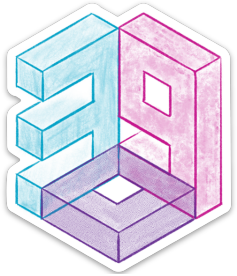


https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 1: Intro and Linear Regression

Shen Shen

Sept 4, 2025

11am, Room 10-250

Interactive Slides and Lecture Recording

Welcome to 6.390!

https://www.youtube-nocookie.com/embed/VhqtQdAGb7Q?si=sfKLandZIKl9jS2n

Outline

- Course Overview
  - Team, logistics, and topics overview
- Supervised learning, terminologies
- Ordinary least square regression
  - Formulation
  - Closed-form solutions

Team

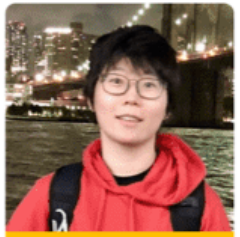

Pulkit Agrawal


Isaac L. Chuang

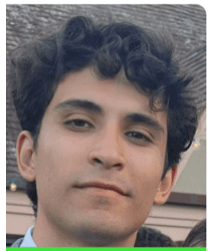

Dirk Englund


Vincent Monardo


Rajeev J. Ram


Shen Shen


Caroline Uhler


Mauricio Barba


Abhay Basireddy


Yiqing Du


Isaac Duitz


Chris Ge


Zhiyi Li


Anh Nguyen


Diego Rivero


DongHun Ryu


Inimai Subramanian


Aimee Wang


Elisa Xia


Alexander Zhang

~50 awesome LAs

Class meetings assignments

Week #	Monday	Tuesday	Wednesday	Thursday	Friday
N-1				9am: Exercise N released	9am: Homework *N* released
				11am-12:30pm: Lecture *N*	Recitation *N*
N		9am: Exercise *N* due			
		Lab *N*			
N+1			11:59pm: Homework *N* due		

Hours:

Lec: 1.5 hr

Rec + Lab: 3 hr

Notes + exercise: 2 hr

Homework: 6-7 hr

Grading and collaboration

- Our objective (and we hope yours) is for you to learn about machine learning
  - take responsibility for your understanding
  - we are here to help!
- Grades formula: exercises 5% + homework 20% + labs 15% + midterm 30% + final 30%
- Lateness: 20% penalty per day, applied linearly (so 1 hour late is -0.83%)
- Extensions:
  - 20 one-day extensions (extend one assignment's deadline by one full day), will be applied *automatically at the end of the term* in a way that is maximally helpful

Grading and collaboration

- Midterm 1: Wednesday, October 8, 730pm-9pm
- Midterm 2: Wednesday, Nov 12, 730pm-9pm
- Final: scheduled by Registrar (posted in 3rd week). ⚠️ – might be as late as Dec 19!

Detailed exam logistics will be posted 3 weeks before the exam date.

- Collaboration:
  - Understand everything you turn in
  - Coding and detailed derivations must be done by you
  - See collaboration policy/examples on course web site

How to get help

- Office hours: lots! (Starting Wed Sept 10)
- Schedule details on OHs page (includes instructors' OHs)
- See Calendar page for holiday/schedule shift
- Make use of Piazza and Pset-partners!
- Logistic, personal issues, reach out to ***6.390-personal@mit.edu*** **(**looping in ***S^3 and/or DAS)***

What we're teaching: Machine Learning

Given:

- a collection of examples (gene sequences, documents, …)
- an encoding of those examples in a computer (as vectors)

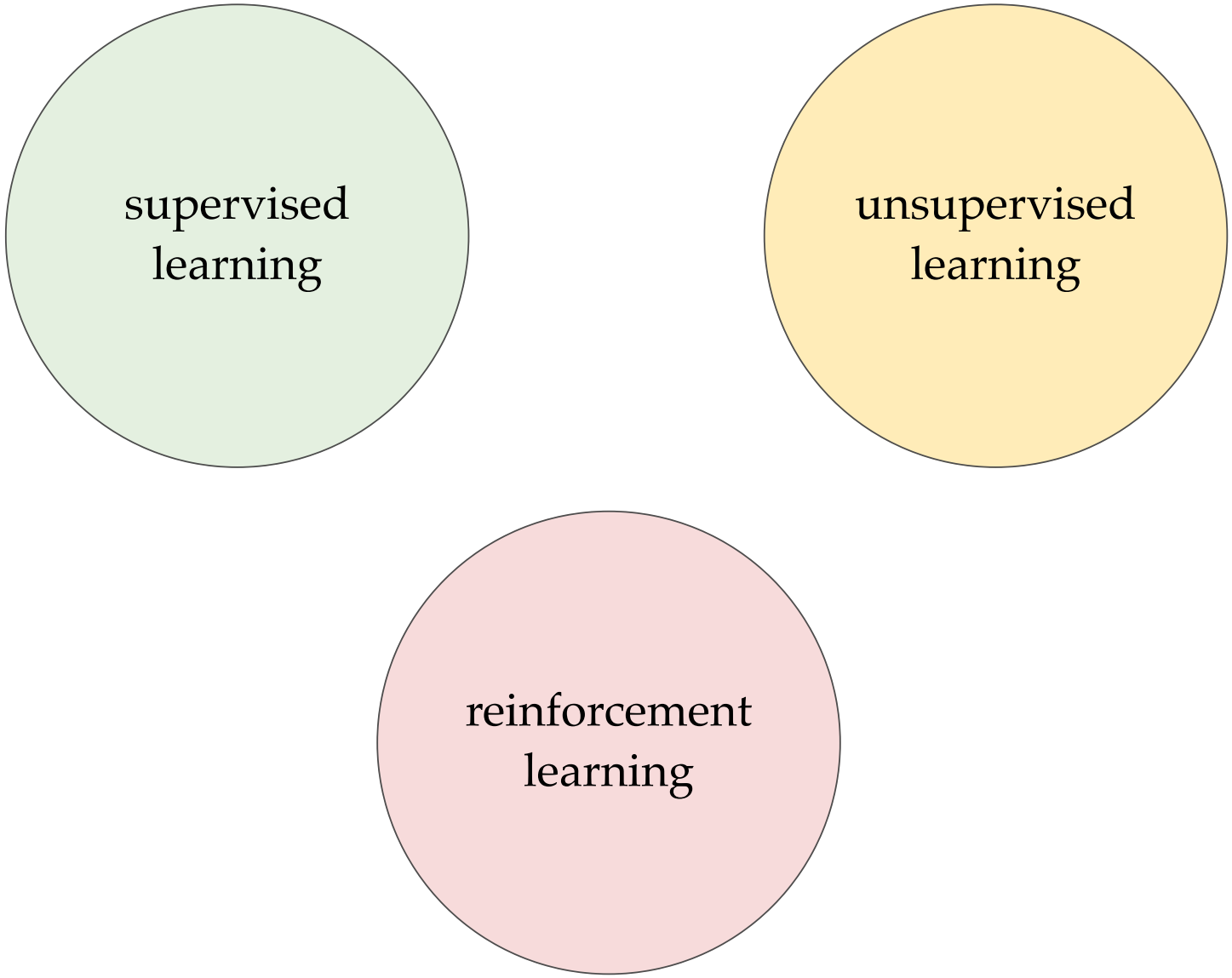
Derive:

- a computational model that describes relationships within and among the examples that is expected to characterize well new examples from that same population, to make good predictions or decisions

A model might:

- classify images of cells as to whether they're cancerous
- specify groupings (clusters) of documents that address similar topics
- steer a car appropriately given lidar images of the surroundings

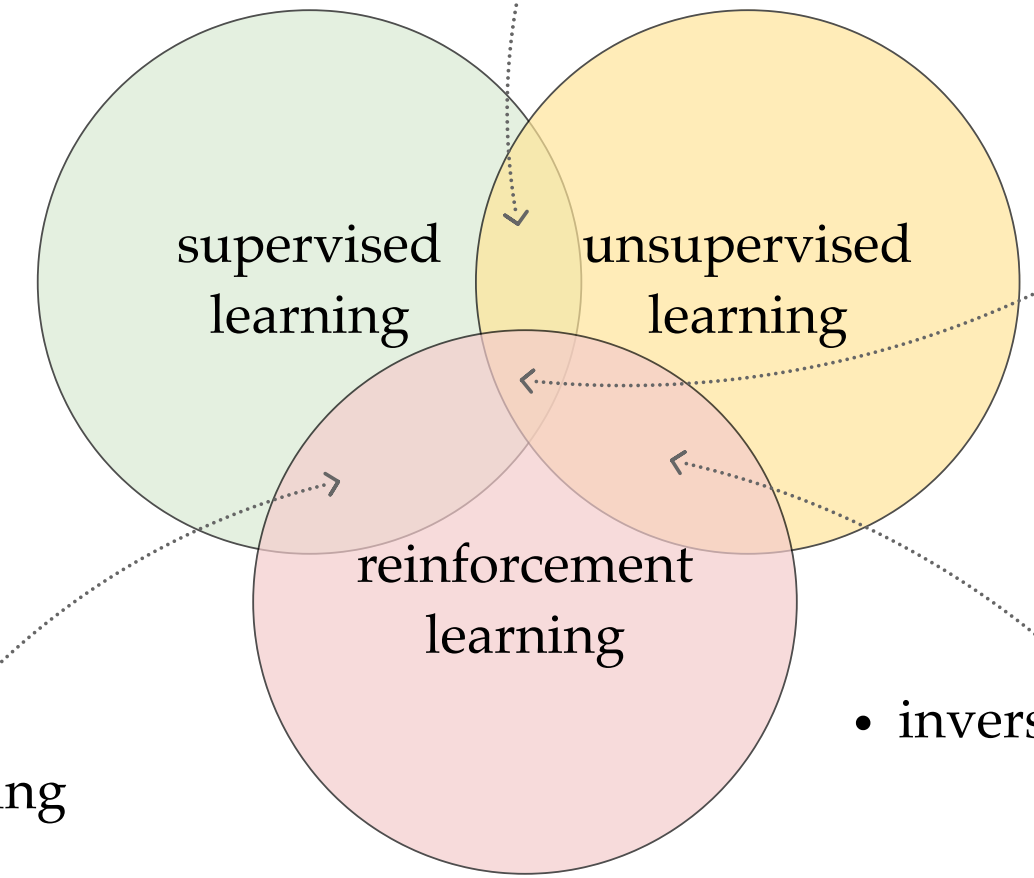
traditionally



nowadays

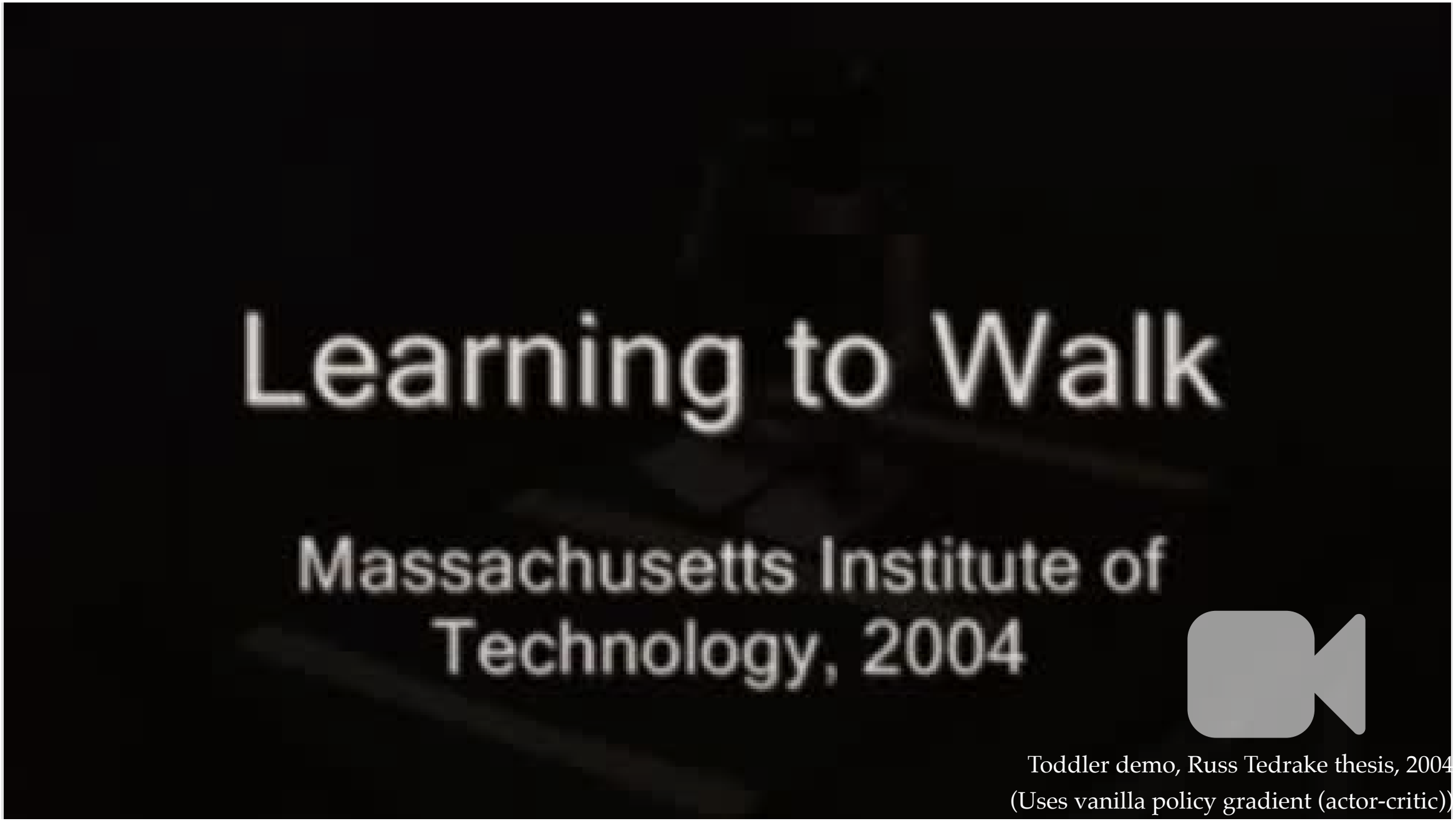
- self-supervised
- contrastive learning (DALLE)

RLHF (ChatGPT etc.)



- behavior cloning

- inverse reinforcement learning



Toddler demo, Russ Tedrake thesis, 2004
(Uses vanilla policy gradient (actor-critic))

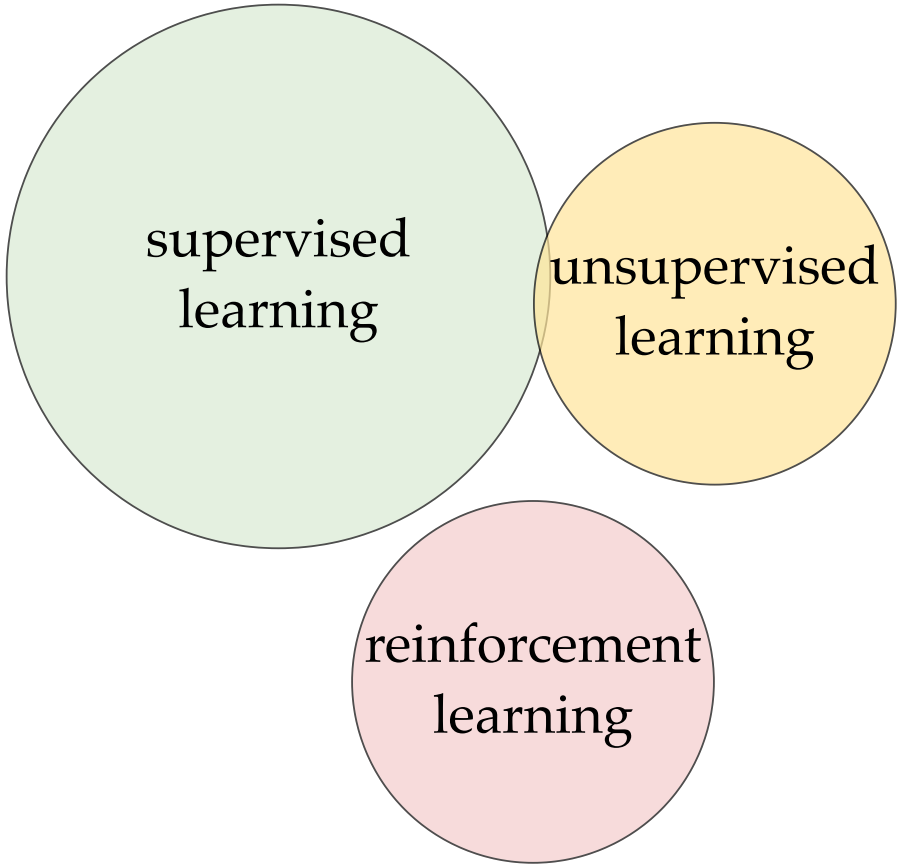
DARPA Robotics Competition

2015

Optimization + first-principle physics

https://www.physicalintelligence.company/blog/pi0

In 6.390:


supervised learning
unsupervised learning
reinforcement learning

Topics in order:

- Intro to ML
- Regularization
- Gradient Descent
- Linear Classification

supervised

- Features, Neural Networks I
- Neural Networks II (Backprop)
- Convolutional Neural Networks
- Representation Learning
- Transformers

unsupervised

- Non-parametric Models

- Markov Decision Processes
- Reinforcement Learning

reinforcement

Many other ways to dissect

Model class:

- linear models
- linear model on non-linear features
- fully connected feed-forward nets
- convolutional nets
- transformers
- Q-table
- tree, k-nearest neighbor, k-means

Modeling choices:

- Supervised:
  - regression
  - classification
- Unsupervised/self-supervised
- Reinforcement/sequential

Learning process:

- training/validation/testing
- overfitting/underfitting
- regularization
- hyper parameters

Optimization:

- analytical solutions
- gradient descent
- back propagation
- value iteration, Q-learning
- non-parametric methods

[These lists are neither exhaustive nor exclusive.]

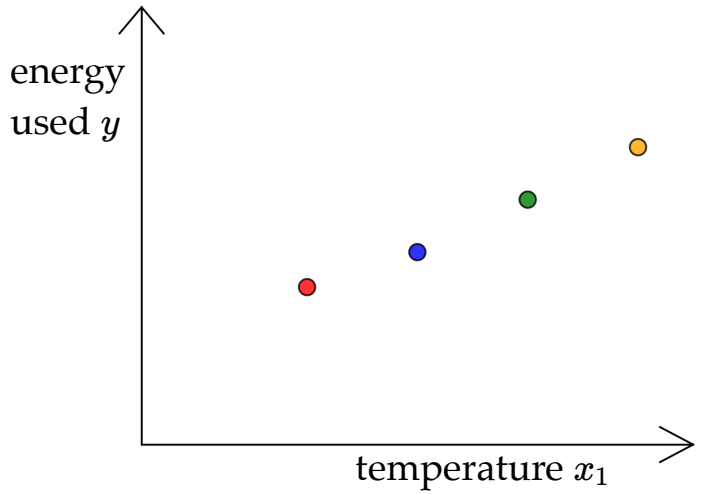
Outline

- Course Overview
- Supervised learning, terminologies
- Ordinary least square regression
  - Formulation
  - Closed-form solutions

We first focus on an instance of *supervised learning* known as *regression*.

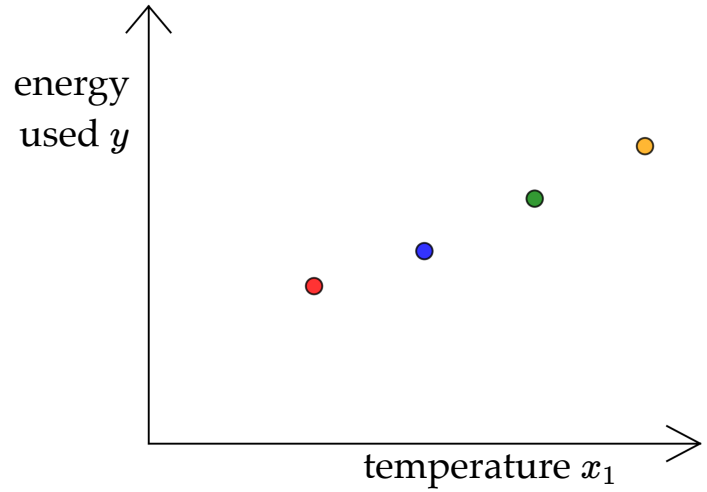
`example: city daily energy consumption prediction`

	Features	**Label**
City	Temperature (°C)	Energy used (GWh)
Chicago	25	51
New York	28	57
Boston	31	63
San Diego	35	71

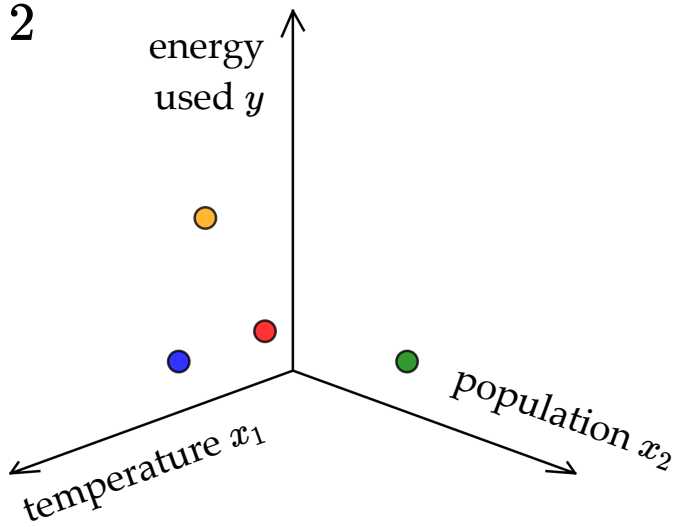


toy data, for illustration only

$n = 4, d = 1$



$n = 4, d = 2$



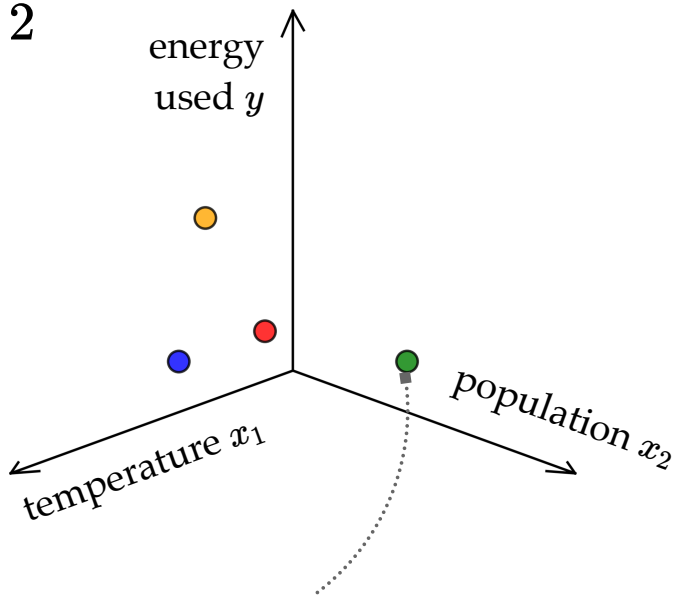
Training data:

$$\mathcal{D}_{\text{train}} := \left\{ \left(x^{(1)}, y^{(1)} \right), \ldots, \left(x^{(n)}, y^{(n)} \right) \right\}$$

feature vector — label

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_d^{(1)} \end{bmatrix} \in \mathbb{R}^d \qquad y^{(1)} \in \mathbb{R}$$

$n = 4, d = 2$



$$(x^{(1)}, y^{(1)})$$

$$= \left(\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix}, y^{(1)} \right)$$

Training data:

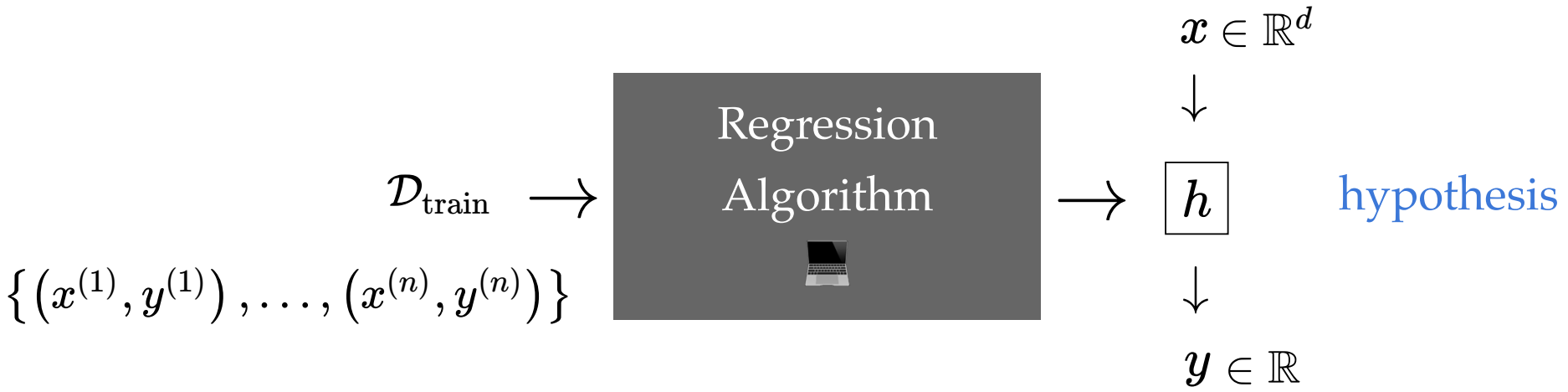
$$\mathcal{D}_{\text{train}} := \left\{ \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \right\}$$

feature vector

label

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_d^{(1)} \end{bmatrix} \in \mathbb{R}^d$$

$$y^{(1)} \in \mathbb{R}$$

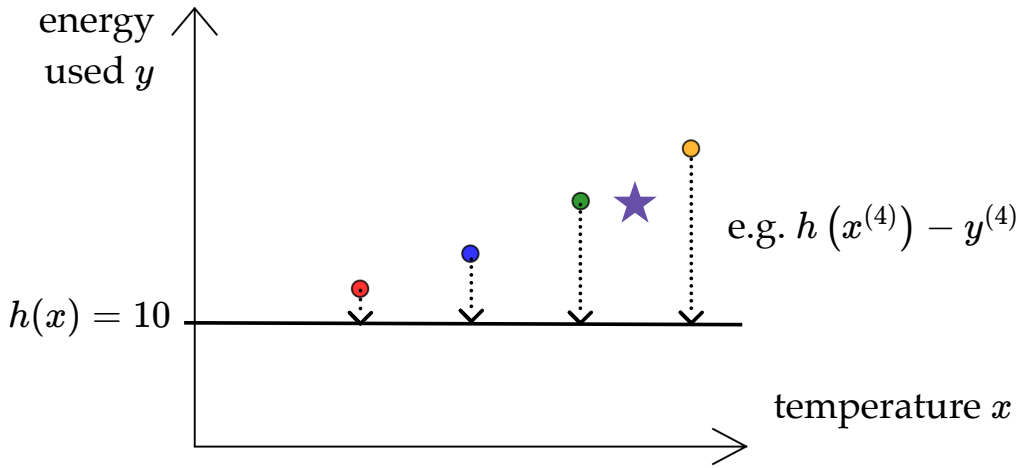


What do we want from the regression algortim?

A good way to label *new* features, i.e. a *good* hypothesis.

Suppose our friend's algorithm proposes $h(x) = 10$

- Is this a hypothesis?
- Is this a "good" hypothesis? Or, what would be a "good" hypothesis?
- What can affect if and how we can find a "good" hypothesis?



- Loss $\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right)$

  e.g. squared loss $\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right) = \left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$

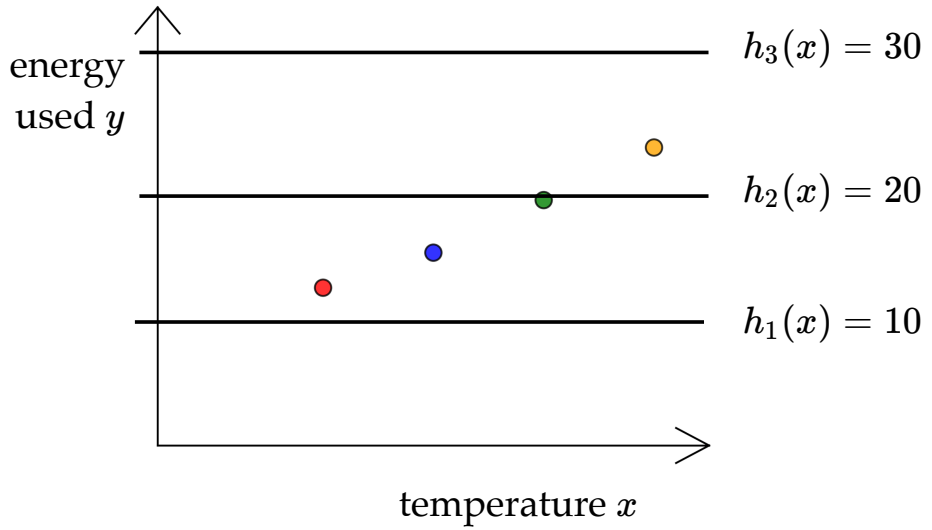
- Training error $\mathcal{E}_{\text{train}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right)$

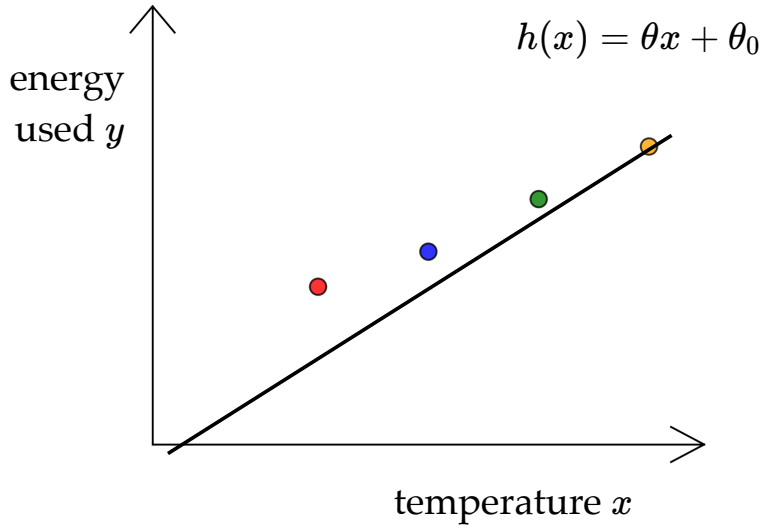
  e.g. with squared loss, the training error is the mean-squared-error (MSE)

- Test error $\mathcal{E}_{\text{test}}(h) = \frac{1}{n'}\sum_{i=n+1}^{n+n'} \mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right)$

  n' unseen data points, i.e. test data

Hypothesis class $\mathcal{H}$: set of h we ask the algorithm to search over

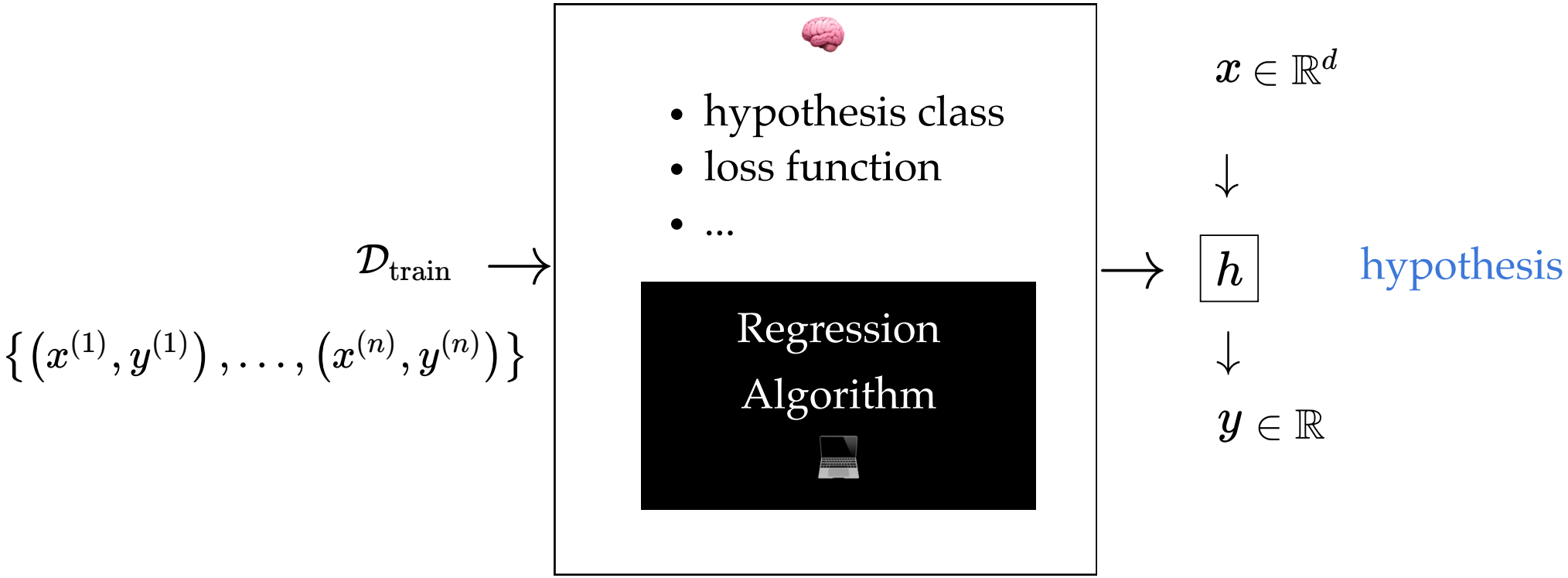




{constant functions} $\subset$ {linear functions}$_1$

less expressive | more expressive

1. technically, affine functions. ppl tend to be flexible about this terminology in ML.


$\mathcal{D}_{\text{train}}$
$\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right\}$
• hypothesis class
• loss function
• ...
Regression Algorithm
$x \in \mathbb{R}^d$
h
hypothesis
$y \in \mathbb{R}$

Quick summary

- Supervised learning
- Regression
- Training data, test data
- Features, label
- Loss function, training error, test error
- Hypothesis, hypothesis class

Outline

- Course Overview
- Supervised learning, terminologies
- Ordinary least square regression
  - Formulation
  - Closed-form solutions

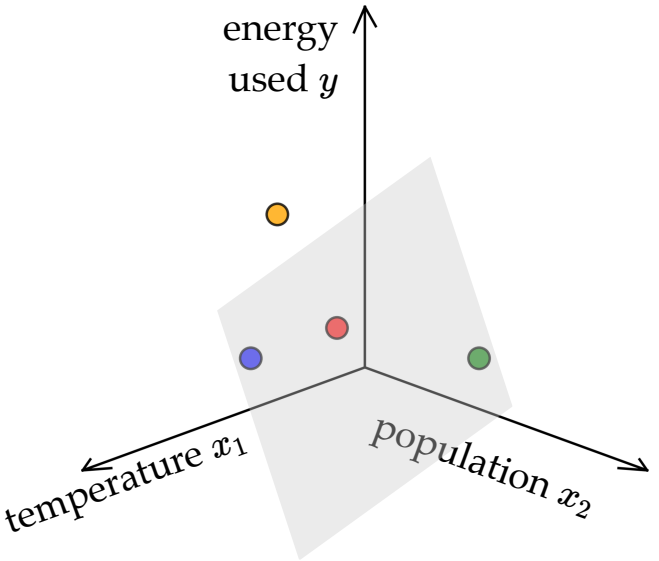
Linear least square regression

- Linear hypothesis class:

$$h(x;\theta) = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$= \theta^T x$$

parameters features



- Squared loss function:

$$\mathcal{L}\left(h\left(x^{(i)}\right), y^{(i)}\right) = (\theta^T x - y^{(i)})^2$$

for now, ignoring the offset

- MSE training error:

	Features		Label
City	**Temperature**	**Population**	**Energy Used**
Chicago	90	7.2	45
New York	20	9.5	32
Boston	35	8.4	99
San Diego	18	4.3	39

$$J(\theta_1, \theta_2) = \frac{1}{4}[(\theta_1 \cdot 90 + \theta_2 \cdot 7.2 - 45)^2 + (\theta_1 \cdot 20 + \theta_2 \cdot 9.5 - 32)^2 + (\theta_1 \cdot 35 + \theta_2 \cdot 8.4 - 99)^2 + (\theta_1 \cdot 18 + \theta_2 \cdot 4.3 - 39)^2]$$

- J denotes training error, sometimes the more explicitly $J(\theta; \text{data})$
- want a more compact way to write this out

$$J(\theta_1, \theta_2) = \frac{1}{4}[(e_1^2 + e_2^2 + e_3^2 + e_4^2)]$$

Let

$$X = \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \in \mathbb{R}^{n \times d} \qquad Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathbb{R}^{n \times 1} \qquad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \in \mathbb{R}^{d \times 1}$$

Then

$$J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y) \quad \in \mathbb{R}^{1 \times 1}$$

---

e.g.

	Features		**Label**
City	**Temperature**	**Population**	**Energy Used**
Chicago	90	7.2	45
New York	20	9.5	32
Boston	35	8.4	99
San Diego	18	4.3	39

$$X = \begin{bmatrix} 90 & 7.2 \\ 20 & 9.5 \\ 35 & 8.4 \\ 18 & 4.3 \end{bmatrix} \qquad Y = \begin{bmatrix} 45 \\ 32 \\ 99 \\ 39 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

	Features		Label
City	**Temperature**	**Population**	**Energy Used**
Chicago	90	7.2	45
New York	20	9.5	32
Boston	35	8.4	99
San Diego	18	4.3	39

$$X = \begin{bmatrix} 90 & 7.2 \\ 20 & 9.5 \\ 35 & 8.4 \\ 18 & 4.3 \end{bmatrix} \qquad Y = \begin{bmatrix} 45 \\ 32 \\ 99 \\ 39 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

deviation

$$X\theta - Y = \begin{bmatrix} 90\theta_1 + 7.2\theta_2 - 45 \\ 20\theta_1 + 9.5\theta_2 - 32 \\ 35\theta_1 + 8.4\theta_2 - 99 \\ 18\theta_1 + 4.3\theta_2 - 39 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix}$$

summing deviation squared

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 = \begin{bmatrix} e_1, e_2, e_3, e_4 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = (X\theta - Y)^\top (X\theta - Y)$$

training error (MSE): want to show:

$$J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y)$$

Outline

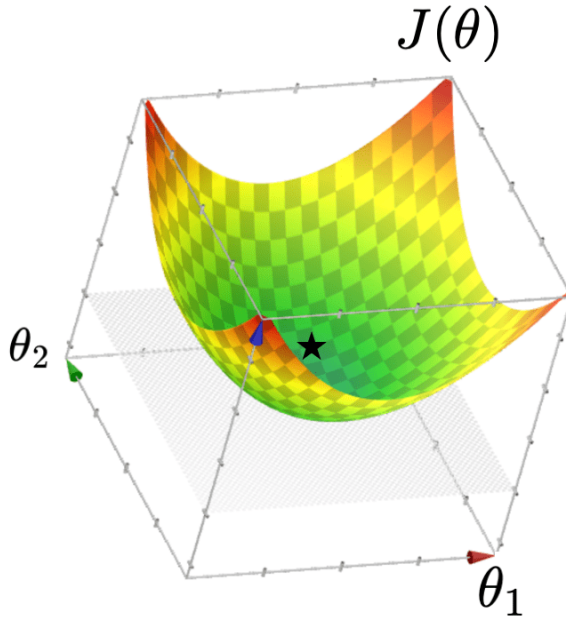
- Course Overview
- Supervised learning, terminologies
- Ordinary least square regression
  - Formulation
  - Closed-form solutions

Objective function (training error)

$$J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y)$$

goal: find θ to minimize $J(\theta)$

- Q: What kind of function is $J(\theta)$?
- A: Quadratic function
- Q: What does $J(\theta)$ look like?
- A: *Typically,* looks like a "bowl"
- Q: How to find the minimizer?



[1d case walk-through on board]

For $f : \mathbb{R}^m \to \mathbb{R}$, its ***gradient*** $\nabla f : \mathbb{R}^m \to \mathbb{R}^m$ is defined at the point $p = (x_1, \ldots, x_m)$ as:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$

1. The gradient generalizes the concept of a derivative to multiple dimensions.
2. By construction, the gradient's dimensionality always matches the function input.

Sometimes the gradient is undefined or ill-behaved, but today it is well-behaved.

3. The gradient can be symbolic or numerical.

example: $f(x, y, z) = x^2 + y^3 + z$

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$

its *symbolic* gradient:

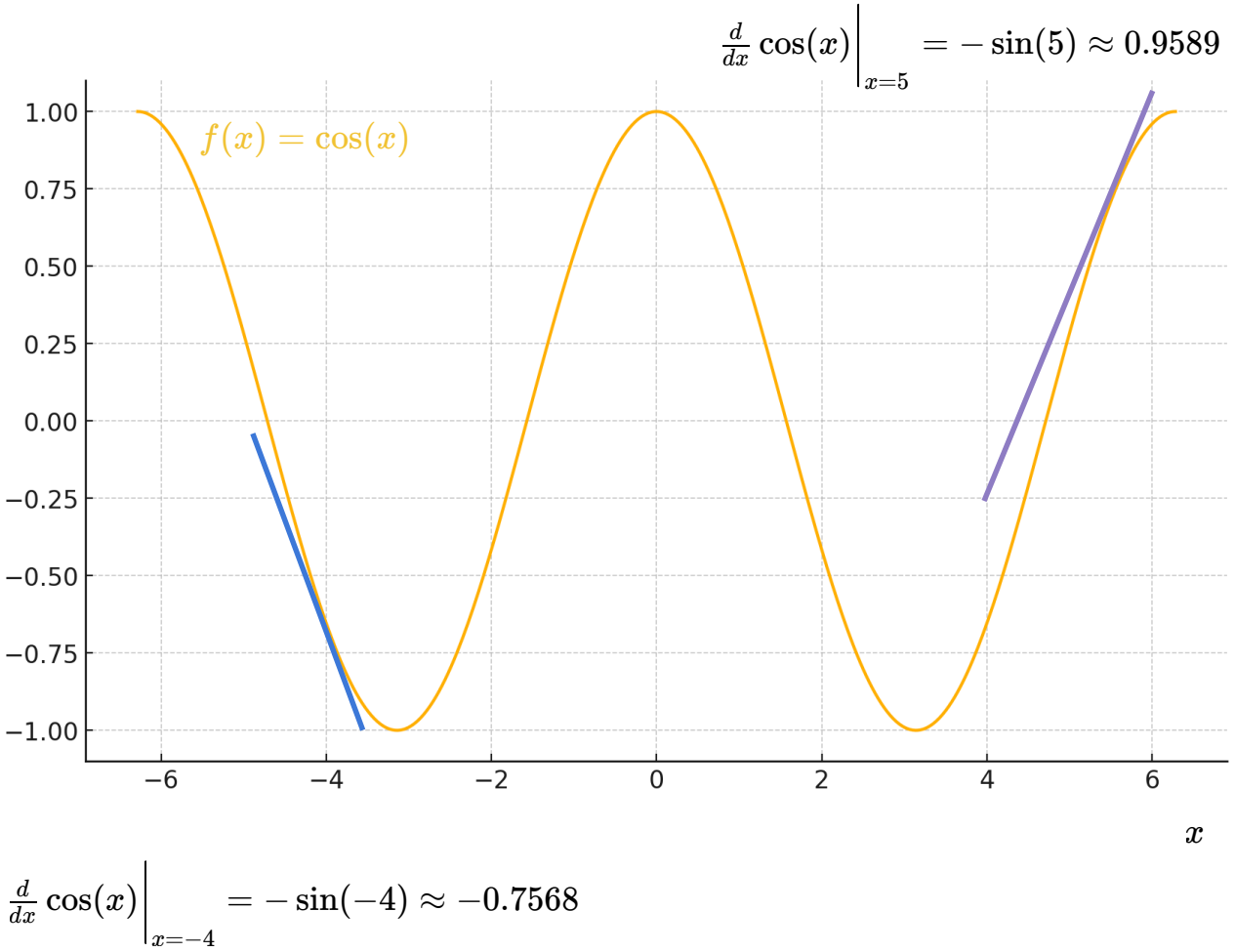
$$\nabla f(x, y, z) = \begin{bmatrix} 2x \\ 3y^2 \\ 1 \end{bmatrix}$$

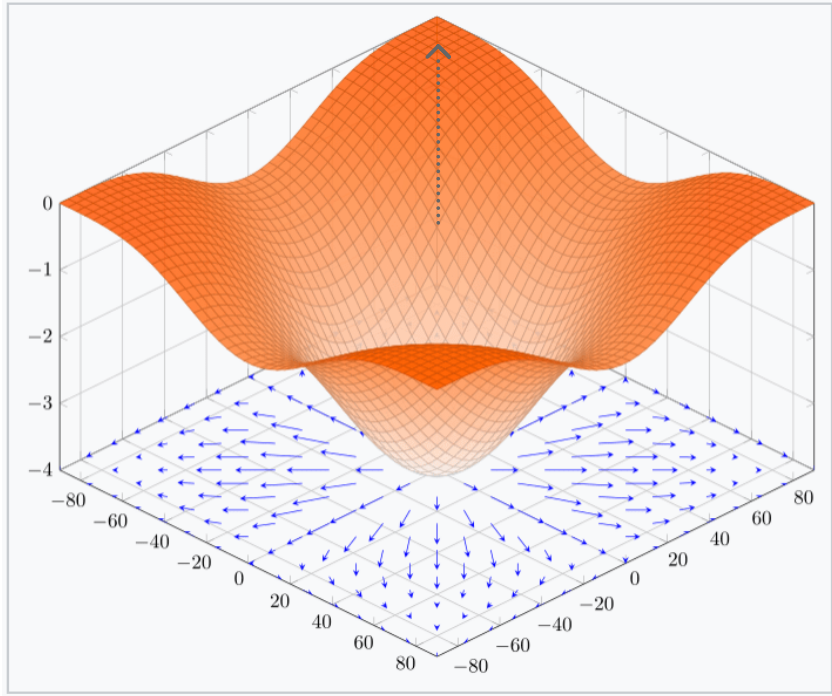
evaluating the symbolic gradient at a point gives a *numerical* gradient:

$$\nabla f(3, 2, 1) = \nabla f(x, y, z)\Big|_{(x,y,z)=(3,2,1)} = \begin{bmatrix} 6 \\ 12 \\ 1 \end{bmatrix}$$

just like a derivative can be a function or a number.

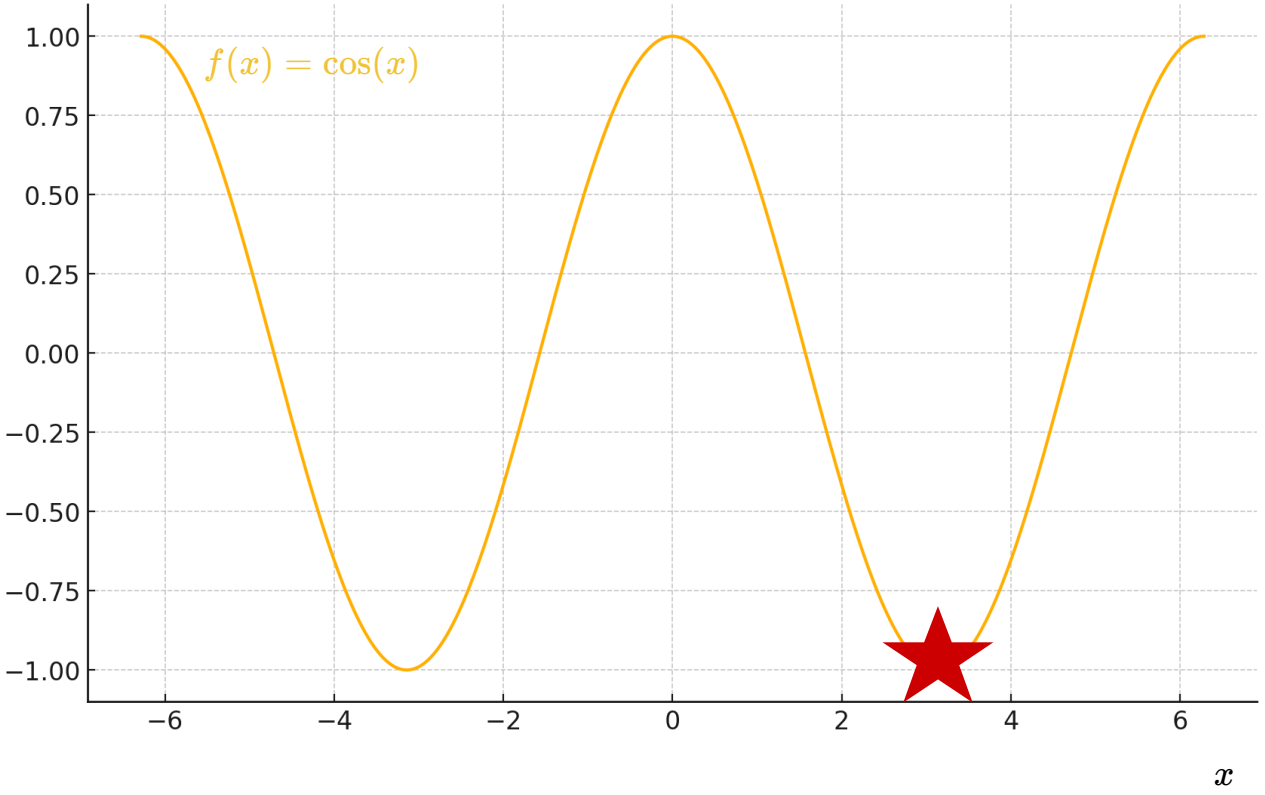
4. The gradient points in the direction of the (steepest) *increase* in the function value.

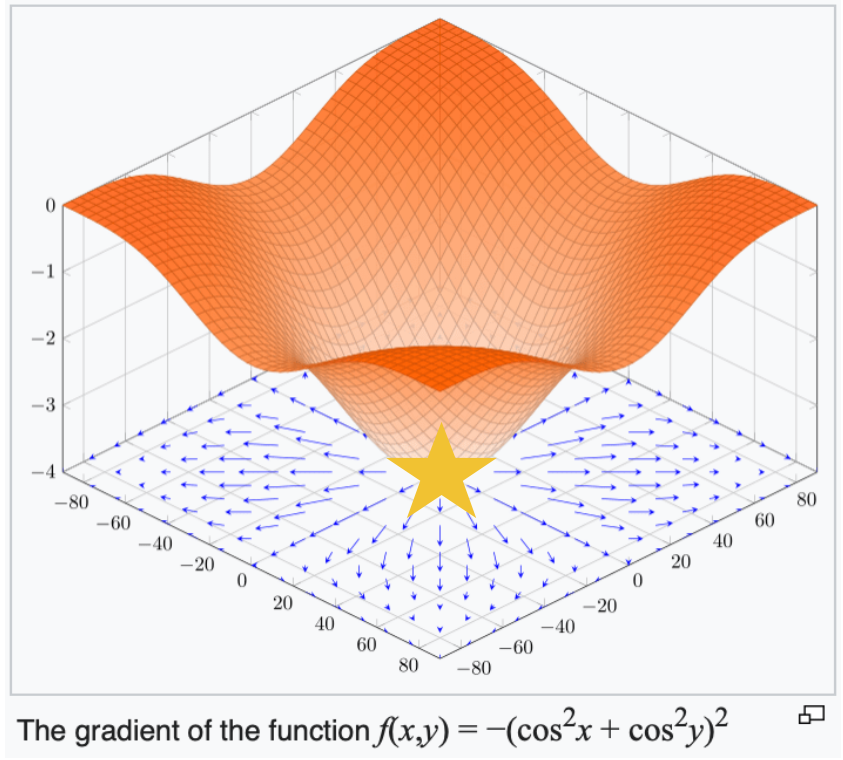




The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$

5. The gradient at the function minimizer is *necessarily* zero

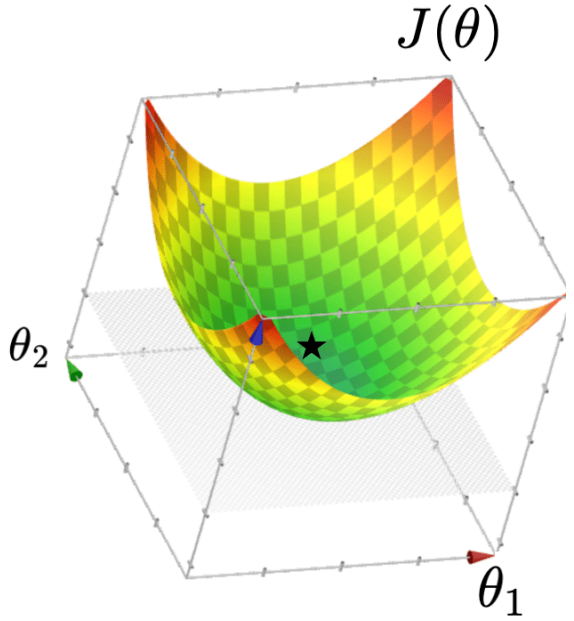




The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$

- *Typically,* $J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y)$ "curves up"
- The minimizer of $J(\theta)$ *necessarily* has a gradient zero

$$\nabla_\theta J = \begin{bmatrix} \partial J/\partial\theta_1 \\ \vdots \\ \partial J/\partial\theta_d \end{bmatrix} = \frac{2}{n}\left(X^T X\theta - X^T Y\right)$$



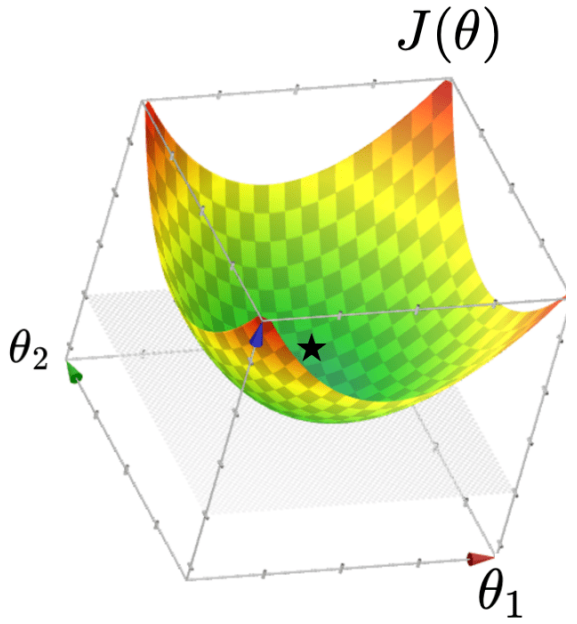
- Set the gradient $\nabla_\theta J \overset{\text{set}}{=} 0$

$$\Longrightarrow \qquad \theta^* = \left(X^\top X\right)^{-1} X^\top Y$$

Beauty of

$$\theta^* = \left(X^\top X\right)^{-1} X^\top Y$$

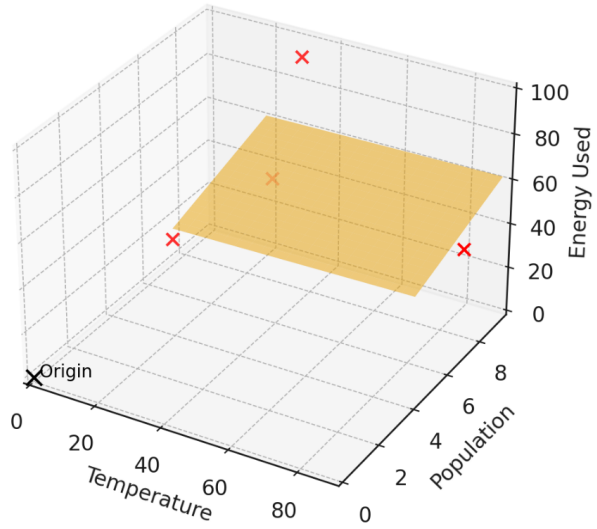
- When θ^* is well defined, it's indeed guaranteed to be the unique minimizer of $J(\theta)$
- Closed-form solution, does not feel like "training"
- The very rare case where we get such general, clean, solution with theoretical guarantee.



How to deal with θ_0?

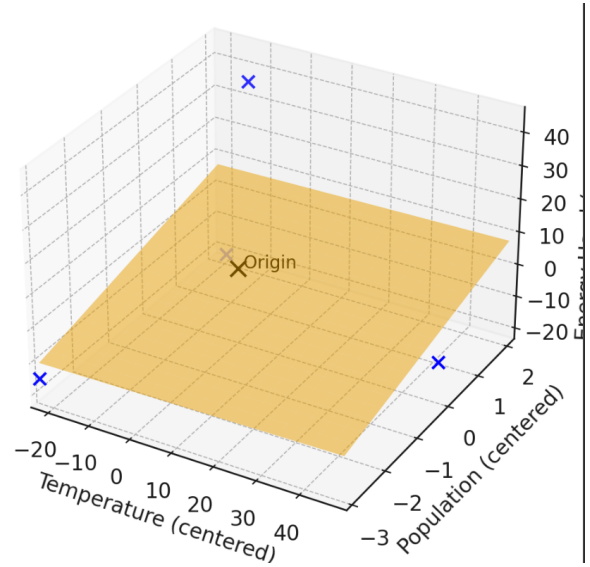
1. "center" the data

	Features		Label
City	**Temperature**	**Population**	**Energy Used**
Chicago	90	7.2	45
New York	20	9.5	32
Boston	35	8.4	100
San Diego	18	4.3	39



⇓ centering

	Features		Label
City	**Temperature**	**Population**	**Energy Used**
Chicago	49.25	-0.15	-9.00
New York	-20.75	2.15	-22.00
Boston	-5.75	1.05	46.00
San Diego	-22.75	-3.05	-15.00

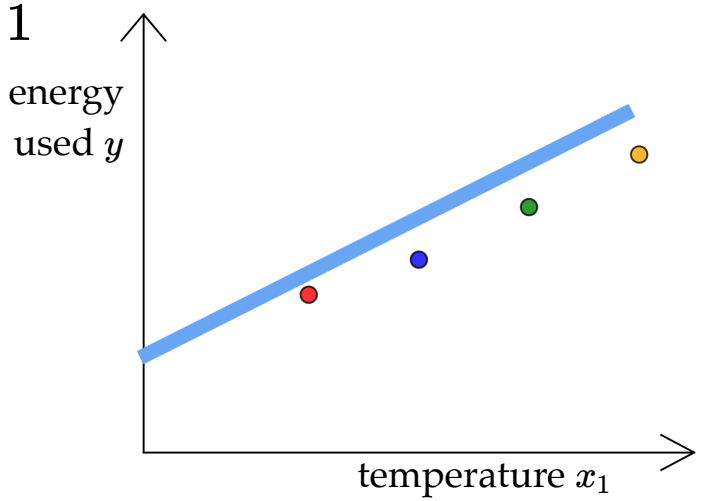


all column-wise $\Sigma = 0$

when data is centered, the optimal offset is guaranteed to be 0

How to deal with θ_0?

2. Append a "fake" feature of 1



$$h(x;\theta,\theta_0) = \theta^T x + \theta_0 = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} + \theta_0$$

$$= \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_d & \theta_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} = \theta_{\text{aug}}^T x_{\text{aug}}$$

Another way to handle offsets is to trick our model:
treat the bias as just another feature, always equal to 1.

Summary

- Terminologies:
  - supervised learning
  - training data, testing data,
  - features, label,
  - loss function, training error, testing error,
  - hypothesis, hypothesis class
  - parameters
- Ordinary least squares problem:
  - linear hypothesis class, squared loss
-
  - scalar form, matrix-vector form
  - closed-form solution

$$J(\theta) = \frac{1}{n}(X\theta - Y)^\top (X\theta - Y)$$

$$\theta^* = \left(X^\top X\right)^{-1} X^\top Y$$

Looking ahead:

$$\theta^* = \left(X^\top X\right)^{-1} X^\top Y$$

- When θ^* is well defined, it's the unique minimizer of $J(\theta)$

Now:

- When is θ^* not well defined?
- What can cause this "not well defined"?
- What happens if we are just "close to not well-defined", aka "ill-conditioned"?

we'll discuss all these next week.

https://forms.gle/Wx47AdKeNAUe4wyr9

We'd love to hear
your thoughts.

Thanks!