10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Stochastic Gradient Descent

+

Probabilistic Learning

(Binary Logistic Regression)

Matt Gormley

Lecture 9

Mar. 01, 2021

Reminders

- Homework 3: KNN, Perceptron, Lin.Reg.
  - Out: Mon, Feb. 22
  - Due: Mon, Mar. 01 at 11:59pm
  - IMPORTANT: you may only use 2 grace days on Homework 3 (last possible moment to submit HW3: Wed, Mar. 03 at 11:59pm)
- Practice for Exam
  - Mock Exam 1
    - Wed, Mar. 03 at 7:00pm – 9:00pm
    - See @261 for participation point details
  - Practice Problems 1A (Gradescope)
  - Practice Problems 1B (PDF)
- Midterm Exam 1
  - Tue, Feb. 18, 7:00pm – 9:00pm

MIDTERM EXAM LOGISTICS

Midterm Exam

- **Time / Location**
  - **Time:** Saturday Exam
    Saturday, March 6, at 10:30am - 12:30pm EST
  - **Location:** We will contact you with additional details about how to join the appropriate Zoom meeting.
  - **Seats:** There will be **assigned Zoom rooms.** Please arrive online early.
  - Please watch Piazza carefully for announcements.
- **Logistics**
  - Covered material: Lecture 1 – Lecture 8
  - Format of questions:
    - Multiple choice
    - True / False (with justification)
    - Derivations
    - Short answers
    - Interpreting figures
    - Implementing algorithms on paper
  - ~~No electronic devices~~

Midterm Exam

- **How to Prepare**
  - Attend the midterm review lecture (right now!)
  - Participate in the Mock Exam
  - Review exam practice problems (we'll post them)
  - Review this year's homework problems
  - Consider whether you have achieved the "learning objectives" for each lecture / section

Midterm Exam

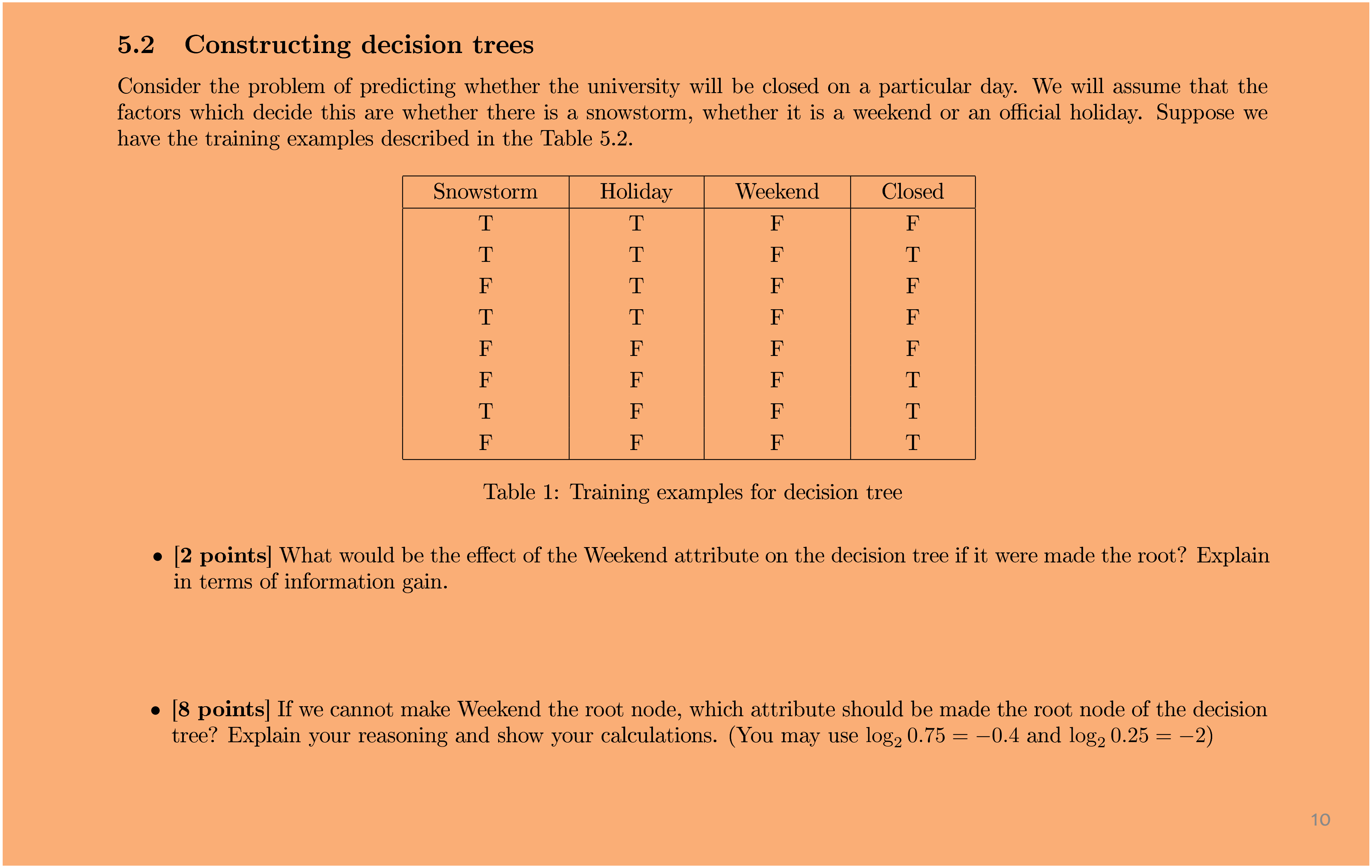
- **Advice (for during the exam)**
  - Solve the easy problems first (e.g. multiple choice before derivations)
    - if a problem seems extremely complicated you're likely missing something
  - Don't leave any answer blank!
  - If you make an assumption, write it down
  - If you look at a question and don't know the answer:
    - we probably haven't told you the answer
    - but we've told you enough to work it out
    - imagine arguing for some answer and see if you like it

Topics for Midterm 1

- Foundations
  - Probability, Linear Algebra, Geometry, Calculus
  - Optimization
- Important Concepts
  - Overfitting
  - Experimental Design
- Classification
  - Decision Tree
  - KNN
  - Perceptron
- Regression
  - Linear Regression

SAMPLE QUESTIONS

Sample Questions



5.2 Constructing decision trees

Consider the problem of predicting whether the university will be closed on a particular day. We will assume that the factors which decide this are whether there is a snowstorm, whether it is a weekend or an official holiday. Suppose we have the training examples described in the Table 5.2.

Snowstorm	Holiday	Weekend	Closed
T	T	F	F
T	T	F	T
F	T	F	F
T	T	F	F
F	F	F	F
F	F	F	T
T	F	F	T
F	F	F	T

Table 1: Training examples for decision tree

- **[2 points]** What would be the effect of the Weekend attribute on the decision tree if it were made the root? Explain in terms of information gain.

- **[8 points]** If we cannot make Weekend the root node, which attribute should be made the root node of the decision tree? Explain your reasoning and show your calculations. (You may use $\log_2 0.75 = -0.4$ and $\log_2 0.25 = -2$)

Sample Questions

4 K-NN [12 pts]

Now we will apply K-Nearest Neighbors using Euclidean distance to a binary classification task. We assign the class of the test point to be the class of the majority of the k nearest neighbors. A point can be its own neighbor.

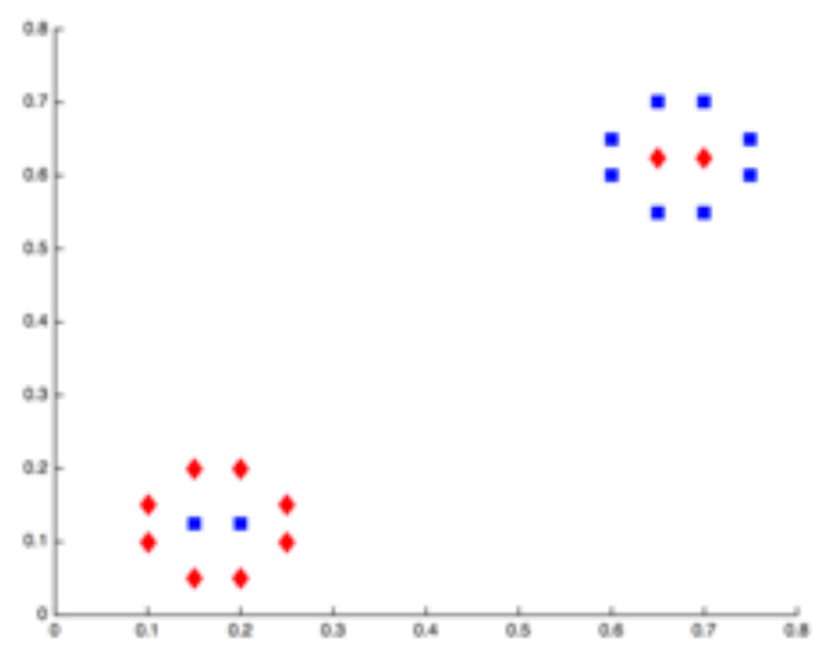
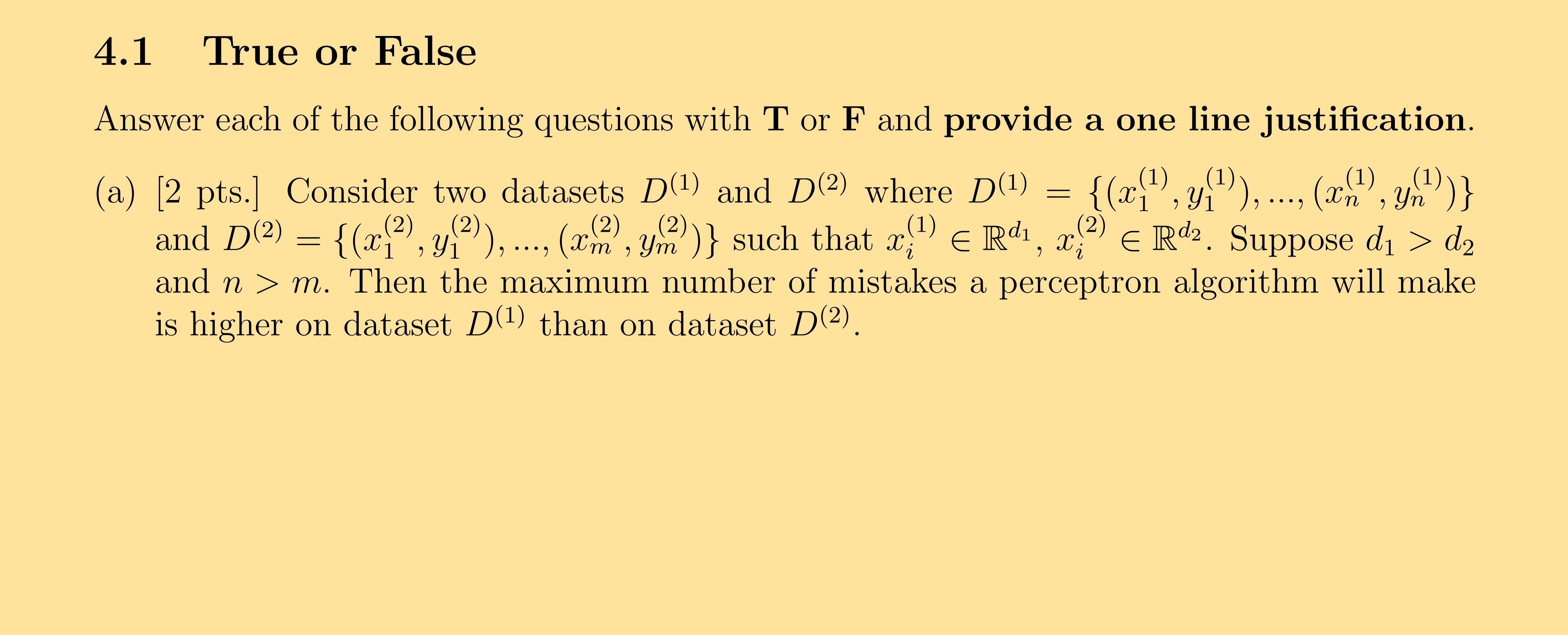


Figure 5

3. [**2 pts**] What value of k minimizes leave-one-out cross-validation error for the dataset shown in Figure 5? What is the resulting error?

Sample Questions



4.1 True or False

Answer each of the following questions with **T** or **F** and **provide a one line justification**.

(a) [2 pts.] Consider two datasets $D^{(1)}$ and $D^{(2)}$ where $D^{(1)} = \{(x_1^{(1)}, y_1^{(1)}), ..., (x_n^{(1)}, y_n^{(1)})\}$ and $D^{(2)} = \{(x_1^{(2)}, y_1^{(2)}), ..., (x_m^{(2)}, y_m^{(2)})\}$ such that $x_i^{(1)} \in \mathbb{R}^{d_1}$, $x_i^{(2)} \in \mathbb{R}^{d_2}$. Suppose $d_1 > d_2$ and $n > m$. Then the maximum number of mistakes a perceptron algorithm will make is higher on dataset $D^{(1)}$ than on dataset $D^{(2)}$.

Sample Questions

3.1 Linear regression

Consider the dataset S plotted in Fig. 1 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 3, indicate which regression line (relative to the original one) in Fig. 2 corresponds to the regression line for the new data set. Write your answers in the table below.

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line					

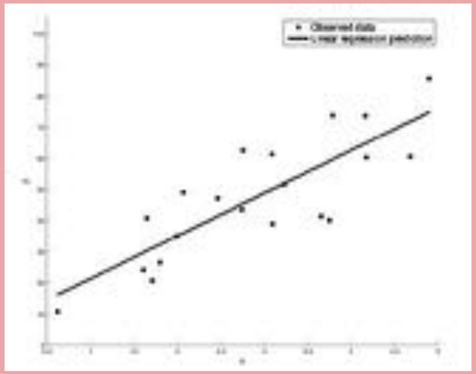
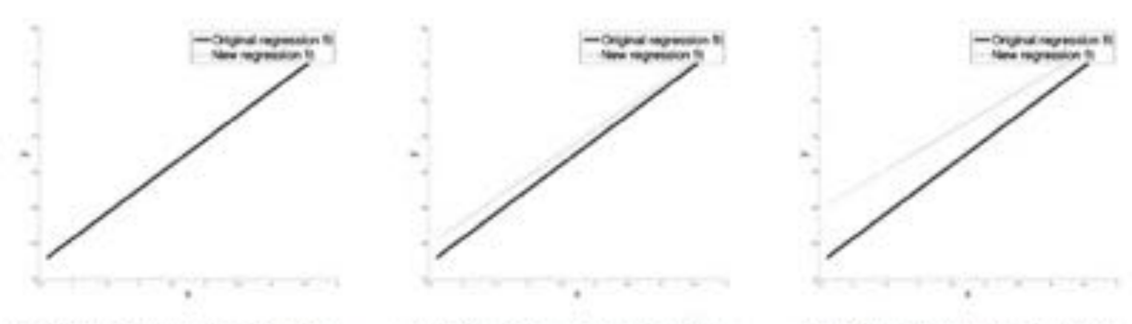


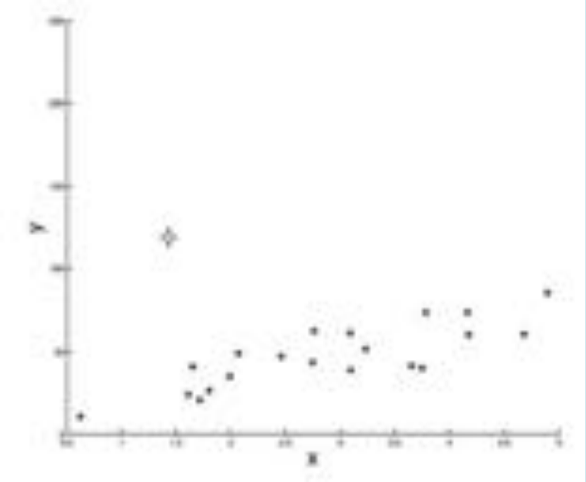
Figure 1: An observed data set and its associated regression line.



(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

Figure 2: New regression lines for altered data sets S^{new}.

Dataset



(a) Adding one outlier to the original data set.

Sample Questions

3.1 Linear regression

Consider the dataset S plotted in Fig. 1 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 3, indicate which regression line (relative to the original one) in Fig. 2 corresponds to the regression line for the new data set. Write your answers in the table below.

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line					

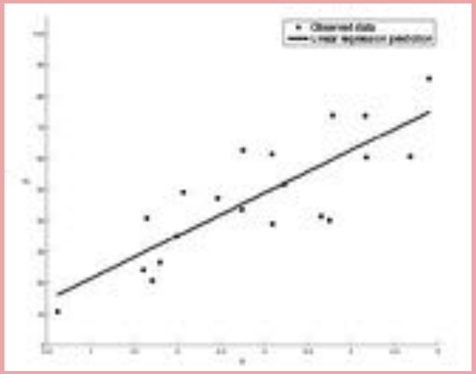
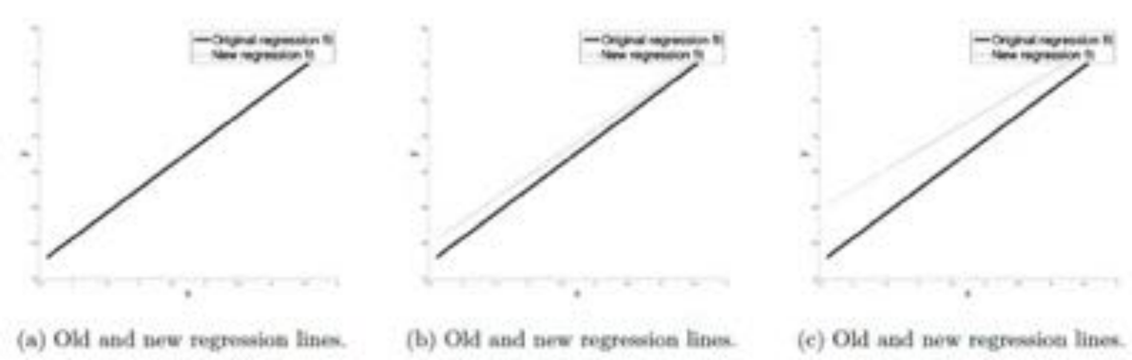


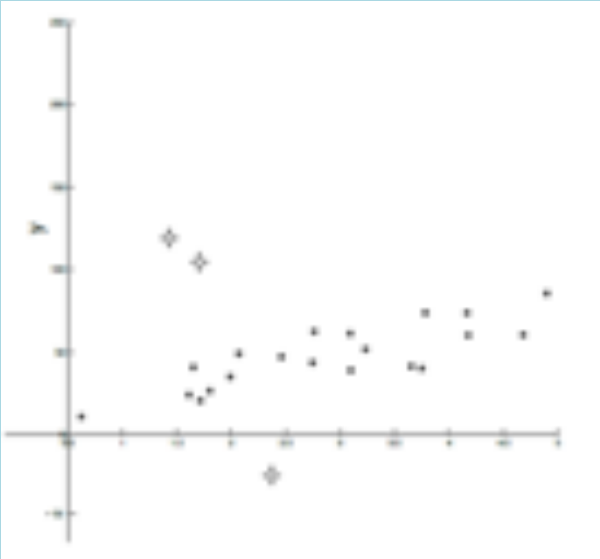
Figure 1: An observed data set and its associated regression line.



(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

Figure 2: New regression lines for altered data sets S^{new}.

Dataset



(c) Adding three outliers to the original data set. Two on one side and one on the other side.

Sample Questions

3.1 Linear regression

Consider the dataset S plotted in Fig. 1 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 3, indicate which regression line (relative to the original one) in Fig. 2 corresponds to the regression line for the new data set. Write your answers in the table below.

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line					

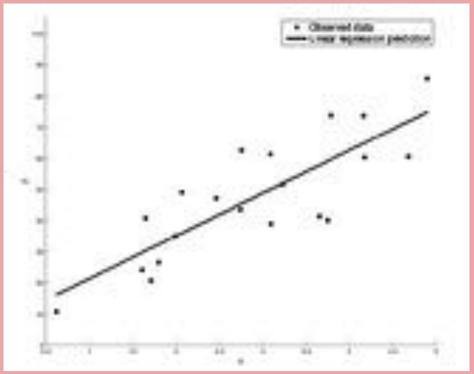
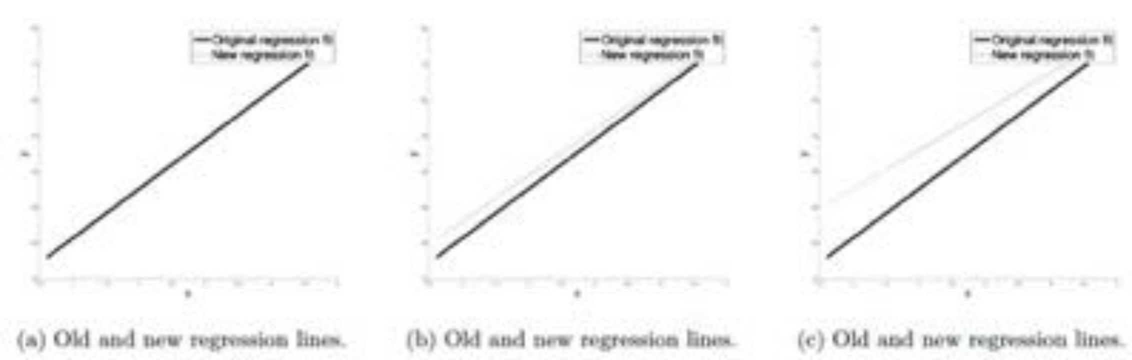


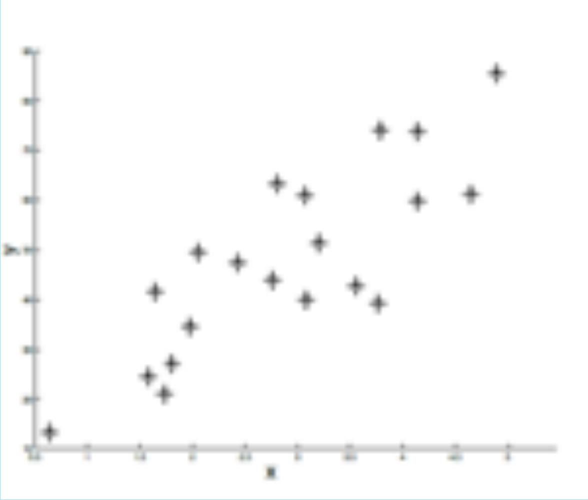
Figure 1: An observed data set and its associated regression line.



(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

Figure 2: New regression lines for altered data sets S^{new}.

Dataset



(d) Duplicating the original data set.

Sample Questions

3.1 Linear regression

Consider the dataset S plotted in Fig. 1 along with its associated regression line. For each of the altered data sets S^{new} plotted in Fig. 3, indicate which regression line (relative to the original one) in Fig. 2 corresponds to the regression line for the new data set. Write your answers in the table below.

Dataset	(a)	(b)	(c)	(d)	(e)
Regression line					

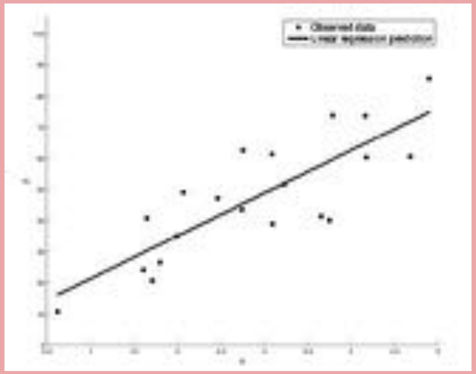
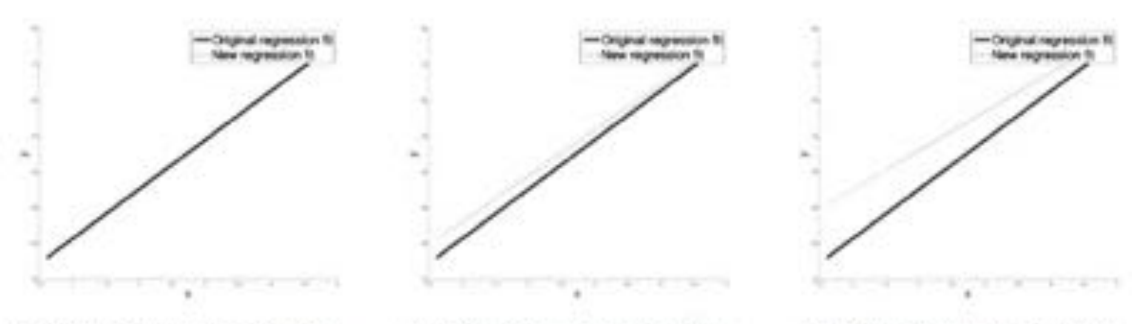


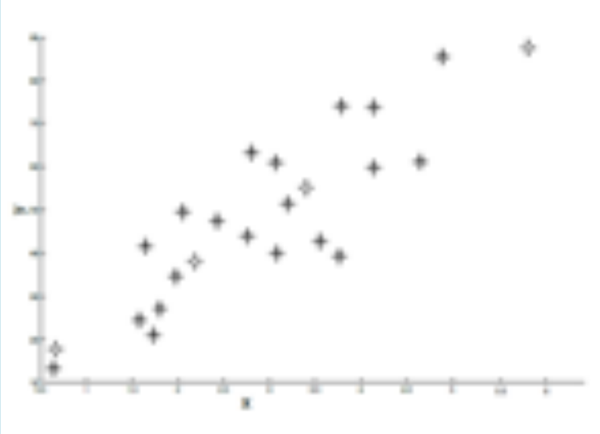
Figure 1: An observed data set and its associated regression line.



(a) Old and new regression lines. (b) Old and new regression lines. (c) Old and new regression lines.

Figure 2: New regression lines for altered data sets S^{new}.

Dataset



(e) Duplicating the original data set and adding four points that lie on the trajectory of the original regression line.

Q&A

Q&A

Q: Is there one recitation timeslot or two for this class?

A: Back to just one, i.e. Friday, same time as lecture.

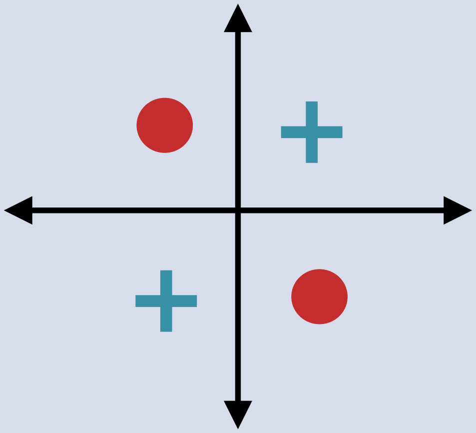
We tried hosting a Thursday evening recitation, but attendance remained around half a dozen students. So we are **not** hosting it anymore.

Q&A

Q: Why did we focus mostly on the Perceptron mistake bound for **linearly separable data;** isn't that an unrealistic setting?

A: Not at all! Even if your data isn't linearly separable to begin with, we can often add features to make it so.

x_1	x_2	y
+1	+1	+
+1	-1	-
-1	+1	-
-1	-1	+



Exercise: Add another feature to transform this nonlinearly separable data into linearly separable data.

CLOSED FORM SOLUTION FOR LINEAR REGRESSION

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	$O(M^2N + M^{2.373})$

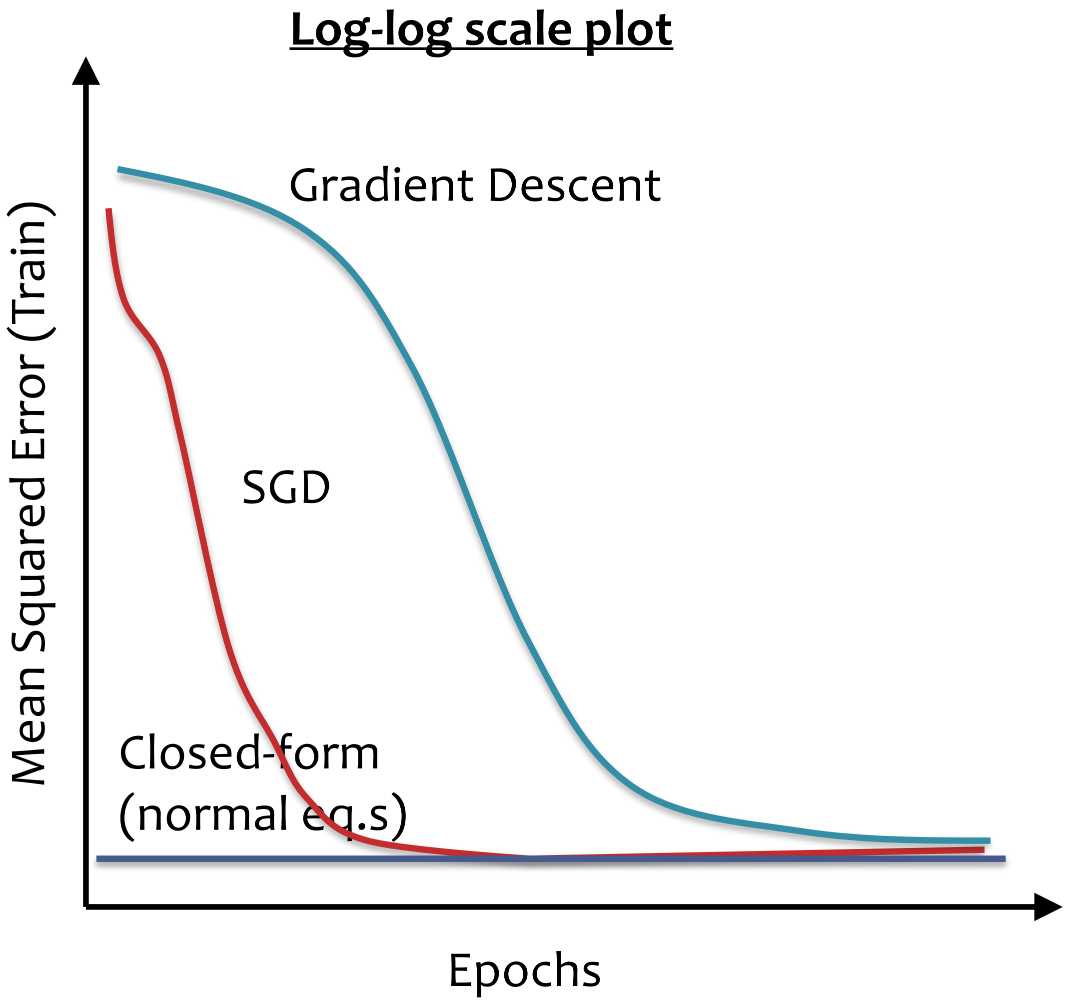
Linear in # of examples, N
Polynomial in # of features, M

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

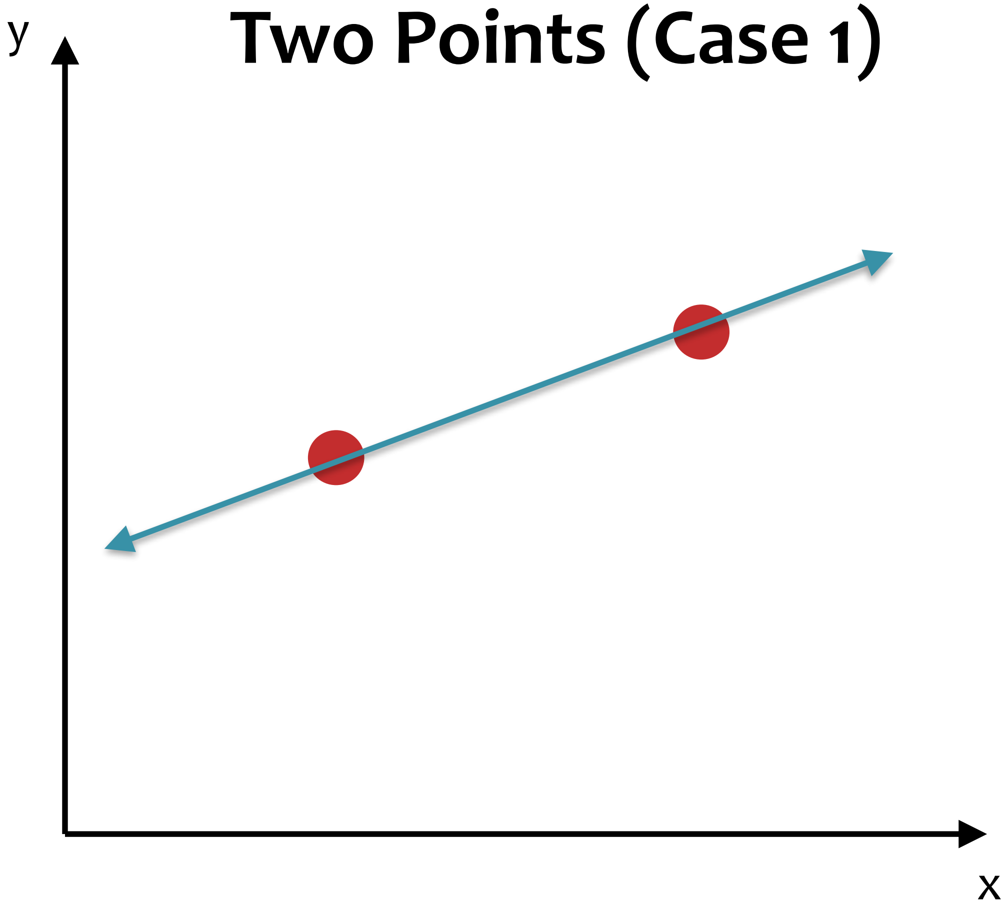
LINEAR REGRESSION: SOLUTION UNIQUENESS

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?

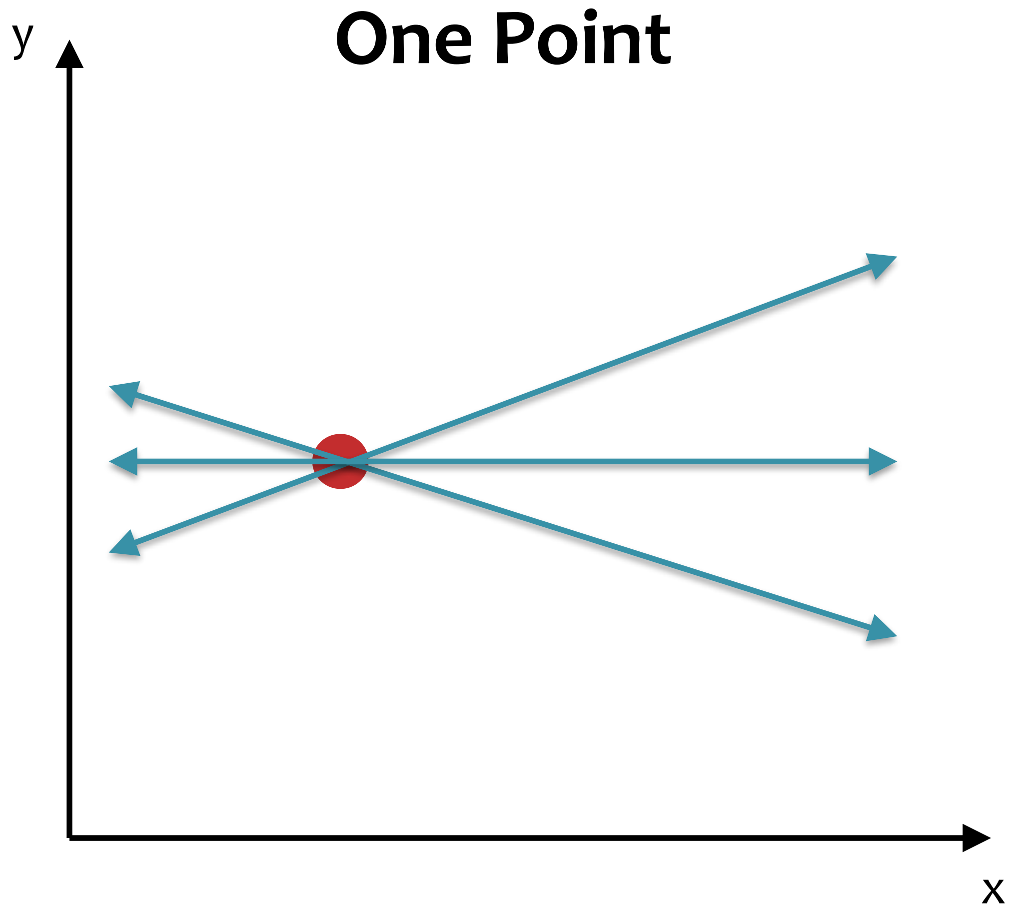


Linear Regression: Uniqueness

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?

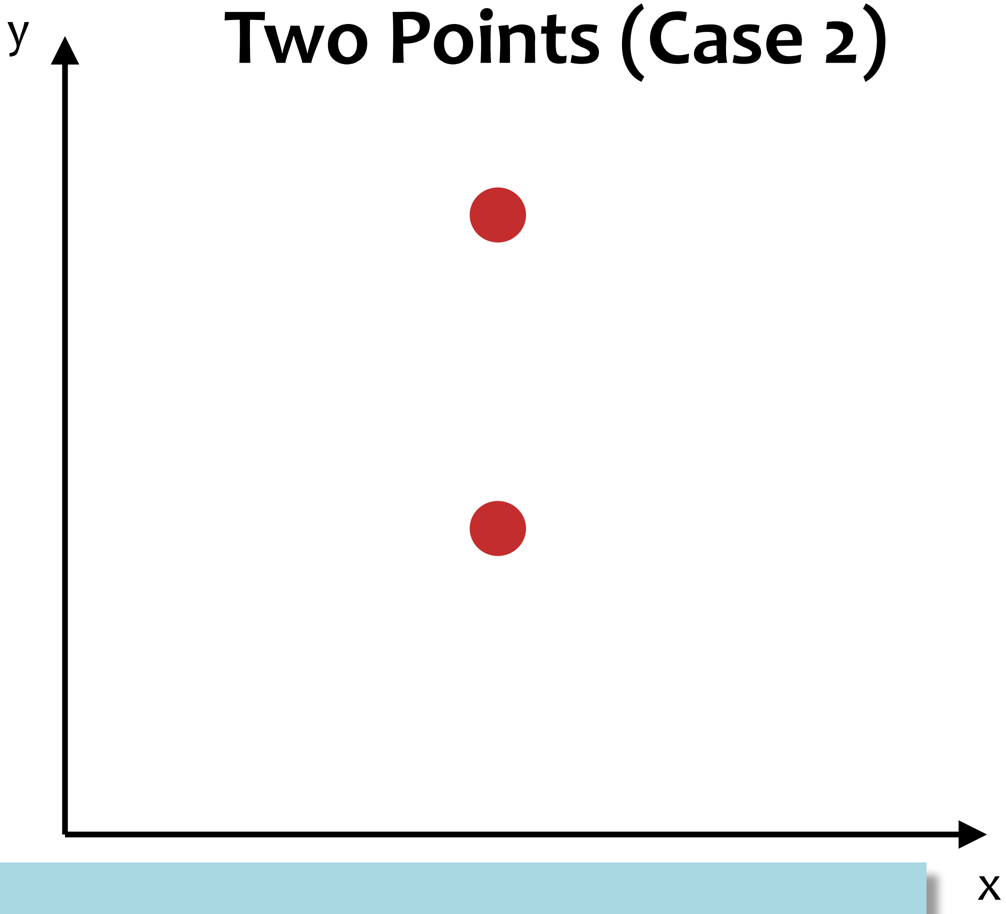


Linear Regression: Uniqueness

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?



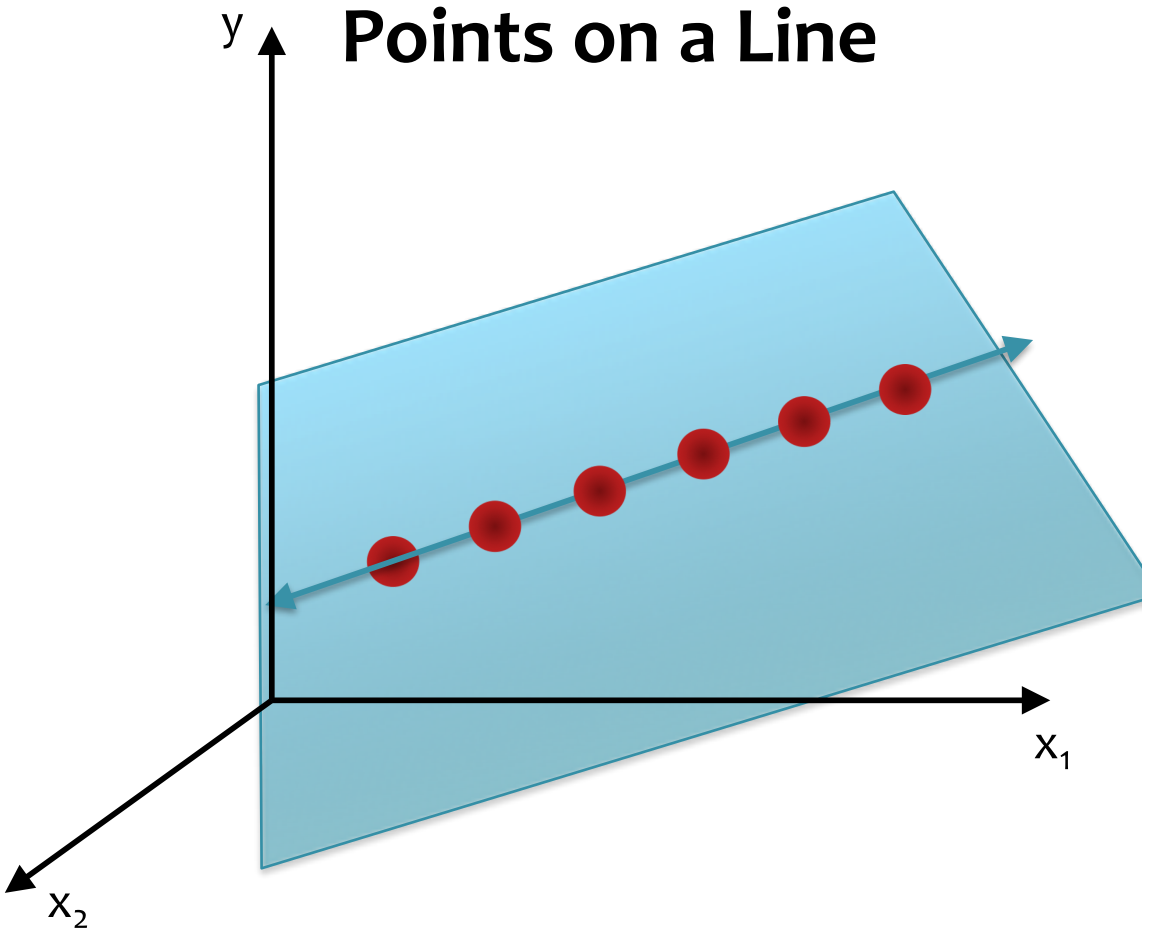
Answer:

A: 0 B: 1 C: 2 D: $+\infty$

Linear Regression: Uniqueness

Question:

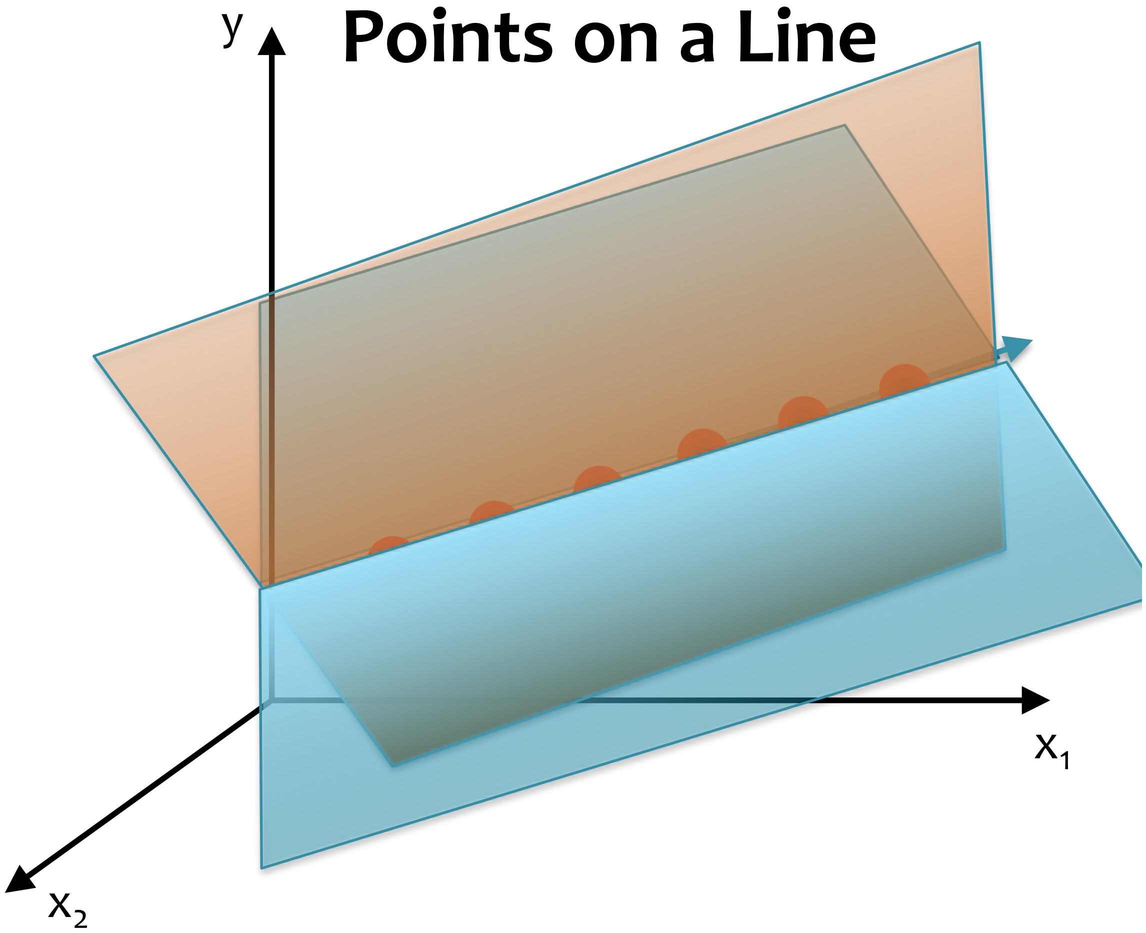
- Consider a **2D** linear regression model trained to minimize MSE
- How many solutions (i.e. sets of parameters w_1, w_2, b) are there for the given dataset?



Linear Regression: Uniqueness

Question:

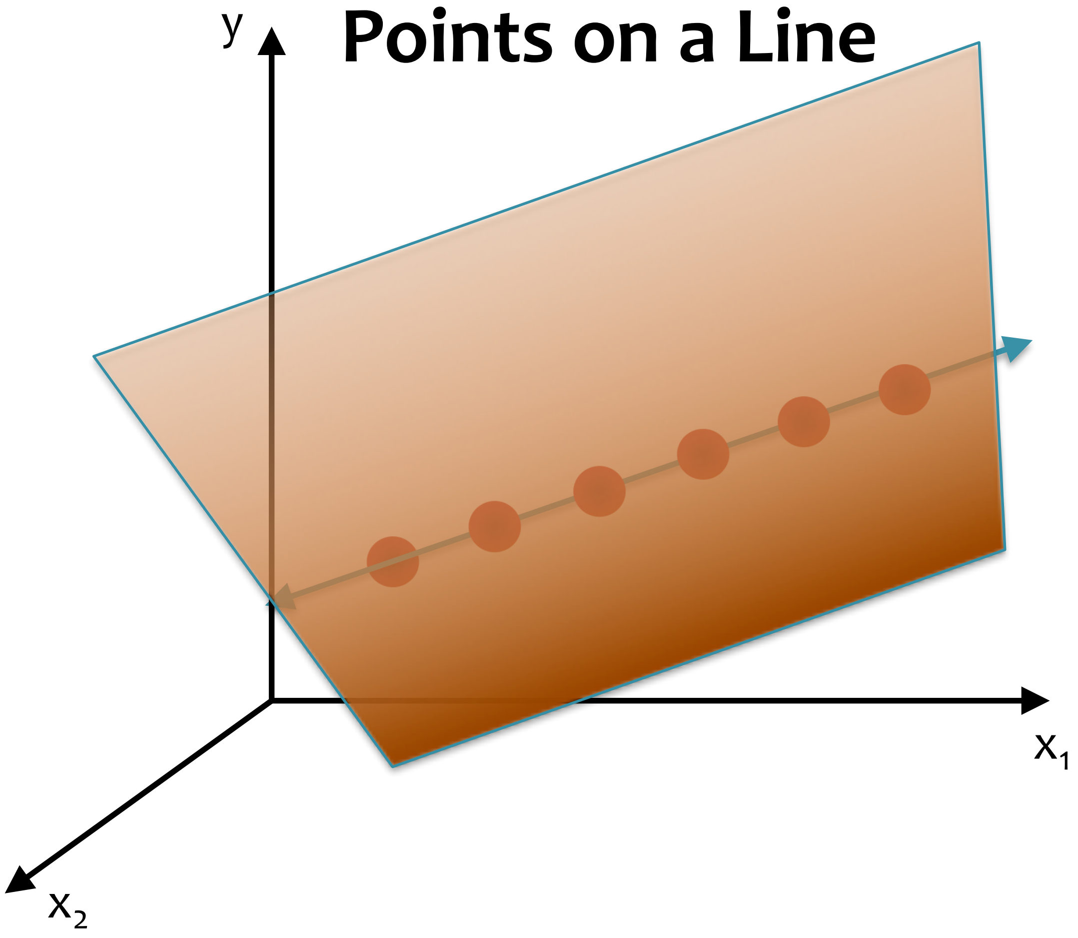
- Consider a **2D** linear regression model trained to minimize MSE
- How many solutions (i.e. sets of parameters w_1, w_2, b) are there for the given dataset?



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:

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

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:

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

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

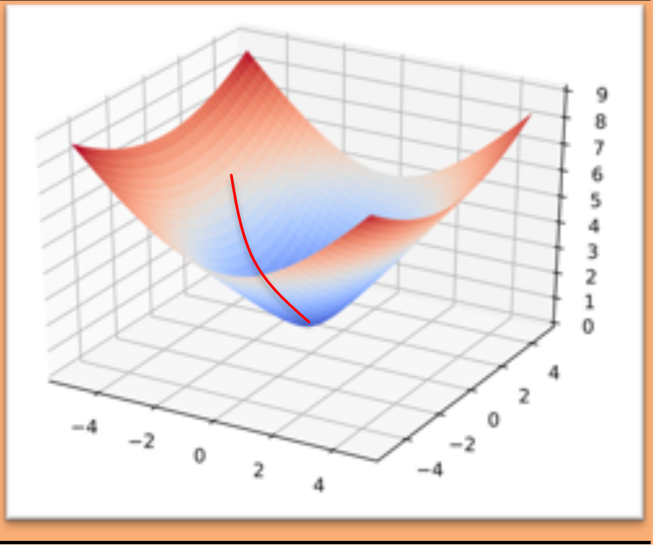
Invertability of $(\mathbf{X}^T\mathbf{X})$ is equivalent to X being **full rank.** That is, there is **no feature that is a linear combination of the other features.**

OPTIMIZATION METHOD #3: STOCHASTIC GRADIENT DESCENT

Gradient Descent

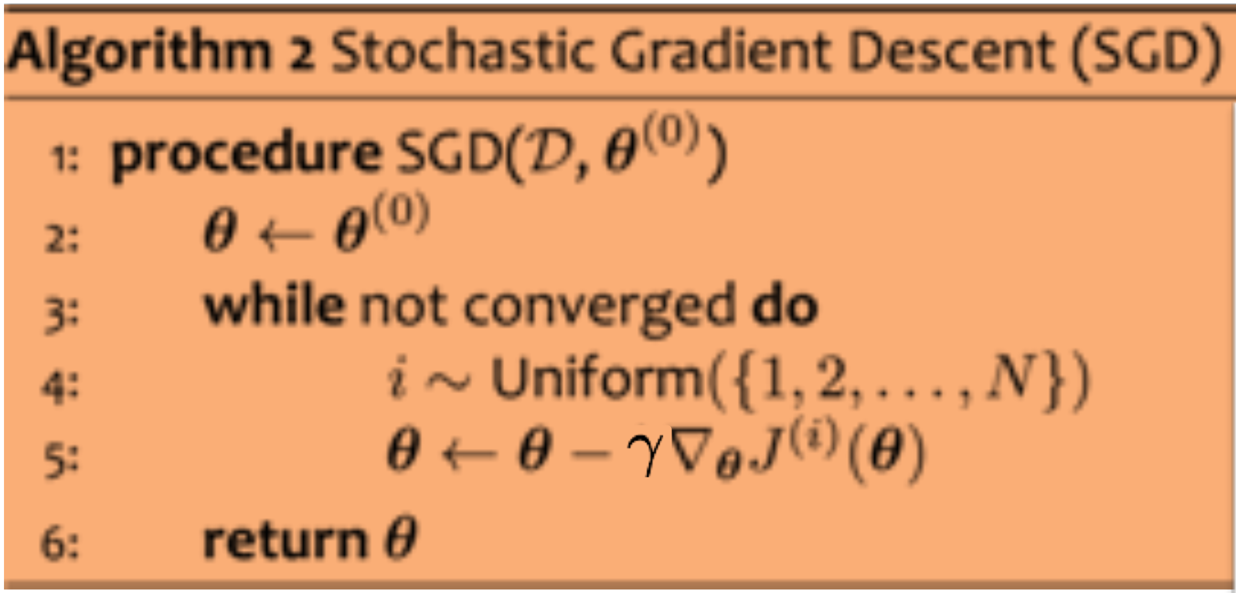
Algorithm 1 Gradient Descent

```
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 $\boldsymbol{\theta}$
```



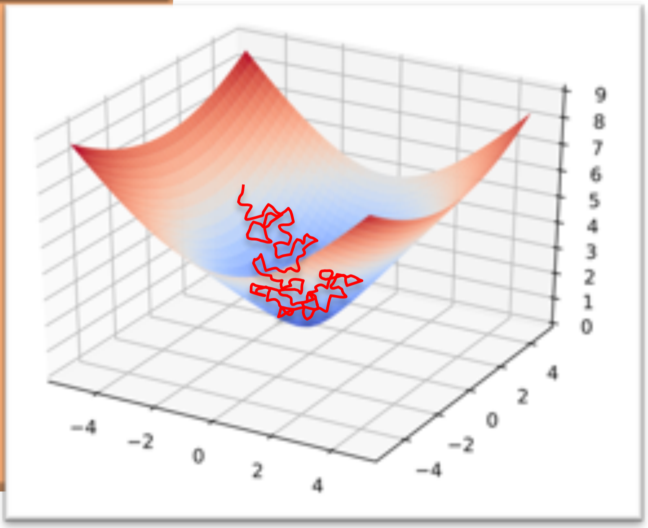
M

Stochastic Gradient Descent (SGD)



Algorithm 2 Stochastic Gradient Descent (SGD)

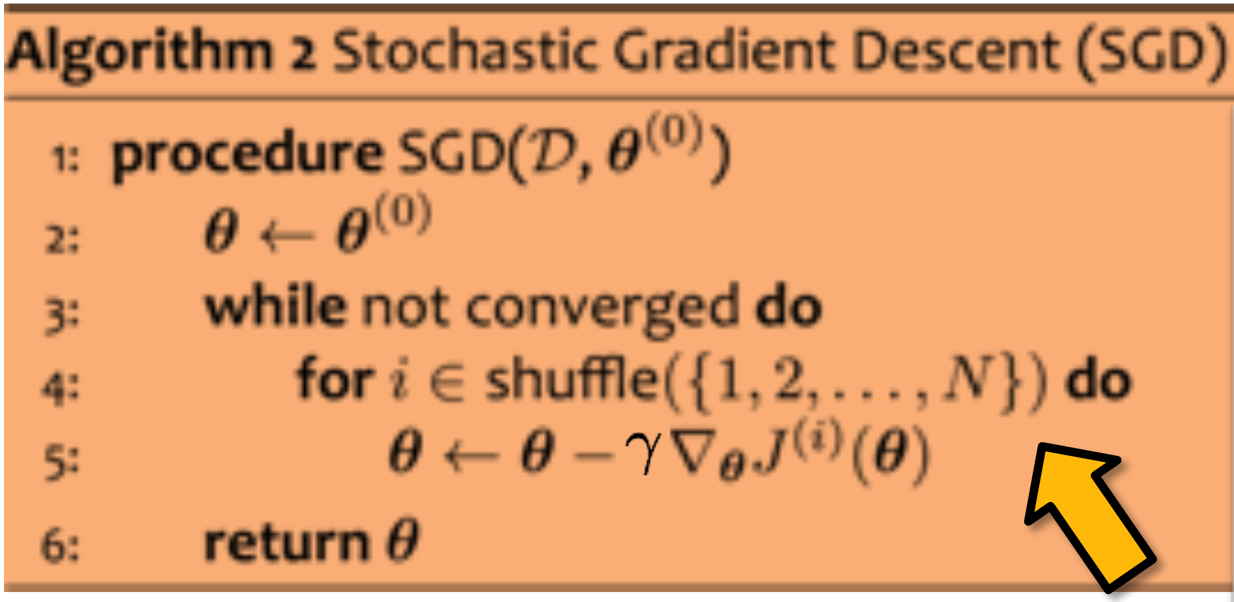
```
procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged do
        $i \sim$ Uniform($\{1, 2, \ldots, N\}$)
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$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



We need a per-example objective:

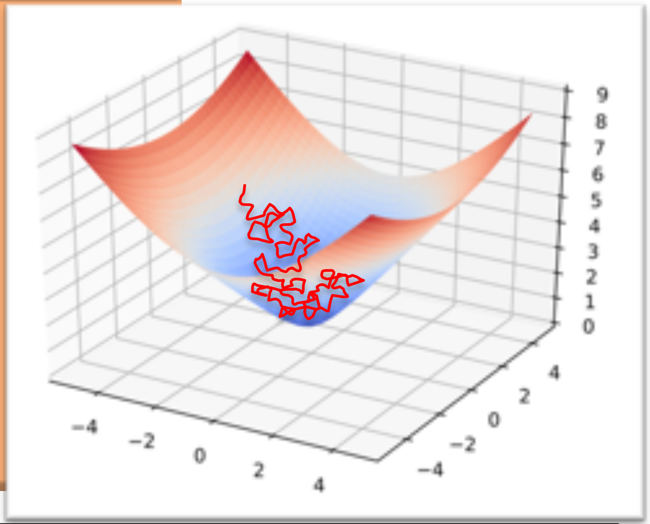
$$\text{Let } J(\boldsymbol{\theta}) = \textstyle\sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent (SGD)



Algorithm 2 Stochastic Gradient Descent (SGD)

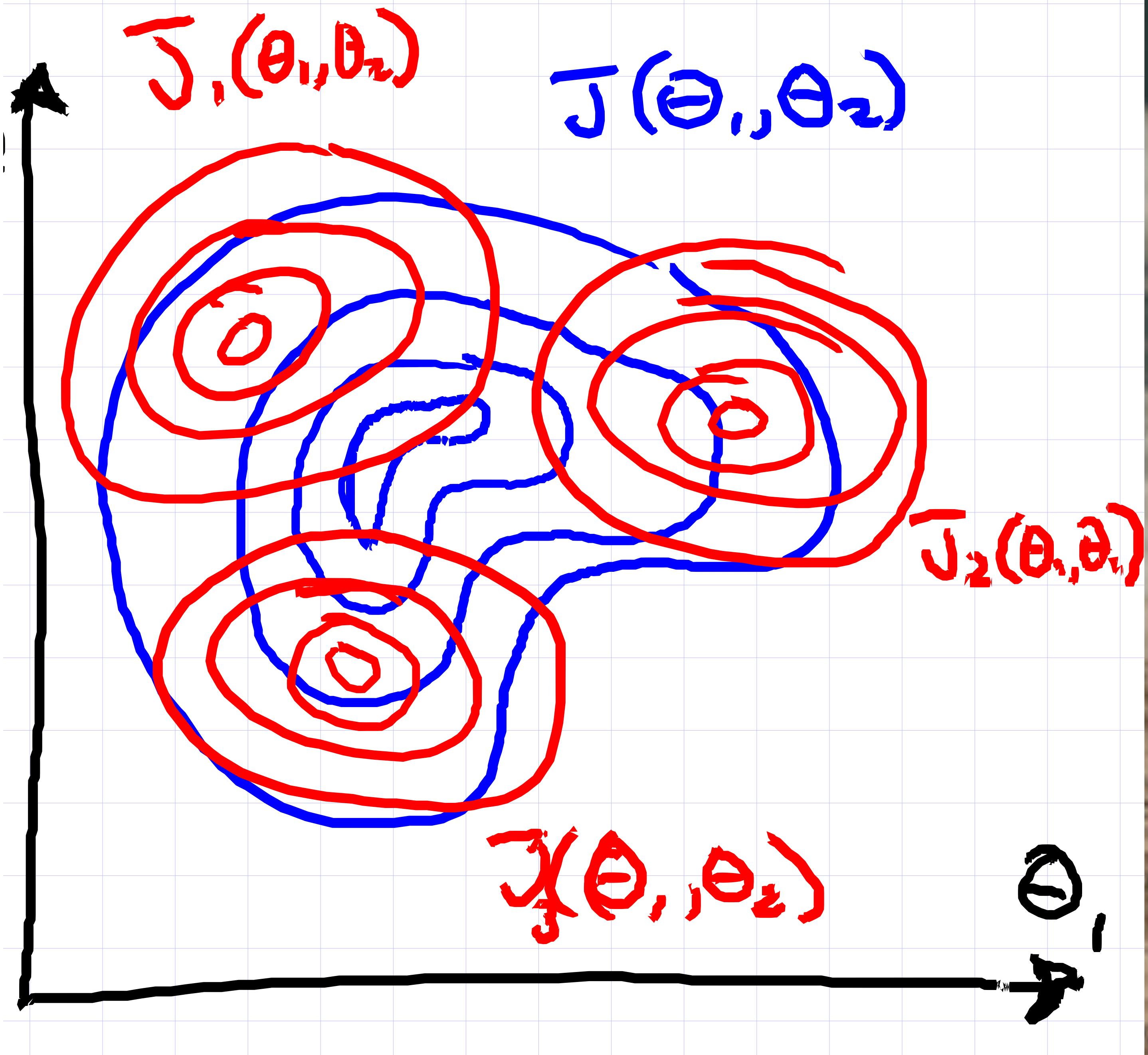
```
procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
    while not converged do
        for $i \in$ shuffle($\{1, 2, \dots, N\}$) do
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
    return $\boldsymbol{\theta}$
```



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}).

We need a per-example objective:

$$\text{Let } J(\boldsymbol{\theta}) = \textstyle\sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$


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

Expectations of Gradients

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

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

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

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

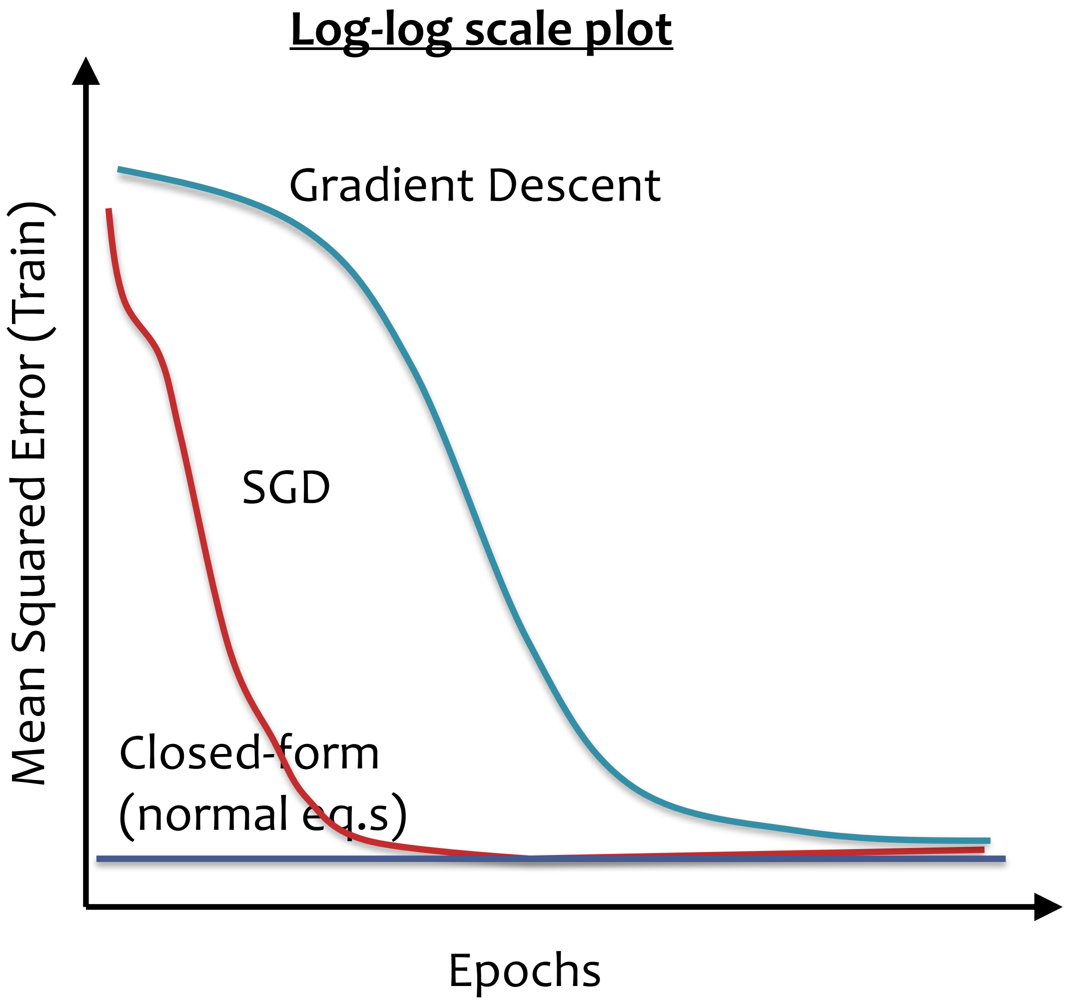
$$E_I[\nabla J_I(\vec{\theta})] = \sum_{i=1}^{N} P(I=i)\nabla J_i(\vec{\theta})$$

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

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

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:

true unknown min

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

not covered

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

"almost sure" convergence

lots of caveats and conditions

way less computation

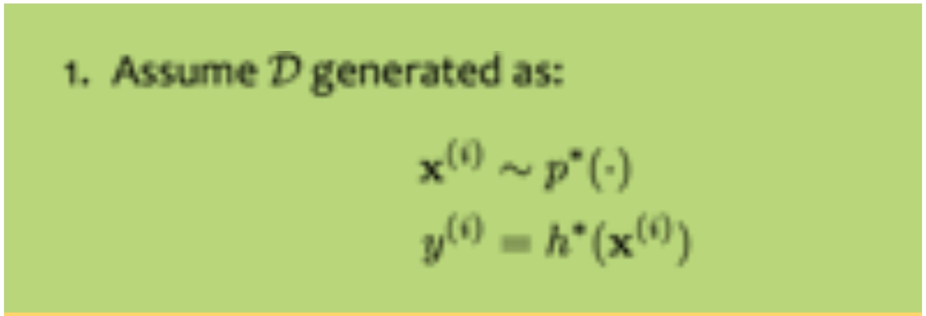
Takeaway: SGD has much slower asymptotic convergence. but is often faster in practice.

SGD FOR
LINEAR REGRESSION

Linear Regression as Function Approximation

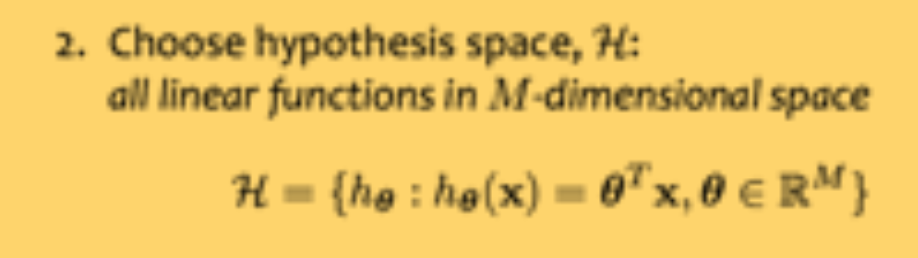
$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$



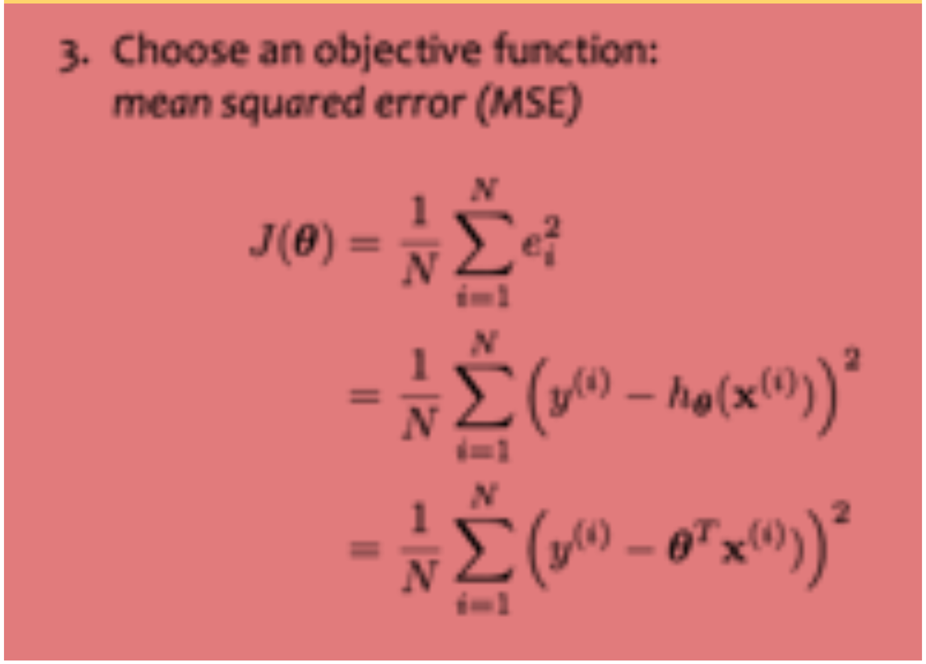
1. Assume $\mathcal{D}$ generated as:

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



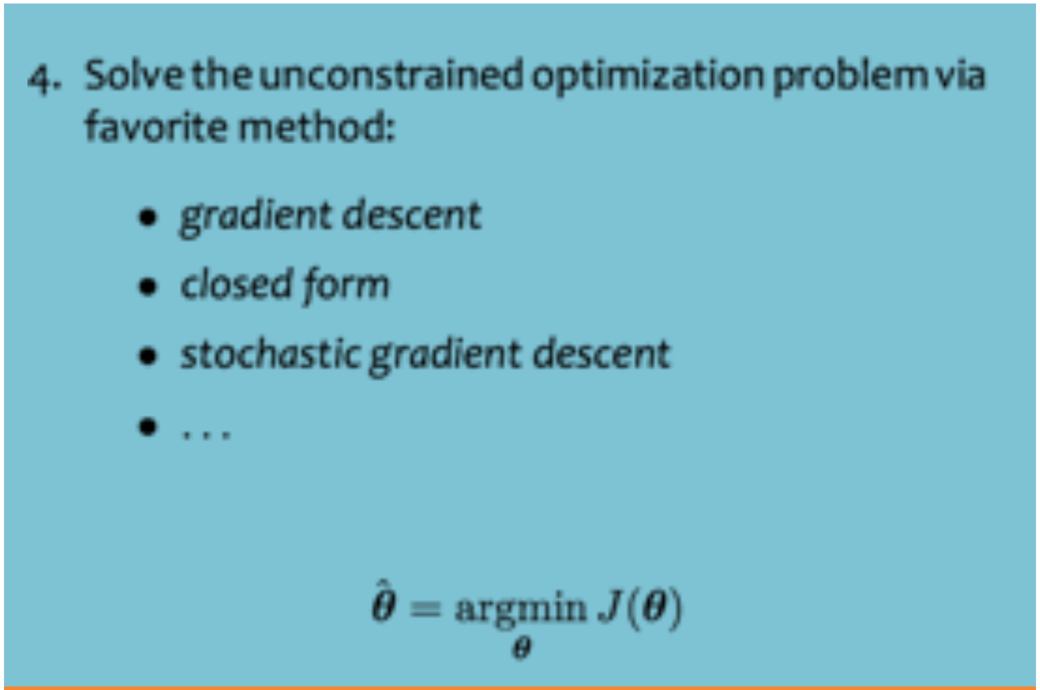
2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

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



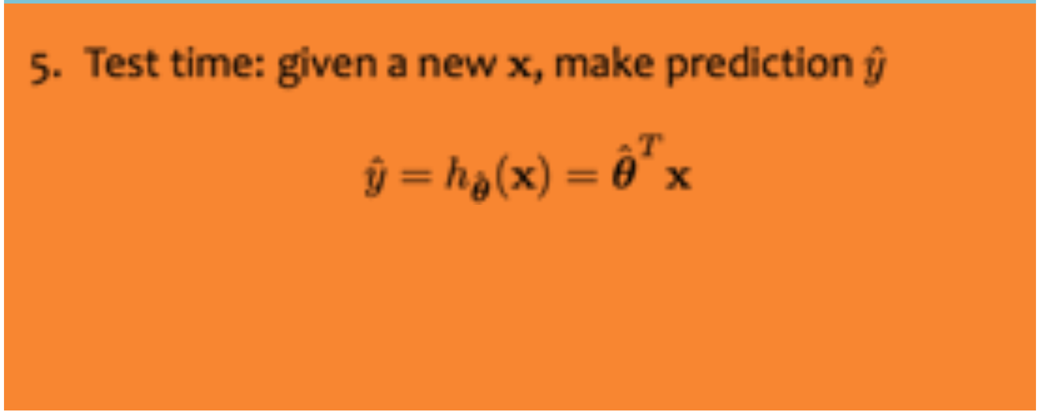
3. Choose an objective function:
mean squared error (MSE)

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



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

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



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]

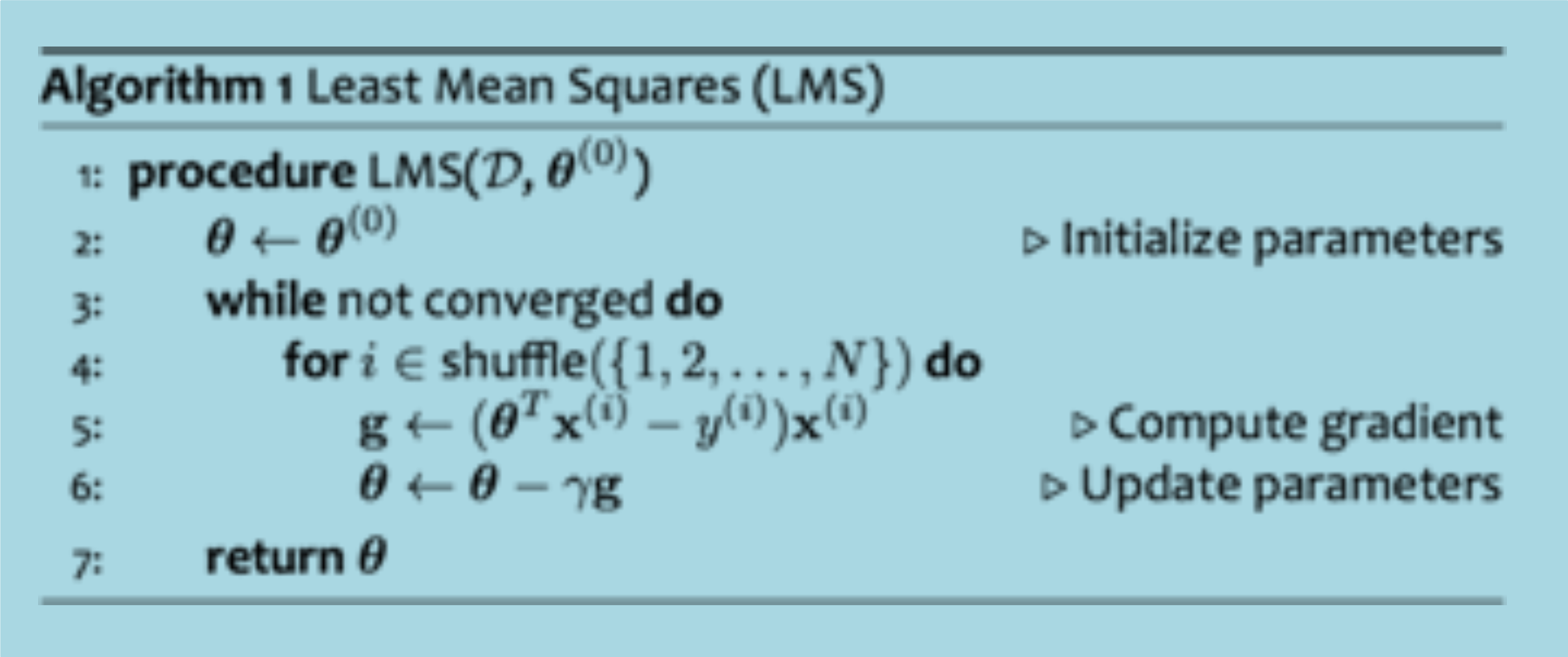
$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) &= \begin{bmatrix} \frac{d}{d\theta_1} J^{(i)}(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J^{(i)}(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J^{(i)}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^T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

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

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

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($\{1, 2, \ldots, N\}$) 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 $\boldsymbol{\theta}$
```

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

```
procedure GDLR($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$                                        ▷ Initialize parameters
    while not converged do
        $\mathbf{g} \leftarrow \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$    ▷ 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.

PROBABILISTIC LEARNING

Probabilistic Learning

Function Approximation

Previously, we assumed that our output was generated using a **deterministic target function:**

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

$$y^{(i)} = c^*(\mathbf{x}^{(i)})$$

Our goal was to learn a hypothesis h(**x**) that best approximates c^*(**x**)

Probabilistic Learning

Today, we assume that our output is **sampled** from a conditional **probability distribution:**

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

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

Our goal is to learn a probability distribution p(y|**x**) that best approximates p^*(y|**x**)

Robotic Farming

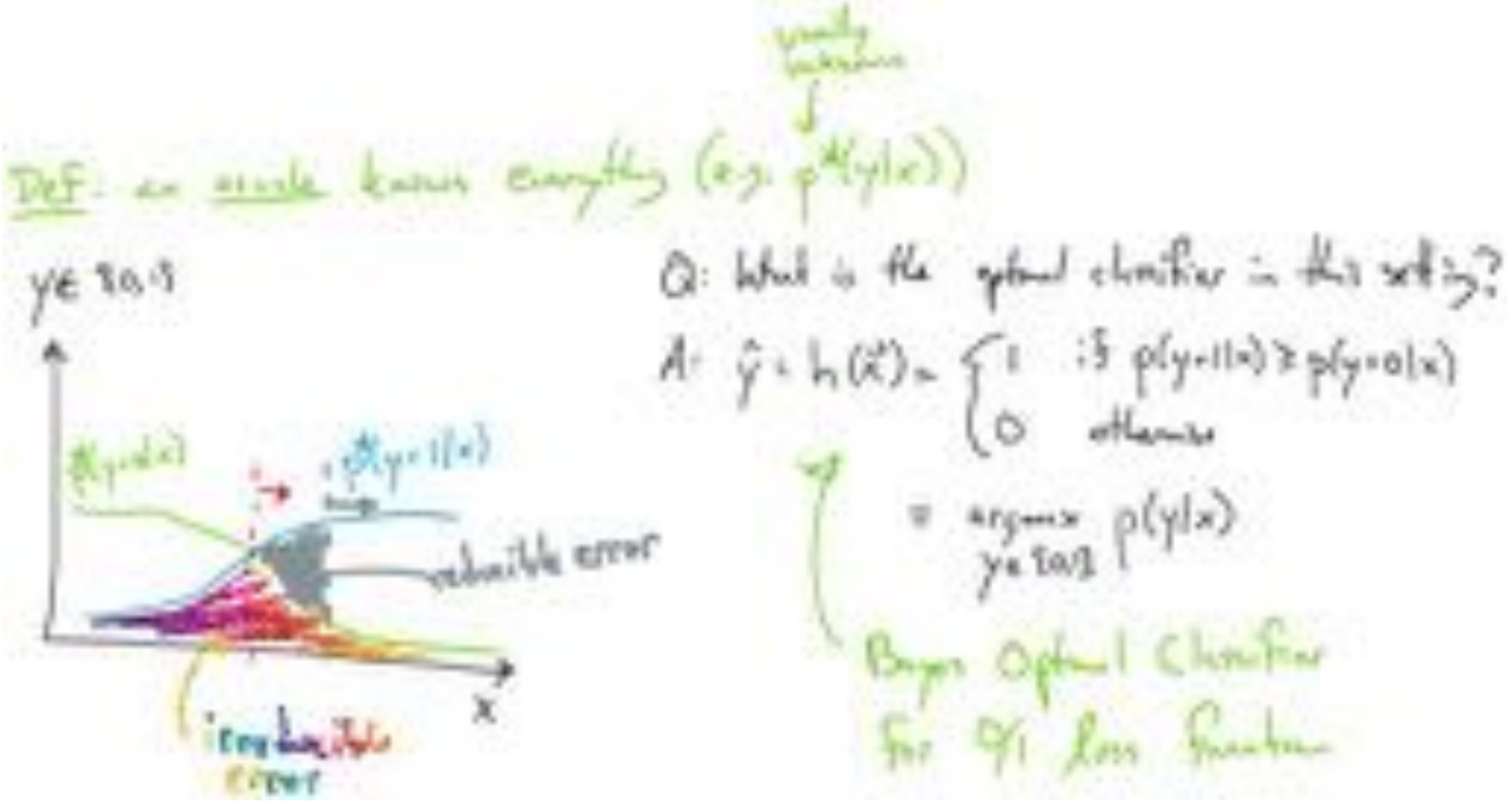
	Deterministic	**Probabilistic**
Classification (binary output)	Is this a picture of a wheat kernel?	Is this plant drought resistant?
Regression (continuous output)	How many wheat kernels are in this picture?	What will the yield of this plant be?

Bayes Optimal Classifier

Whiteboard

- Bayes Optimal Classifier
- Reducible / irreducible error
- Ex: Bayes Optimal Classifier for 0/1 Loss

Bayes Optimal Classifier



usually unknown

Def: an oracle knows everything (e.g. $p^*(y|x)$)

Q: What is the optimal classifier in this setting?

A: $\hat{y} = h(\vec{x}) = \begin{cases} 1 & \text{if } p(y=1|x) \geq p(y=0|x) \\ 0 & \text{otherwise} \end{cases}$

$= \underset{y \in \{0,1\}}{\operatorname{argmax}}\ p(y|x)$

Bayes Optimal Classifier for 0/1 loss function

MLE

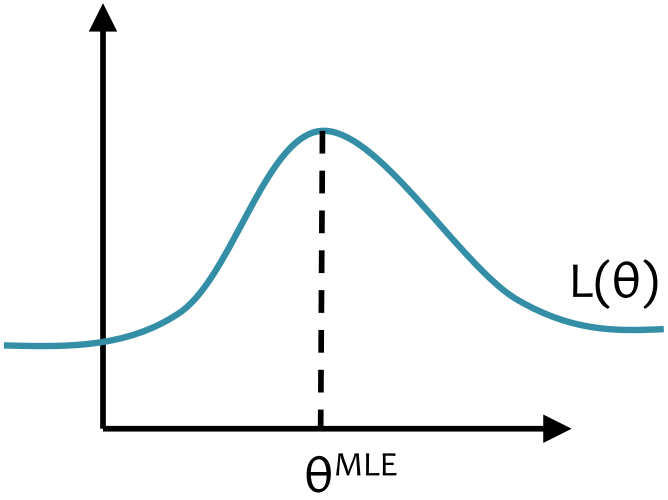
Suppose we have data $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$

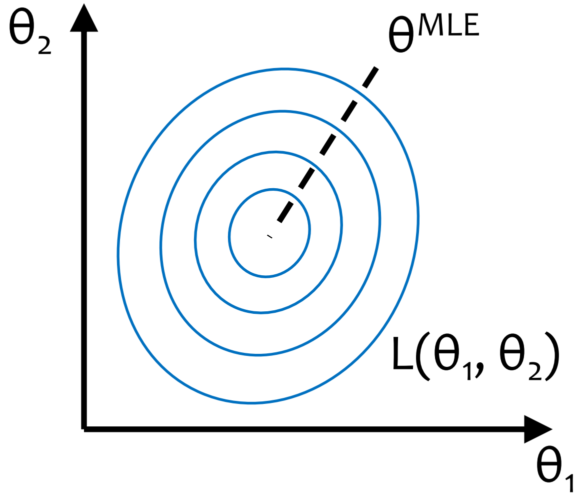
Principle of Maximum Likelihood Estimation:

Choose the parameters that maximize the likelihood of the data.

$$\boldsymbol{\theta}^{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta})$$

Maximum Likelihood Estimate (MLE)





MLE

What does maximizing likelihood accomplish?

- There is only a finite amount of probability mass (i.e. sum-to-one constraint)
- MLE tries to allocate **as much** probability mass **as possible** to the things we have observed…

  …**at the expense** of the things we have **not** observed

Maximum Likelihood Estimation

The principle of Maximum likelihood estimation (MLE):

Choose parameters that make the data "most likely".

Assumptions: Data generated iid from distribution $p^*(x|\vec{\theta}^*)$ and comes from a family of distr. parameterized $\theta \in \Theta$

← set of possible parameters

Formally:

$$\theta_{MLE} = \underset{\theta \in \Theta}{\arg\max}\ p(D|\theta)$$

$$= \underset{\theta \in \Theta}{\arg\max}\ \log p(D|\theta)$$

$$= \underset{\theta \in \Theta}{\arg\max}\ \ell(\theta)$$

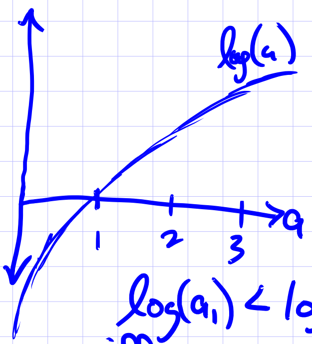
usually a continuous optimization

where $\ell(\theta) \triangleq \log p(D|\theta)$

"log-likelihood"

treat as function of θ where D is constant

since log is monotonic



$\log(a_1) < \log(a_2)$
iff $a_1 < a_2$

$\Rightarrow \log(f(a_1)) < \log(f(a_2))$
iff $f(a_1) < f(a_2)$