $\hat{p}$

$P(O \mid \hat{p})$ = likelihood

$\max_{\hat{p}}$

Regression and SSE loss

- Training input is $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
  - Noisy outputs from a linear function
  - $y_i = \theta^T x_i + \epsilon$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2)$: Gaussian noise, mean 0, fixed variance σ^2
  - $y_i \sim \mathcal{N}(\mu_i, \sigma^2)$, $\mu_i = \theta^T x_i$
- Model gives us an estimate for θ, so regression learns μ_i for each x_i
- Want Maximum Likelihood Estimator (MLE) — maximize

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} P(y_i \mid x_i; \theta)$$

- Instead, maximize log likelihood

$$\ell(\theta) = \log\left(\prod_{i=1}^{n} P(y_i \mid x_i; \theta)\right) = \sum_{i=1}^{n} \log(P(y_i \mid x_i; \theta))$$

Coin

$\hat{p} = 0.1$

$\hat{p} = 0.6$

HTHTH

$(0.1)^3(0.9)^2$

$(0.6)^3(0.4)^2$

$\hat{p} \cdot (1-\hat{p}) \cdot \hat{p} \, (1-\hat{p}) \, \hat{p}$

$O_1 \quad O_2 \quad O_5$

$\log\left(\prod_{i=1}^{5} P(O_i \mid \hat{p})\right)$

$\sum_{i=1}^{5} \log(P(O_i \mid \hat{p}))$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}}$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$
- Log likelihood

$$\ell(\theta) = \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}} \right)$$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$
- Log likelihood (assuming natural logarithm)

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\underline{\frac{1}{\sqrt{2\pi\sigma^2}}} \times \underline{e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}}\right) = \underline{n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)} - \underline{\sum_{i=1}^{n} \frac{(y-\theta^T x_i)^2}{2\sigma^2}}$$

$\log(a \cdot b)$

$\log a + \log b$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$
- Log likelihood (assuming natural logarithm)
$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}\right) = \cancel{n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)} - \sum_{i=1}^{n} \frac{(y-\theta^T x_i)^2}{2\sigma^2}$$
- To maximize $\ell(\theta)$ with respect to θ, ignore all terms that do not depend on θ

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$

- Log likelihood (assuming natural logarithm)

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}\right) = n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) \boxed{- \sum_{i=1}^{n} \frac{(y-\theta^T x_i)^2}{2\sigma^2}}$$

- To maximize $\ell(\theta)$ with respect to θ, ignore all terms that do not depend on θ

- Optimum value of θ is given by

$$\hat{\theta}_{\text{MSE}} = \arg\max_{\theta} \left[- \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \right]$$

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$

- Log likelihood (assuming natural logarithm)

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}\right) = n \log\left(\cancel{\frac{1}{\sqrt{2\pi\sigma^2}}}\right) - \sum_{i=1}^{n} \frac{(y-\theta^T x_i)^2}{2\sigma^2}$$

- To maximize $\ell(\theta)$ with respect to θ, ignore all terms that do not depend on θ

- Optimum value of θ is given by

$$\hat{\theta}_{\text{MSE}} = \arg\max_{\theta} \left[-\sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \right] = \arg\min_{\theta} \left[\sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \right]$$

SSE

Log likelihood and SSE loss

- $y_i = \mathcal{N}(\mu_i, \sigma^2)$, so $P(y_i \mid x_i; \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu_i)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}}$

- Log likelihood (assuming natural logarithm)

$$\ell(\theta) = \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^T x_i)^2}{2\sigma^2}} \right) = n \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \sum_{i=1}^{n} \frac{(y-\theta^T x_i)^2}{2\sigma^2}$$

- To maximize $\ell(\theta)$ with respect to θ, ignore all terms that do not depend on θ

- Optimum value of θ is given by

$$\hat{\theta}_{\text{MSE}} = \arg\max_{\theta} \left[-\sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \right] = \arg\min_{\theta} \left[\sum_{i=1}^{n} (y_i - \theta^T x_i)^2 \right]$$

- Assuming data points are generated by linear function and then perturbed by Gaussian noise, SSE is the "correct" loss function to maximize likelihood