Machine Learning Department
School of Computer Science
Carnegie Mellon University

Linear Regression

+

Optimization for ML

Matt Gormley & Henry Chai
Lecture 8
Sep. 22, 2021

Reminders

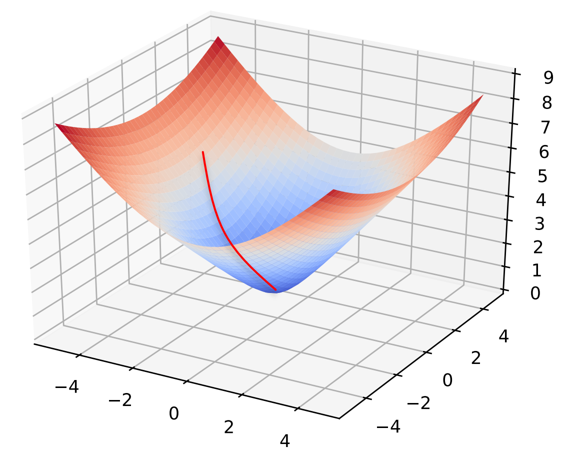
- **Homework 3: KNN, Perceptron, Lin.Reg.**
  - **Out: Mon, Sep. 20**
  - **Due: Sun, Sep. 26 at 11:59pm**
  - **IMPORTANT: you may only use 2 grace days on Homework 3**

OPTIMIZATION METHOD #1: GRADIENT DESCENT

Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: **while** not converged **do**
4: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
5: **return** $\boldsymbol{\theta}$



There are many possible ways to detect **convergence**. For example, we could check whether the L2 norm of the gradient is below some small tolerance.

$$||\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})||_2 \leq \epsilon$$

Alternatively we could check that the reduction in the objective function from one iteration to the next is small.

GRADIENT DESCENT FOR LINEAR REGRESSION

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N e_i^2$$
$$= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2$$
$$= \frac{1}{N} \sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$

4. Solve the unconstrained optimization problem via favorite method:

- *gradient descent*
- *closed form*
- *stochastic gradient descent*
- ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

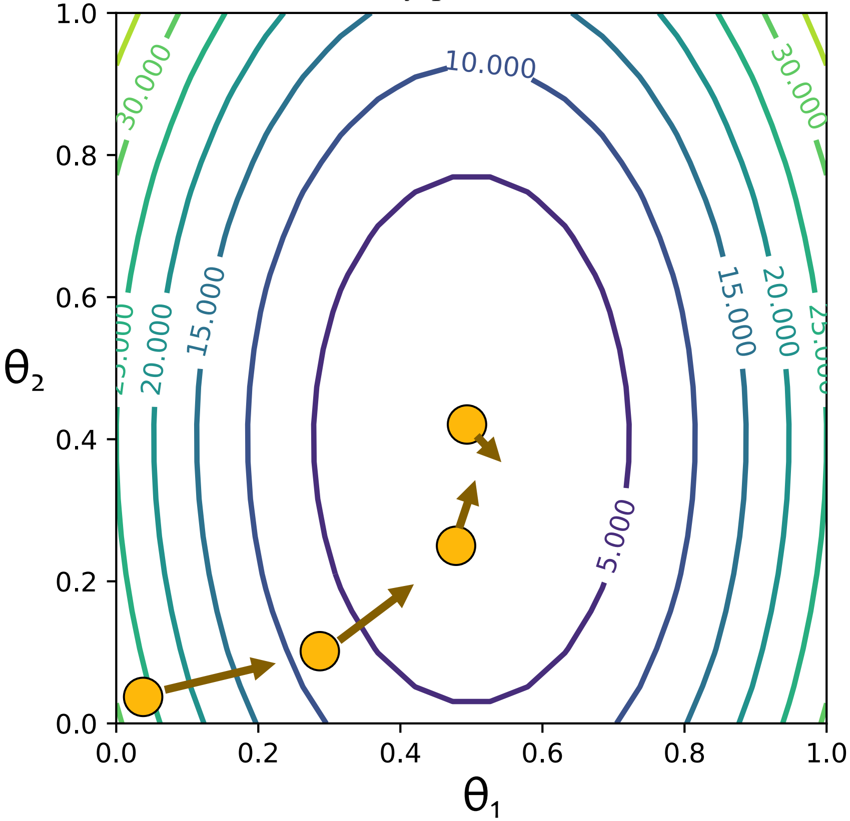
$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random $\boldsymbol{\theta}$
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$



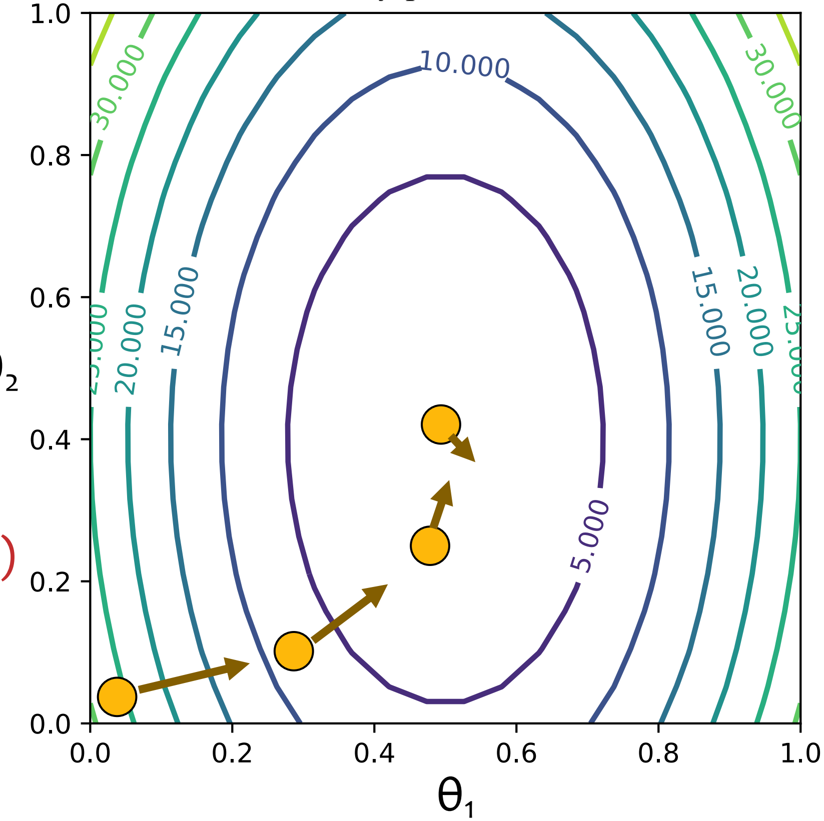
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

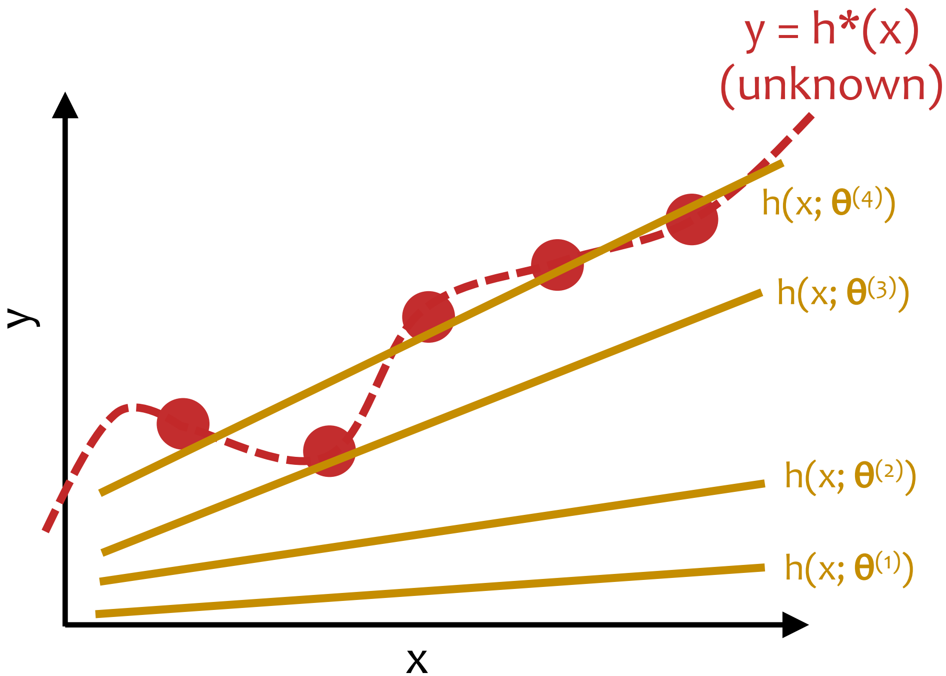
Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random $\boldsymbol{\theta}$
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

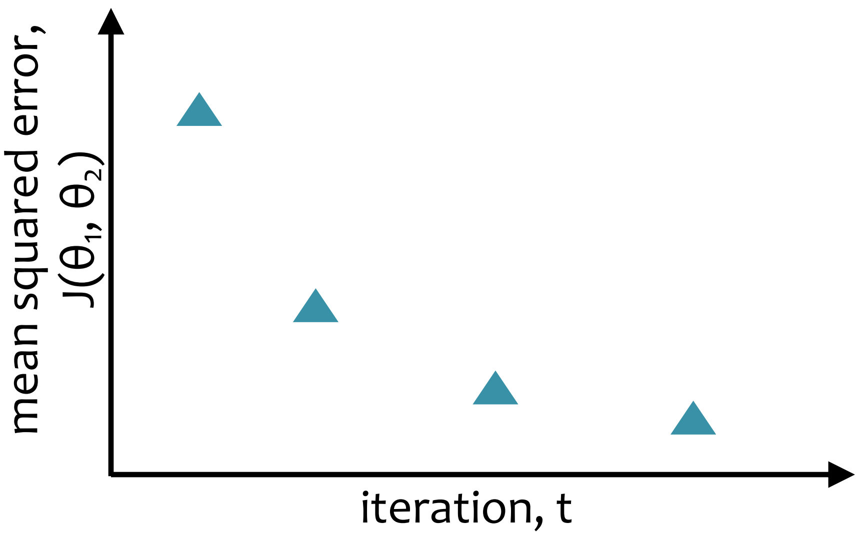
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$



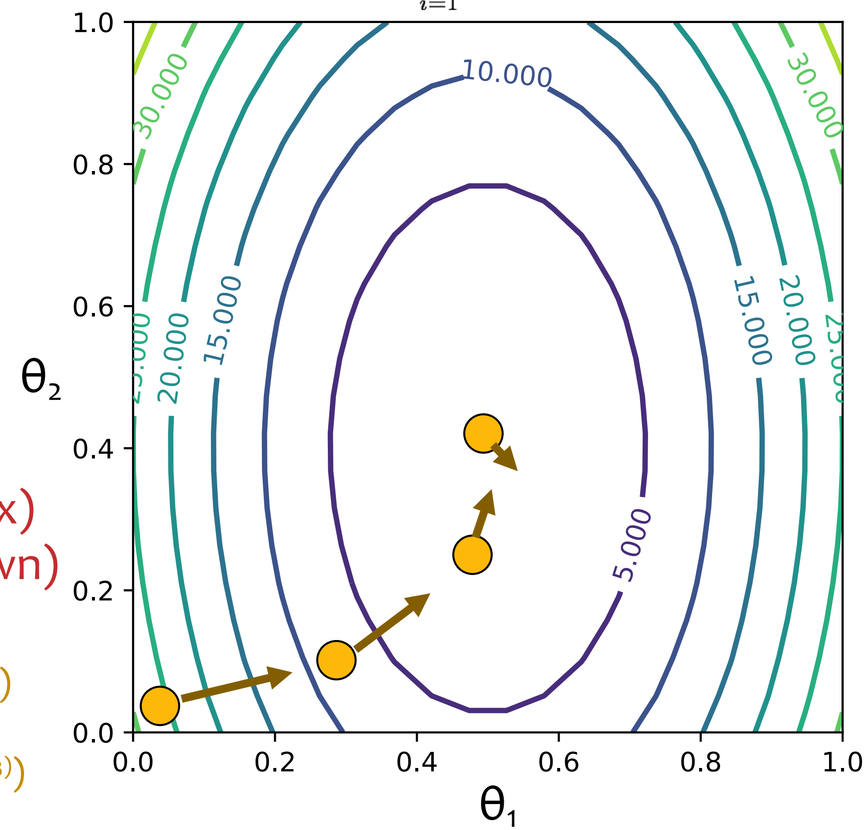


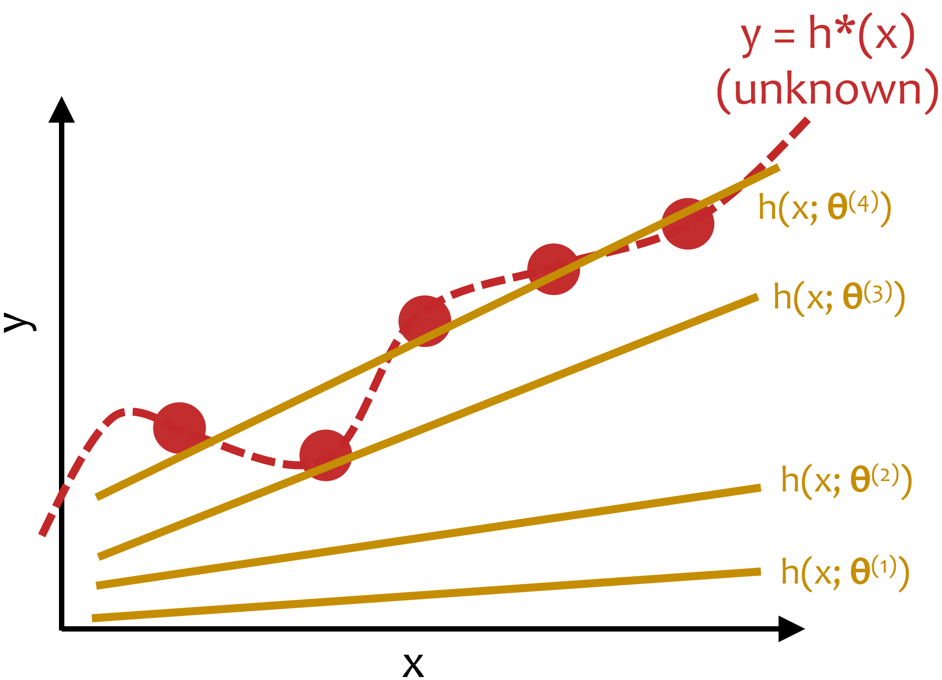
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Linear Regression by Gradient Desc.



$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$





t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Optimization for Linear Regression

Chalkboard

– Computing the gradient for Linear Regression

– Gradient Descent for Linear Regression

GD for Linear Regression

Gradient Descent for Linear Regression repeatedly takes steps opposite the gradient of the objective function

Algorithm 1 GD for Linear Regression

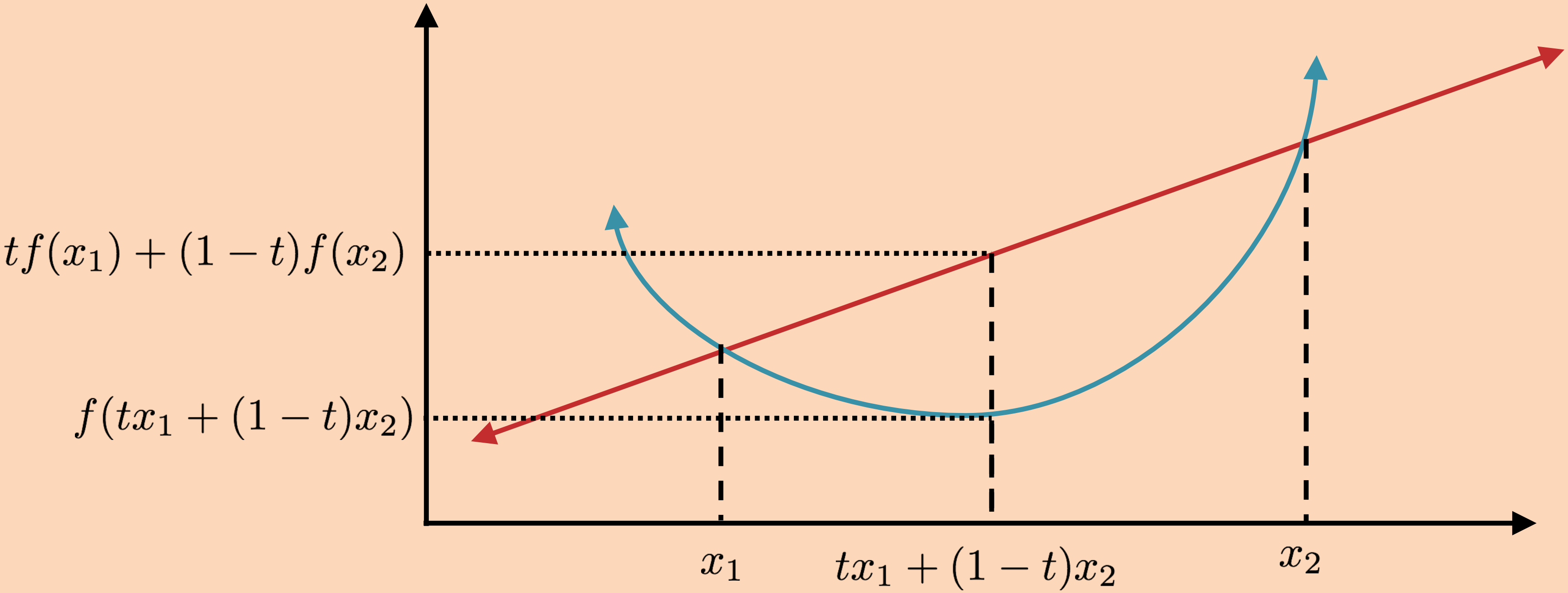
1: **procedure** GDLR($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
3: $\quad$ **while** not converged **do**
4: $\quad\quad \mathbf{g} \leftarrow \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$ ▷ Compute gradient
5: $\quad\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
6: $\quad$ **return** $\boldsymbol{\theta}$

CONVEXITY

Convexity

Function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ is **convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$$

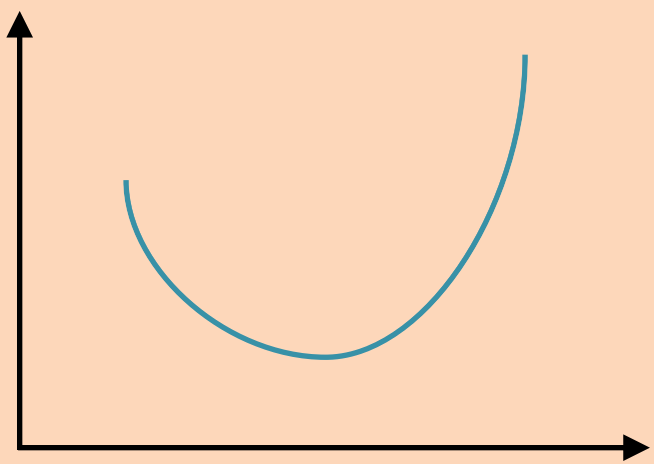


Convexity

Suppose we have a function $f(x) : \mathcal{X} \rightarrow \mathcal{Y}$.

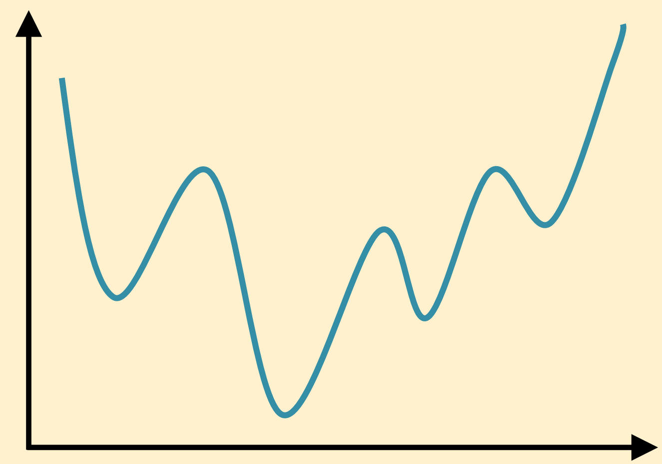
- The value x^* is a **global minimum** of f iff $f(x^*) \leq f(x), \forall x \in \mathcal{X}$.
- The value x^* is a **local minimum** of f iff $\exists \epsilon$ s.t. $f(x^*) \leq f(x), \forall x \in [x^* - \epsilon, x^* + \epsilon]$.

Convex Function



- Each **local minimum** is a **global minimum**

Nonconvex Function

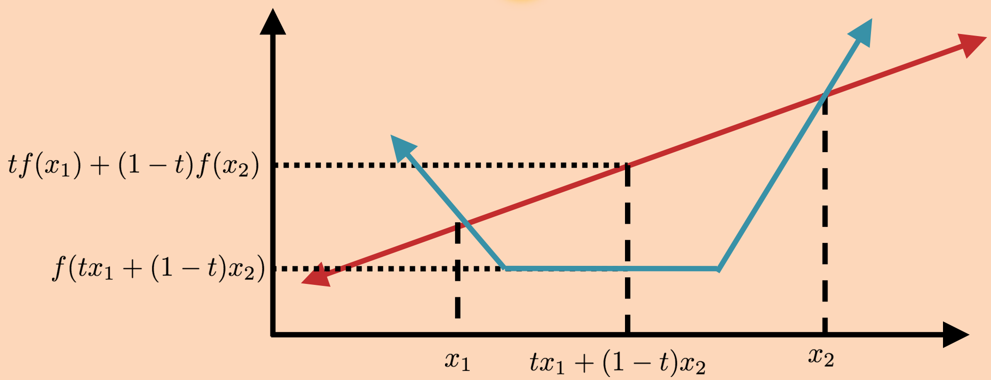


- A *nonconvex* function is **not convex**
- Each **local minimum** is **not** necessarily a **global minimum**

Convexity

Function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ is **convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

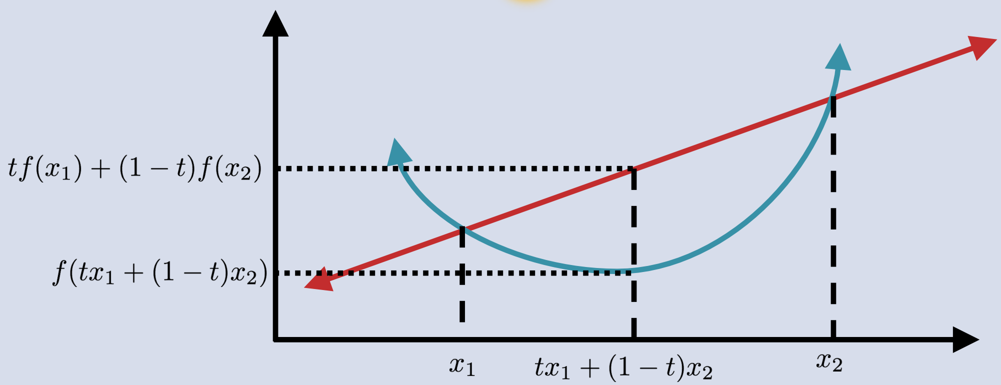
$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq t f(\mathbf{x}_1) + (1-t) f(\mathbf{x}_2)$$



Each **local minimum** of a **convex** function is also a **global minimum.**

Function $f : \mathbb{R}^M \rightarrow \mathbb{R}$ is **strictly convex**
if $\forall\ \mathbf{x}_1 \in \mathbb{R}^M, \mathbf{x}_2 \in \mathbb{R}^M, 0 \leq t \leq 1$:

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) < t f(\mathbf{x}_1) + (1-t) f(\mathbf{x}_2)$$



A **strictly convex** function has a **unique global minimum.**

CONVEXITY AND LINEAR REGRESSION

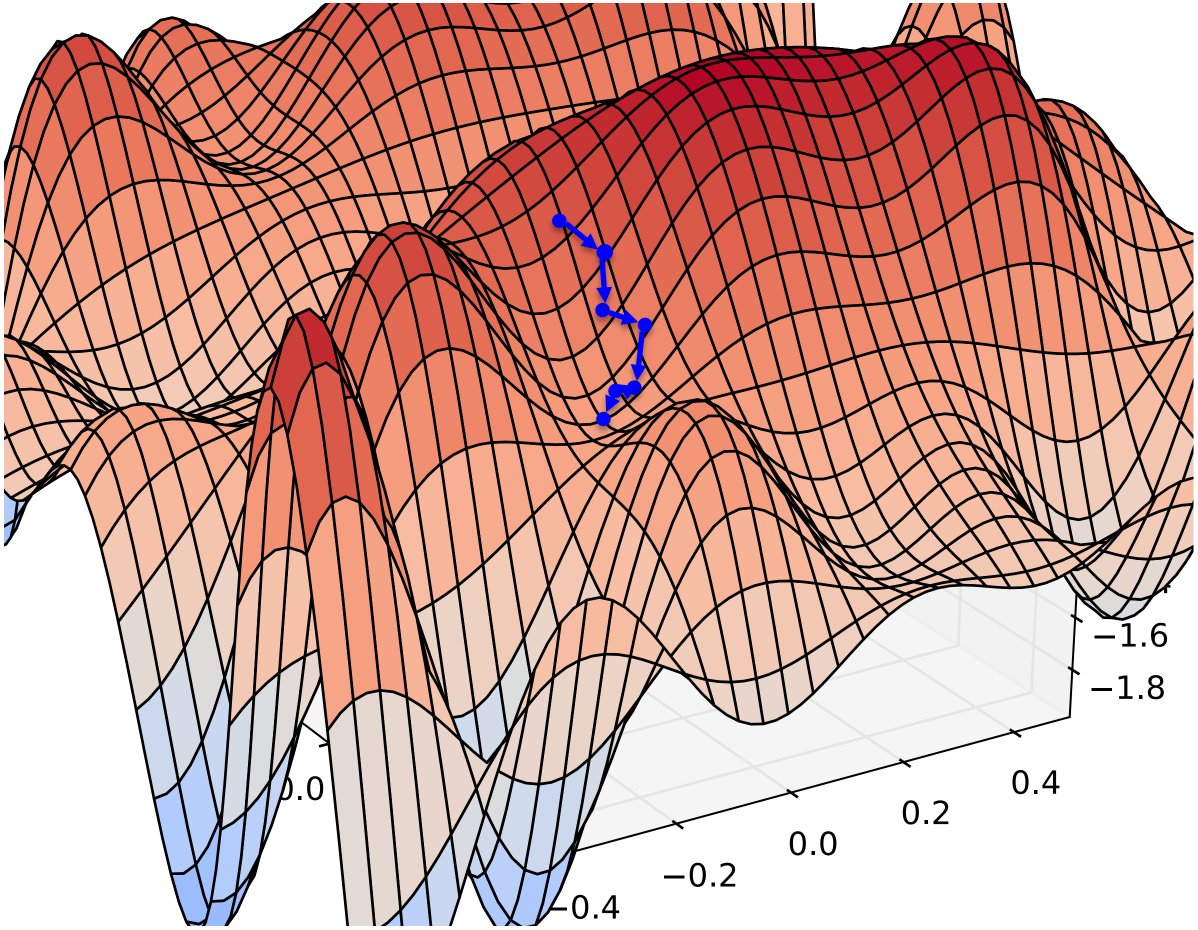
Convexity and Linear Regression

The **Mean Squared Error** function, which we minimize for learning the parameters of Linear Regression, **is convex**!

…but in the general case it is **not strictly convex**.

Gradient Descent & Convexity

- Gradient descent is a **local optimization algorithm**
- If the function is **nonconvex**, it will find a local minimum, not necessarily a global minimum
- If the function is **convex**, it will find a global minimum



Regression Loss Functions

In-Class Exercise:

Which of the following could be used as loss functions for training a linear regression model?

Select all that apply.

A. $\ell(\hat{y}, y) = ||\hat{y} - y||_2$

B. $\ell(\hat{y}, y) = |\hat{y} - y|$

C. $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

D. $\ell(\hat{y}, y) = \frac{1}{4}(\hat{y} - y)^4$

E. $\ell(\hat{y}, y) = \begin{cases} \frac{1}{2}(\hat{y} - y)^2 & \text{if } |\hat{y} - y| \leq \delta \\ \delta|\hat{y} - y| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$

F. $\ell(\hat{y}, y) = \log(\cosh(\hat{y} - y))$

Question 1

A

B

C

D

E

F

OPTIMIZATION METHOD #2: CLOSED FORM SOLUTION

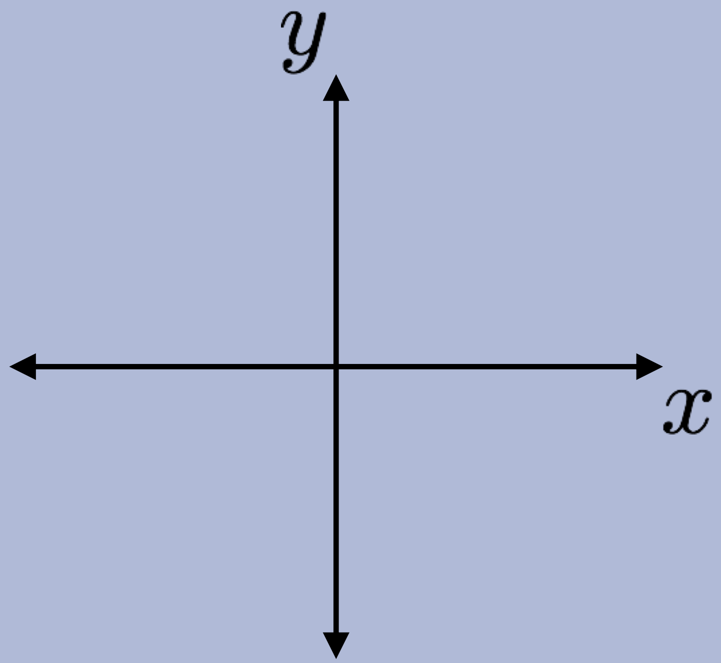
Calculus and Optimization

In-Class Exercise

Plot three functions:

1. $f(x) = x^3 - x$
2. $f'(x) = \frac{\partial y}{\partial x}$
3. $f''(x) = \frac{\partial^2 y}{\partial x^2}$

Answer Here:



Optimization: Closed form solutions

Chalkboard

- Zero Derivatives
- Example: 1-D function
- Example: higher dimensions

CLOSED FORM SOLUTION FOR LINEAR REGRESSION

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^N e_i^2 \\ &= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2 \end{aligned}$$

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

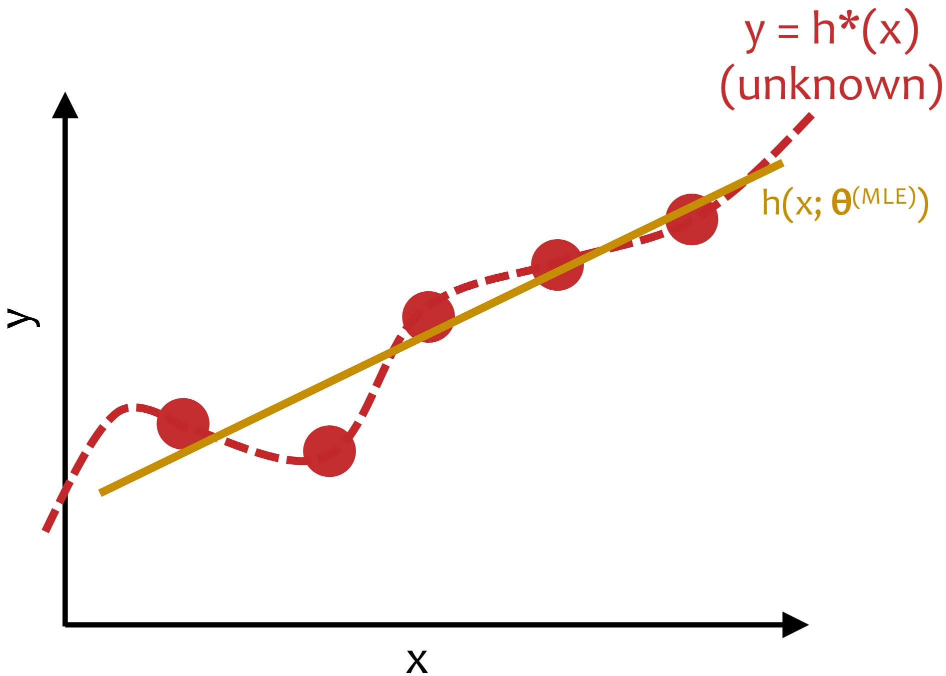
5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T\mathbf{x}$$

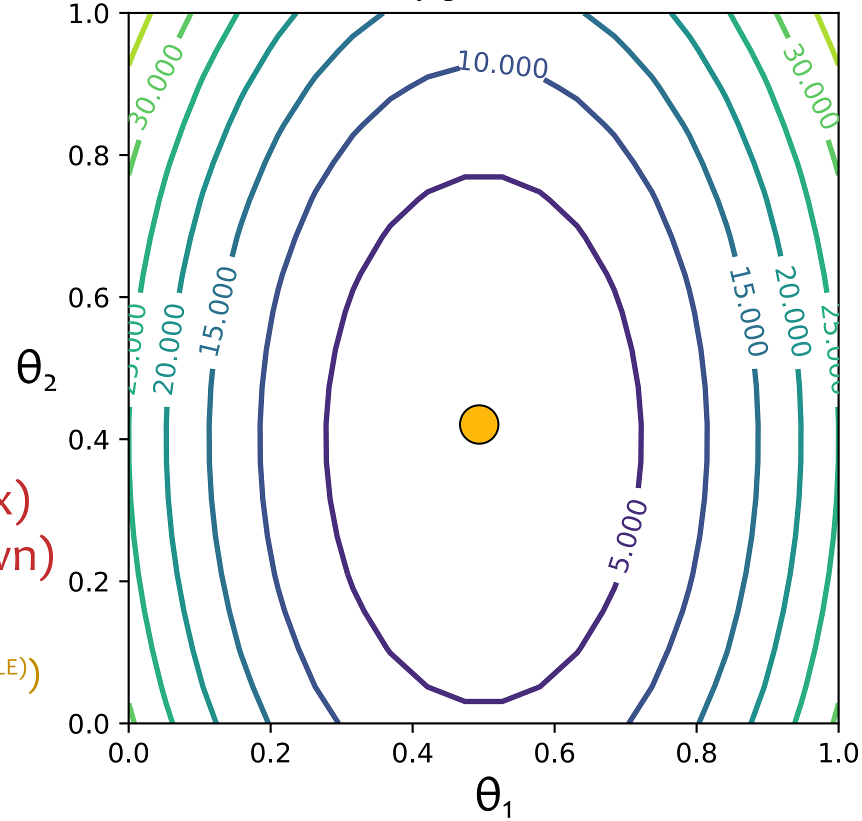
Linear Regression: Closed Form

Optimization Method #2: Closed Form

1. Evaluate
   $$\boldsymbol{\theta}^{\text{MLE}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$
2. Return $\boldsymbol{\theta}^{\text{MLE}}$



$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$



t	θ_1	θ_2	$J(\theta_1, \theta_2)$
MLE	0.59	0.43	0.2

Optimization for Linear Regression

Chalkboard

– Closed-form (Normal Equations)

COMPUTATIONAL COMPLEXITY

Computational Complexity of OLS

To solve the Ordinary Least Squares problem we compute:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y})$$

The resulting shape of the matrices:

$$\underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{X}}_{N\times M})^{-1}}_{M\times M}\ \underbrace{(\underbrace{\mathbf{X}^T}_{M\times N}\ \underbrace{\mathbf{Y}}_{N\times 1})}_{M\times 1}$$

Background: Matrix Multiplication Given matrices $\mathbf{A}$ and $\mathbf{B}$

- If $\mathbf{A}$ is $q \times r$ and $\mathbf{B}$ is $r \times s$, computing $\mathbf{AB}$ takes $O(qrs)$
- If $\mathbf{A}$ and $\mathbf{B}$ are $q \times q$, computing $\mathbf{AB}$ takes $O(q^{2.373})$
- If $\mathbf{A}$ is $q \times q$, computing A^{-1} takes $O(q^{2.373})$.

Computational Complexity of OLS:

$\mathbf{X}^T\mathbf{X}$	$O(M^2N)$
$(\quad)^{-1}$	$O(M^{2.373})$
$\mathbf{X}^T\mathbf{Y}$	$O(MN)$
$(\quad)^{-1}(\quad)$	$O(M^2)$
total	$O(M^2N + M^{2.373})$

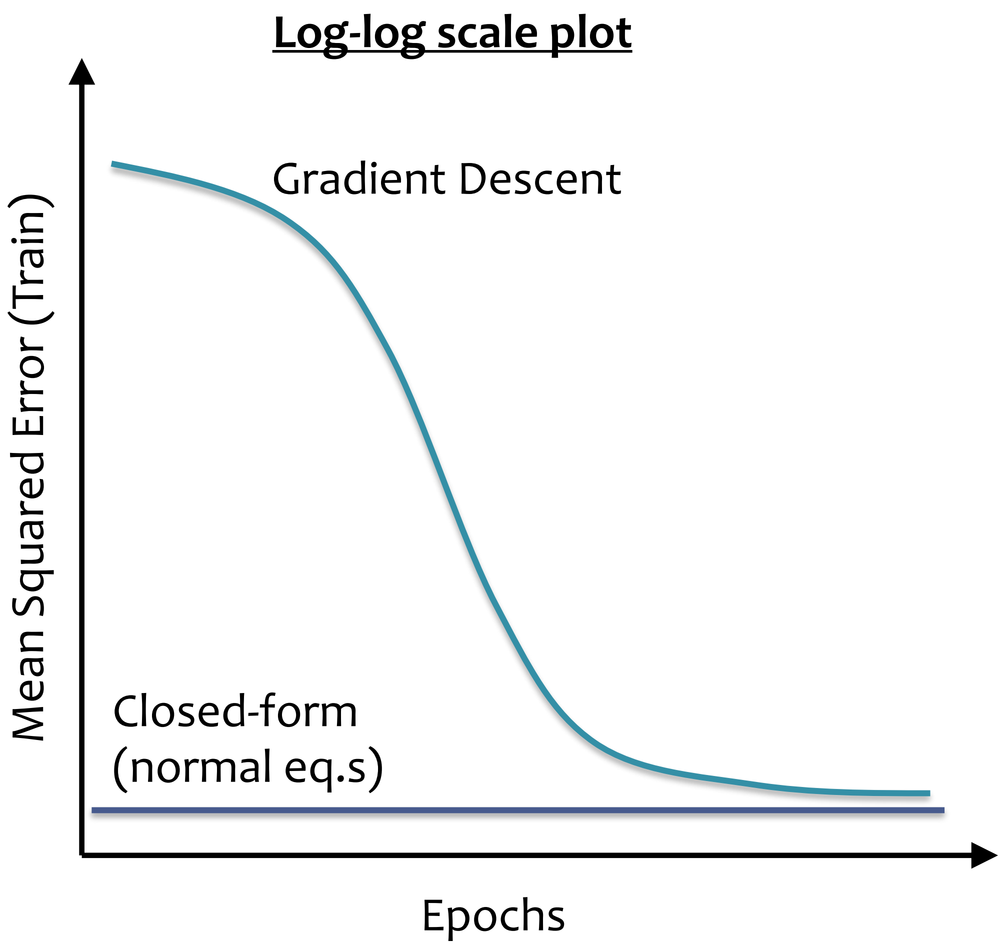
Linear in # of examples, N
Polynomial in # of features, M

Gradient Descent

Cases to consider gradient descent:

1. What if we **can not** find a closed-form solution?
2. What if we **can**, but it's inefficient to compute?
3. What if we **can**, but it's numerically unstable to compute?

Empirical Convergence



- *Def*: an **epoch** is a single pass through the training data

1. For GD, only **one update** per epoch
2. For SGD, ***N* updates** per epoch
   N = (# train examples)

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

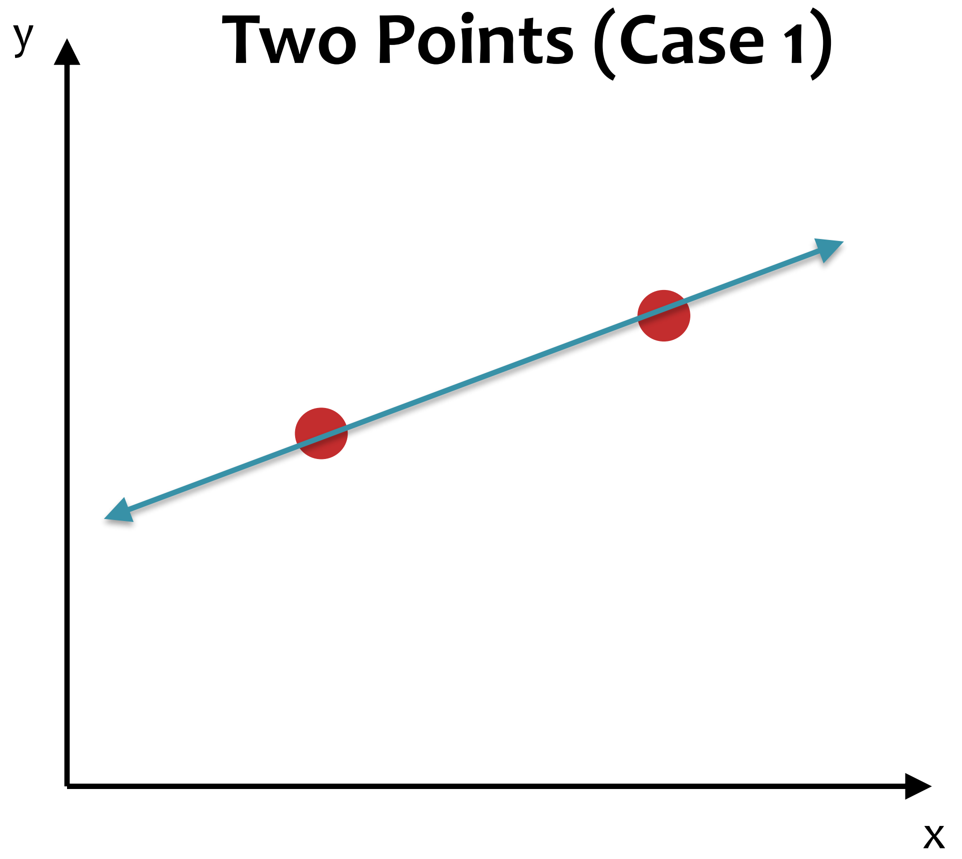
LINEAR REGRESSION: SOLUTION UNIQUENESS

Linear Regression: Uniqueness

Question:

Consider a 1D linear regression model trained to minimize MSE.

How many solutions (i.e. sets of parameters w,b) are there for the given dataset?

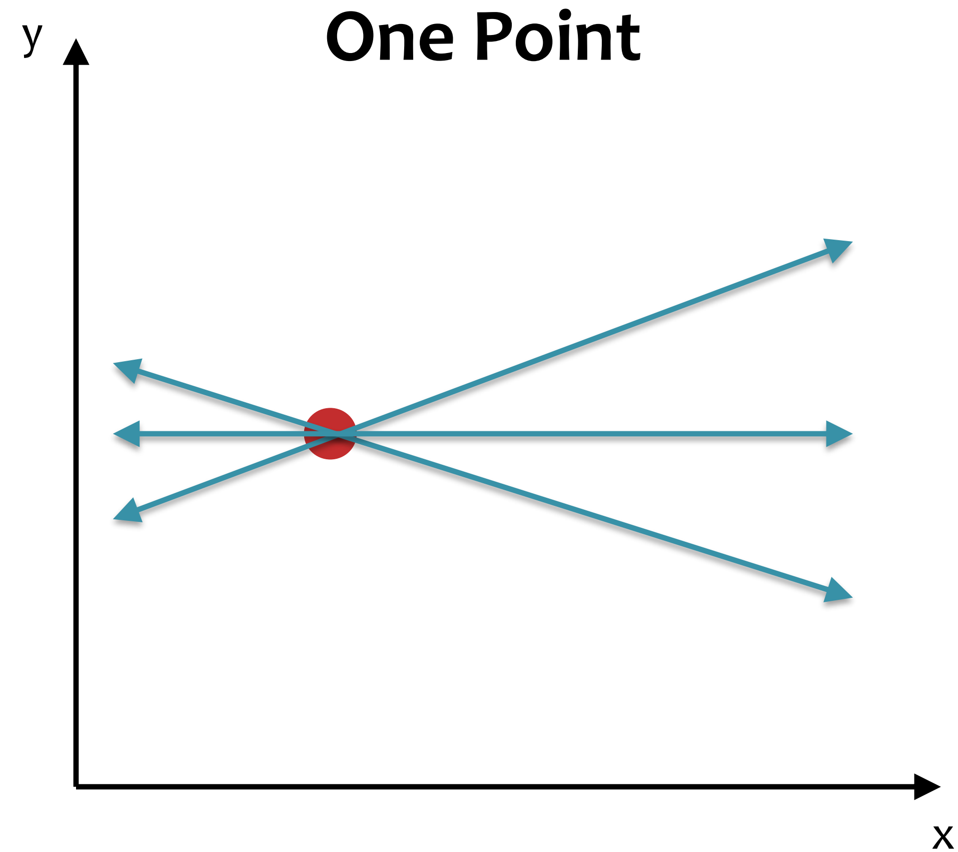


Linear Regression: Uniqueness

Question:

Consider a 1D linear regression model trained to minimize MSE.

How many solutions (i.e. sets of parameters w,b) are there for the given dataset?

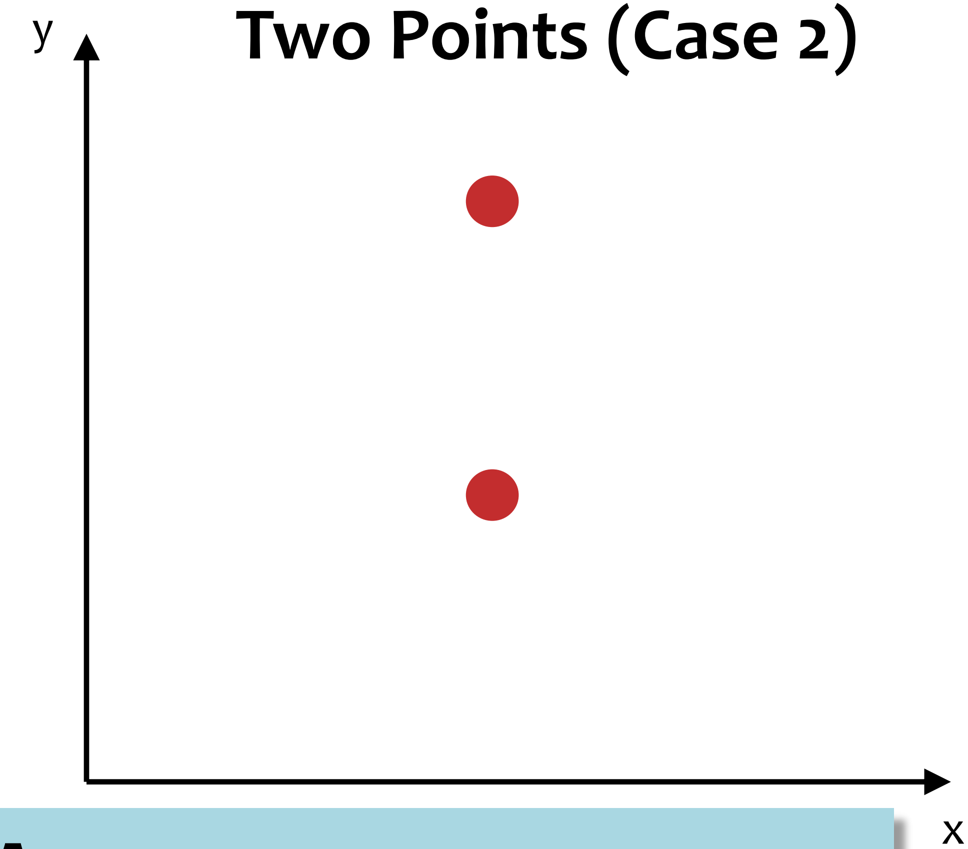


Linear Regression: Uniqueness

Question:

Consider a 1D linear regression model trained to minimize MSE.

How many solutions (i.e. sets of parameters w,b) are there for the given dataset?



Answer:

A: 0 B: 1 C: 2 D: +∞

Question 2

A

B

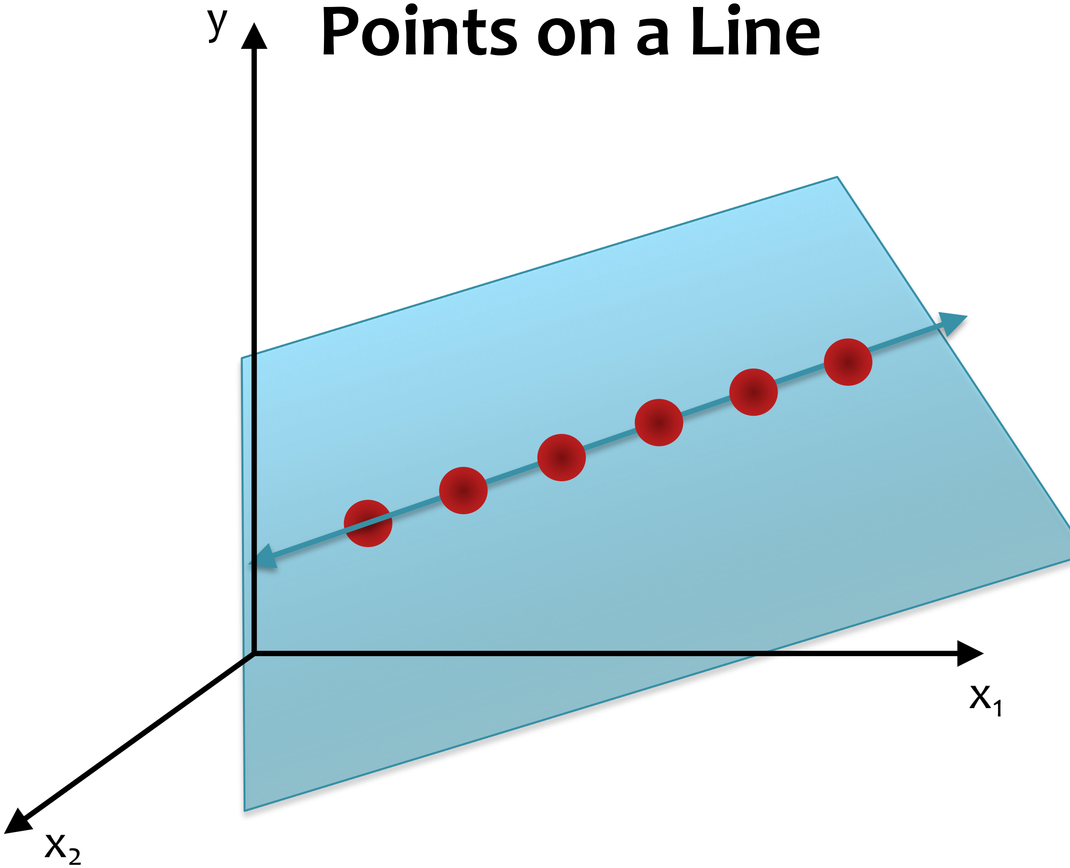
C

D

Linear Regression: Uniqueness

Question:

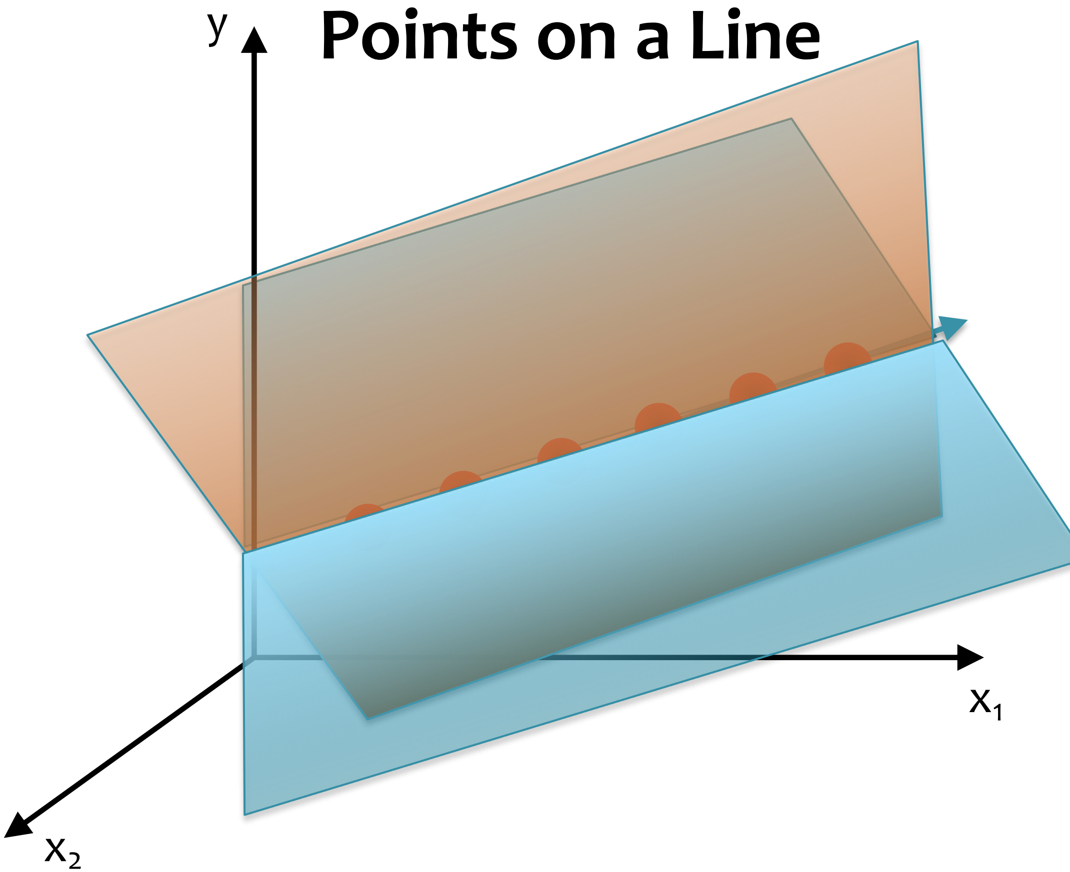
- Consider a **2D** linear regression model trained to minimize MSE
- How many solutions (i.e. sets of parameters w_1, w_2, b) are there for the given dataset?



Linear Regression: Uniqueness

Question:

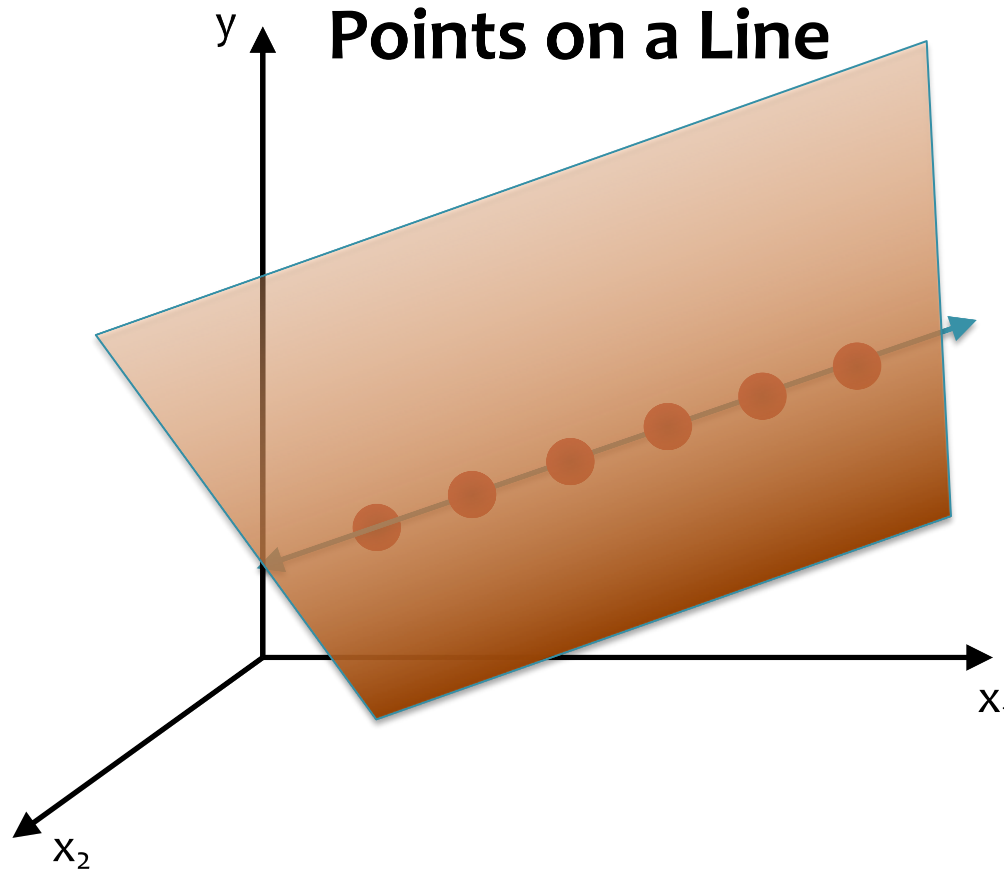
- Consider a **2D** linear regression model trained to minimize MSE
- How many solutions (i.e. sets of parameters w_1, w_2, b) are there for the given dataset?



Linear Regression: Uniqueness

Question:

- Consider a **2D** linear regression model trained to minimize MSE
- How many solutions (i.e. sets of parameters w_1, w_2, b) are there for the given dataset?



Linear Regression: Uniqueness

To solve the Ordinary Least Squares problem we compute:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T\mathbf{x}^{(i)}))^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y})$$

These geometric intuitions align with the linear algebraic intuitions we can derive from the normal equations.

1. If $(\mathbf{X}^T\mathbf{X})$ **is invertible,** then there is exactly one solution.
2. If $(\mathbf{X}^T\mathbf{X})$ **is not invertible,** then there are either no solutions or infinitely many solutions.

Linear Regression: Uniqueness

To solve the Ordinary Least Squares problem we compute:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}(y^{(i)} - (\boldsymbol{\theta}^T \mathbf{x}^{(i)}))^2$$
$$= (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y})$$

These geometric intuitions align with the linear algebraic intuitions we can derive from the normal equations.

1. If $(\mathbf{X}^T\mathbf{X})$ **is invertible,** then there is exactly one solution.
2. If $(\mathbf{X}^T\mathbf{X})$ **is not inv**
   no solutions or inf

Invertability of $(\mathbf{X}^T\mathbf{X})$ is equivalent to X being **full rank.** That is, there is **no feature that is a linear combination of the other features.**

Solving Linear Regression

Question:

True or False: If Mean Squared Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - h(\mathbf{x}^{(i)}))^2$) has a unique minimizer (i.e. argmin), then Mean Absolute Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - h(\mathbf{x}^{(i)})|$) must also have a unique minimizer.

Answer:

Question 3

A

B

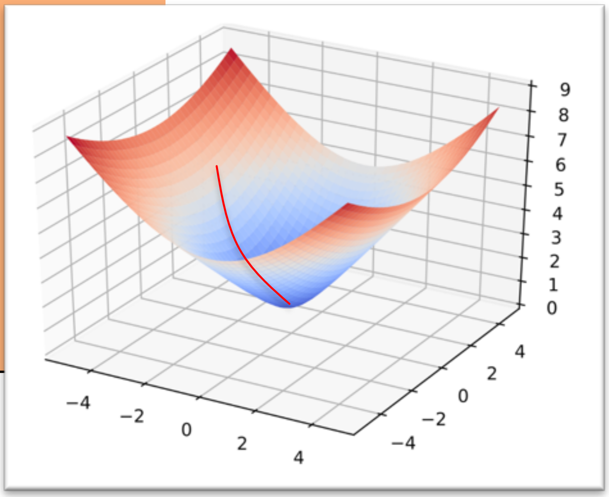
C

OPTIMIZATION METHOD #3: STOCHASTIC GRADIENT DESCENT

Gradient Descent

Algorithm 1 Gradient Descent

procedure GD($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$

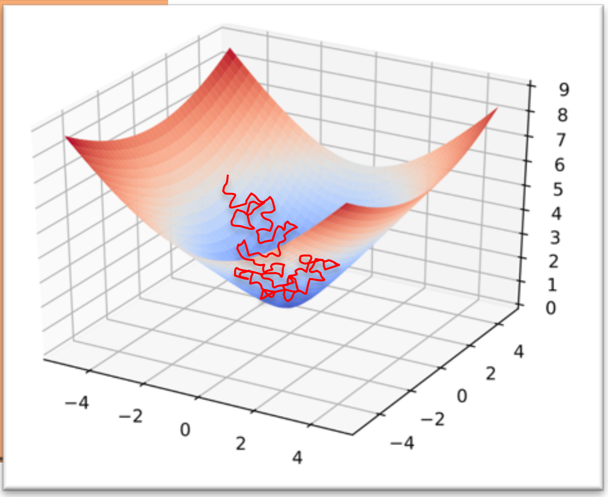


M

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    $i \sim \text{Uniform}(\{1, 2, \ldots, N\})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$



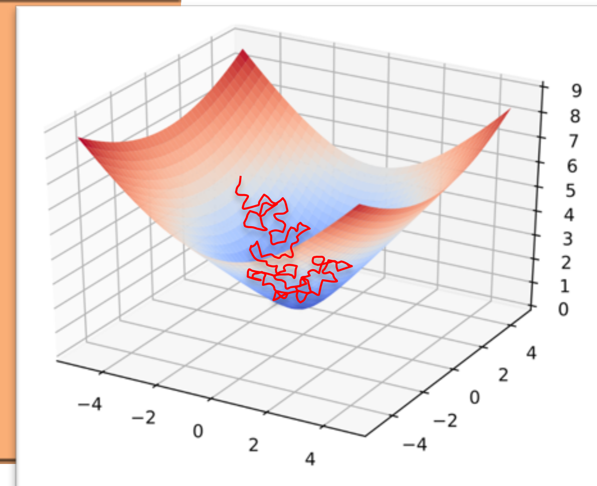
Per-example objective: $J^{(i)}(\boldsymbol{\theta})$

Original objective: $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$

Stochastic Gradient Descent (SGD)

Algorithm 2 Stochastic Gradient Descent (SGD)

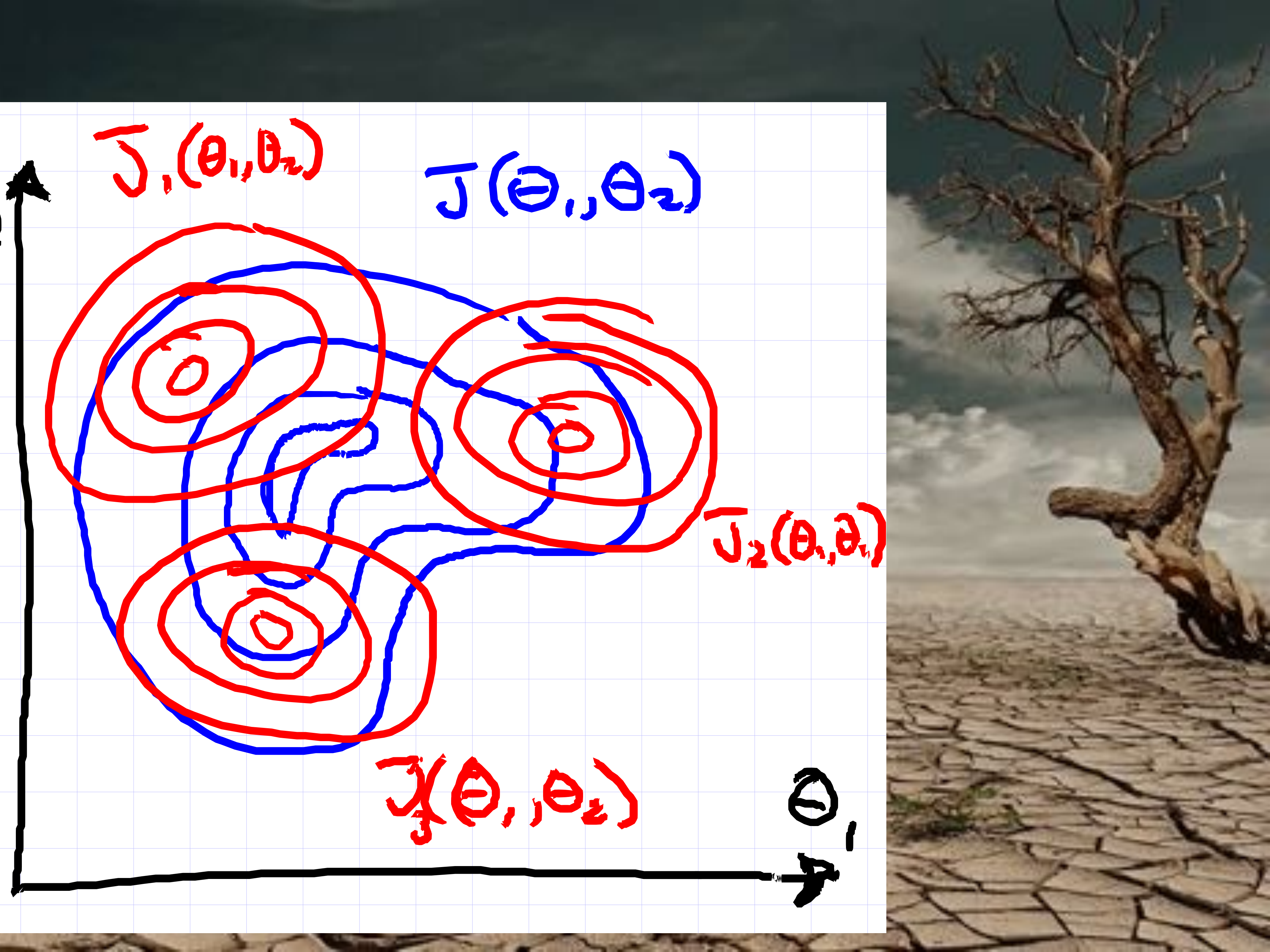
```
procedure SGD(𝒟, θ^(0))
    θ ← θ^(0)
    while not converged do
        for i ∈ shuffle({1, 2, ..., N}) do
            θ ← θ − γ∇_θ J^(i)(θ)
    return θ
```



In practice, it is common to implement SGD using sampling **without** replacement (i.e. shuffle({1,2,… N}), even though most of the theory is for sampling **with** replacement (i.e. Uniform({1,2,… N}).

Per-example objective: $J^{(i)}(\boldsymbol{\theta})$

Original objective: $J(\boldsymbol{\theta}) = \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$


$J_1(\theta_1,\theta_2)$
$J(\theta_1,\theta_2)$
$J_2(\theta_1,\theta_1)$
$J_3(\theta_1,\theta_2)$
θ_1

Expectations of Gradients

$$\frac{dJ(\vec{\theta})}{d\theta_j} = \frac{1}{N}\sum_{i=1}^{N} \frac{d}{d\theta_j}\left(J_i(\vec{\theta})\right)$$

$$\nabla J(\vec{\theta}) = \begin{bmatrix} \vdots \\ \text{(}j^{th}\text{)} \\ \vdots \end{bmatrix} = \frac{1}{N}\sum_{i=1}^{N} \nabla J_i(\vec{\theta})$$

Recall: for any discrete r.v. X

$$E_X[f(X)] \triangleq \sum_x P(X=x) f(x)$$

Q: What is the expected value of a randomly chosen $\nabla J_i(\theta)$?

Let $I \sim \text{Uniform}(\{1, \ldots, N\})$

$\Rightarrow P(I=i) = \frac{1}{N}$ if $i \in \{1 \ldots N\}$

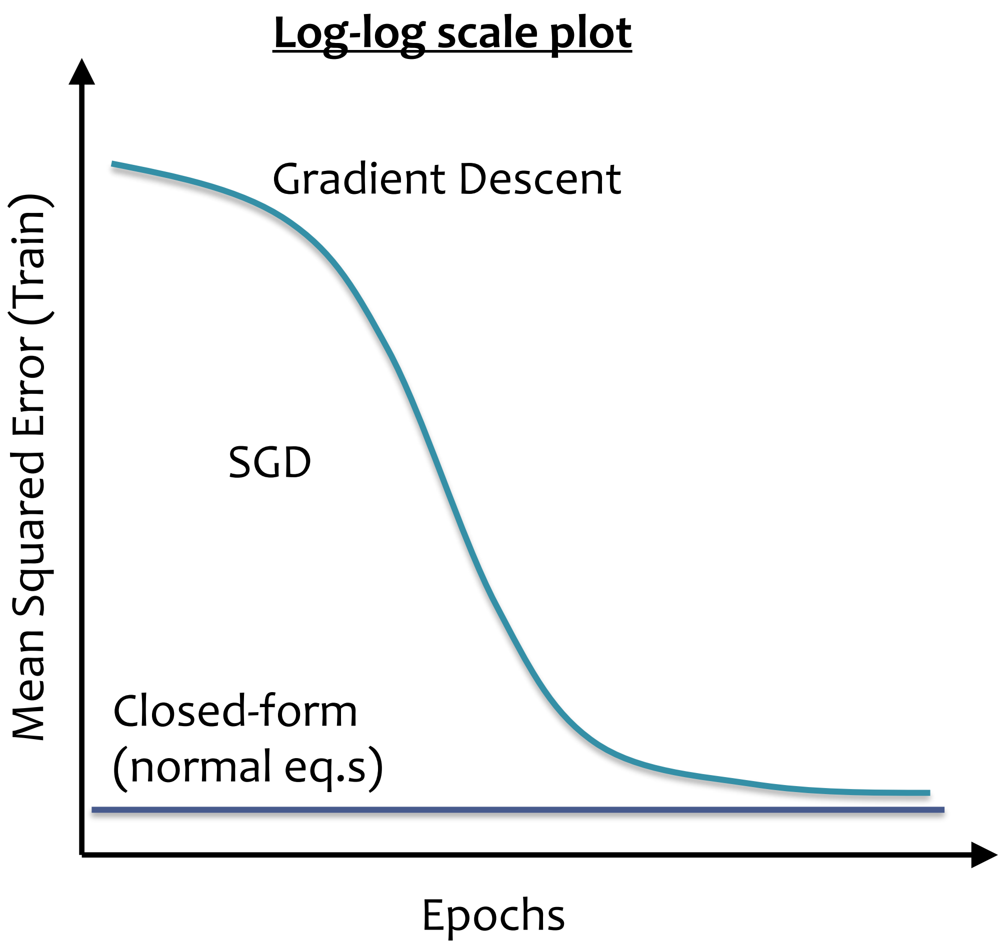
$$E_I[\nabla J_I(\vec{\theta})] = \sum_{i=1}^{N} P(I=i)\nabla J_i(\vec{\theta})$$

$$= \frac{1}{N}\sum_{i=1}^{N} \nabla J_i(\vec{\theta})$$

$$= \nabla J(\vec{\theta})$$

LINEAR REGRESSION: PRACTICALITIES

Empirical Convergence



- *Def*: an **epoch** is a single pass through the training data

1. For GD, only **one update** per epoch
2. For SGD, ***N* updates** per epoch
   N = (# train examples)

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

Convergence of Optimizers

Convergence Analysis:

Def: convergence is when $J(\vec{\theta}) - J(\vec{\theta}^*) < \epsilon$ ← true unknown min

(not covered → Newton's Method)

Methods	Steps to Converge	Computation per iteration
Newton's Method	$O(\ln \ln 1/\epsilon)$	$\nabla J(\theta)$ $\nabla^2 J(\theta)$ ← $O(NM^2)$
GD	$O(\ln 1/\epsilon)$	$\nabla J(\theta)$ ← $O(NM)$
SGD	$O(1/\epsilon)$	$\nabla J_i(\theta)$ ← $O(M)$

"almost sure" convergence (SGD)

lots of caveats and conditions

way less computation (SGD)

Takeaway: SGD has much slower asymptotic convergence, but is often faster in practice.

SGD FOR LINEAR REGRESSION

Linear Regression as Function Approximation

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^N e_i^2$$
$$= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2$$
$$= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$

4. Solve the unconstrained optimization problem via favorite method:

- *gradient descent*
- *closed form*
- *stochastic gradient descent*
- ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

Gradient Calculation for Linear Regression

Derivative of $J^{(i)}(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) &= \frac{d}{d\theta_k} \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= \frac{1}{2} \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} \left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)} \right) \\
&= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

Derivative of $J(\boldsymbol{\theta})$:

$$\begin{aligned}
\frac{d}{d\theta_k} J(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}
\end{aligned}$$

Gradient of $J^{(i)}(\boldsymbol{\theta})$ [used by SGD]

$$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J^{(i)}(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J^{(i)}(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J^{(i)}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Gradient of $J(\boldsymbol{\theta})$ [used by Gradient Descent]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^N (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ \sum_{i=1}^N (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ \sum_{i=1}^N (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

SGD for Linear Regression

SGD applied to Linear Regression is called the "Least Mean Squares" algorithm

Algorithm 1 Least Mean Squares (LMS)

procedure LMS($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
    while not converged **do**
        for $i \in$ shuffle($\{1, 2, \dots, N\}$) **do**
            $\mathbf{g} \leftarrow (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
    return $\boldsymbol{\theta}$

Optimization Objectives

You should be able to…

- Apply gradient descent to optimize a function
- Apply stochastic gradient descent (SGD) to optimize a function
- Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
- Distinguish between convex, concave, and nonconvex functions
- Obtain the gradient (and Hessian) of a (twice) differentiable function

Linear Regression Objectives

You should be able to…

- Design k-NN Regression and Decision Tree Regression
- Implement learning for Linear Regression using three optimization techniques: (1) closed form, (2) gradient descent, (3) stochastic gradient descent
- Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed
- Distinguish the three sources of error identified by the bias-variance decomposition: bias, variance, and irreducible error.