EECS 245, Spring 2026

LEC 9 Regression using Linear Algebra; The Gradient Vector

→ Read: Ch. 7 - Ch. 8

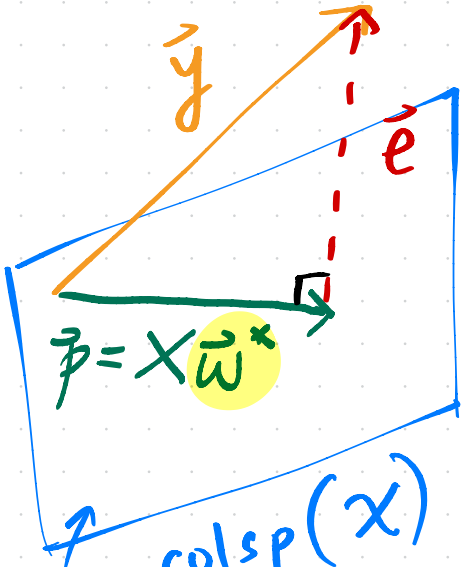
Agenda

- Idea 1: How does any of this relate to linear regression and ML?
  - Projections
  - The modeling recipe
  - The "design matrix"
  - Using multiple features
- Idea 2: Gradients
  - Calculus meets linear algebra

Announcements

- Due dates:
  - Lab 8: tomorrow
  - HW 7: Thursday
  - HW 8: Sunday. no slip days!
- Lab 7, HW 6 solutions posted
- MT 2 on Tuesday, June 9th, 1-3PM: see logistics at eecs245.org/mt2

Recap: Projections



every vector in colsp(X) is of form $X\vec{v}$, $\vec{v} \in \mathbb{R}^d$

X : $n \times d$ matrix

(d columns, each in $\mathbb{R}^n$)

$\vec{x}^{(1)}, \vec{x}^{(2)}, \ldots \vec{x}^{(d)}$

$\vec{y} \in \mathbb{R}^n$

Q: What vector in colsp(X) is closest to $\vec{y}$?

A: The vector $X\vec{w}^*$, where $\vec{y} - X\vec{w}^*$ is orthogonal to colsp(X).

$\vec{y} - X\vec{w}^*$ = error

How did we find $\vec{w}^*$?

It is the solution to the normal equation

$$X^T X \vec{w} = X^T \vec{y}$$

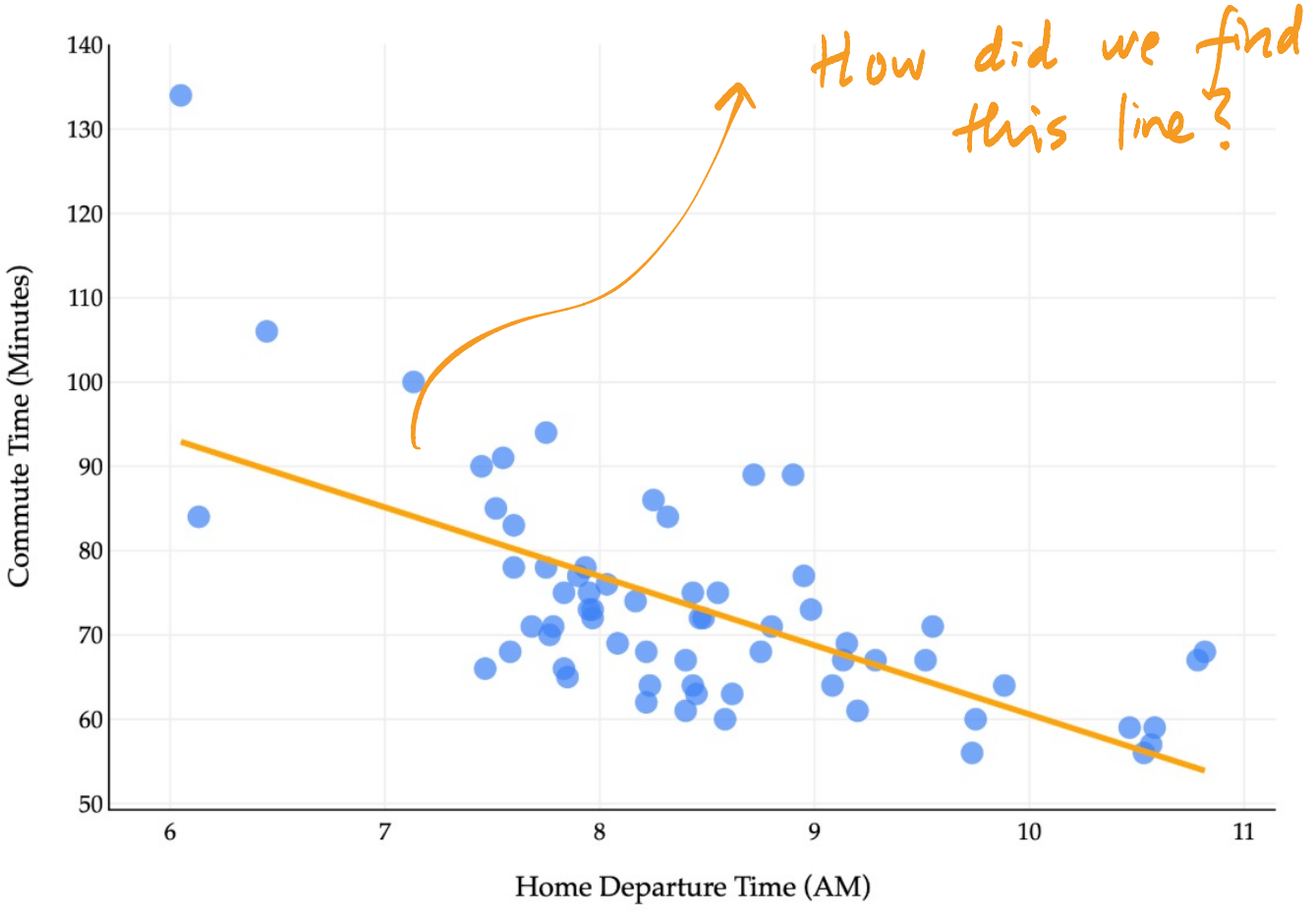
(1) If X's cols are linearly independent, then

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

unique!

(2) otherwise, there are infinitely many best $\vec{w}^*$'s!

→ see HW 7, Problem 3


How did we find this line?
Commute Time (Minutes)
140
130
120
110
100
90
80
70
60
50
6
7
8
9
10
11
Home Departure Time (AM)

Recap: Three-step modeling recipe (empirical risk minimization)

1. Choose a model

$$h(x_i) = w_0 + w_1 x_i$$

2. Choose a loss function

$$L_{sq}(y_i, h(x_i)) = \underbrace{(y_i - h(x_i))}_{\text{(actual-pred)}}^2$$

3. Minimize average loss to find optimal parameters

mean squared error

$$R_{sq}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \overbrace{(w_0 + w_1 x_i)}^{h(x_i)} \right)^2$$

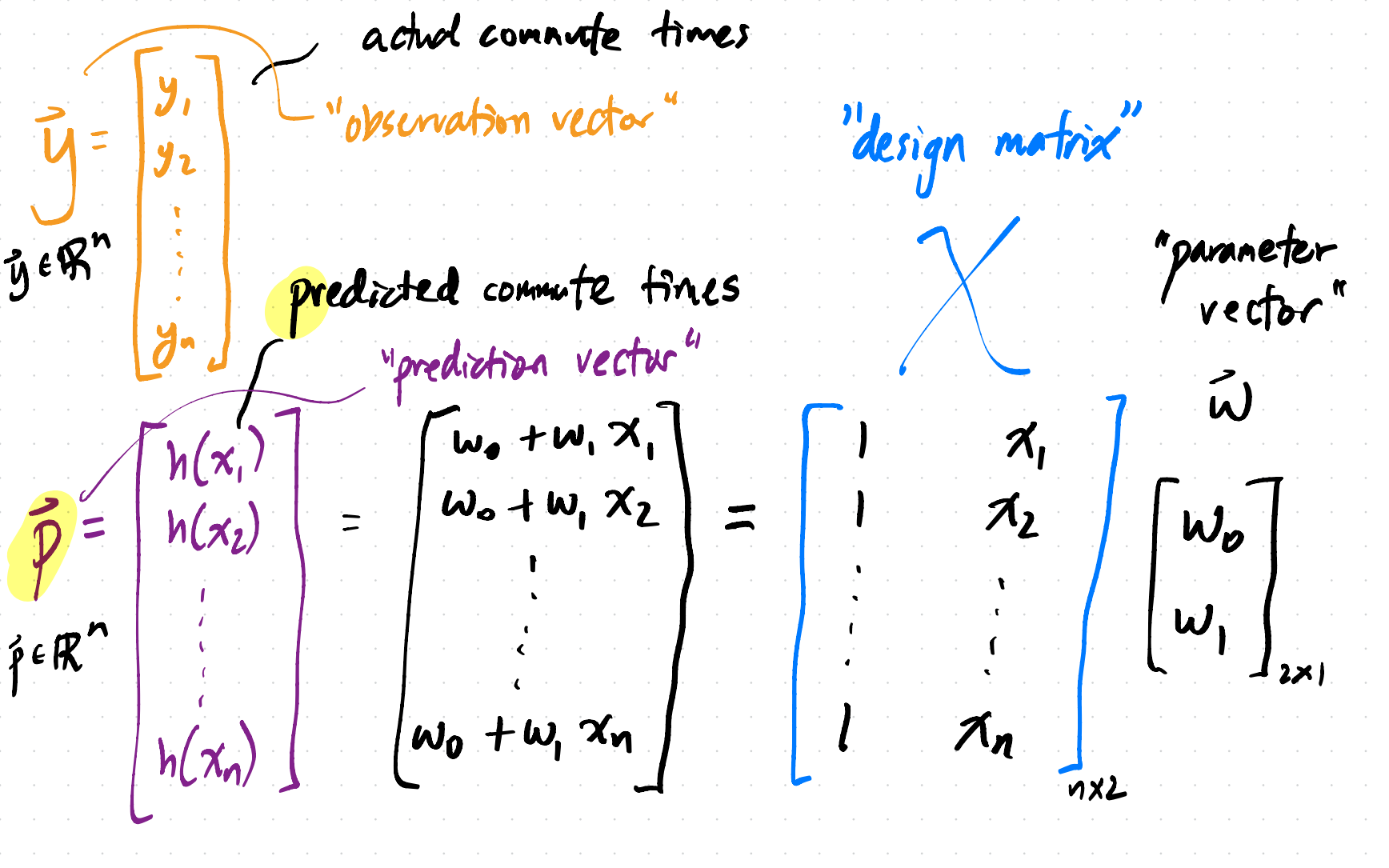
looks like error!

→ before: to find w^*'s, we took partial derivatives

→ if we had more w's (more features), this might get too complicated

+ maybe there's a way to rewrite R_{sq} using linear algebra!

→ goal: "convert" mean squared error


actual commute times
"observation vector"
$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
$\vec{y} \in \mathbb{R}^n$
"design matrix"
"parameter vector"
predicted commute times
"prediction vector"
$\vec{p} = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \\ \vdots \\ w_0 + w_1 x_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}_{n \times 2} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}_{2 \times 1}$
$\vec{p} \in \mathbb{R}^n$
$\vec{w}$

before: $R_{sq}(w_0, w_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (w_0 + w_1 x_i))^2$

now: $R_{sq}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2$

error vector

$$\begin{bmatrix} y_1 - (w_0 + w_1 x_1) \\ y_2 - (w_0 + w_1 x_2) \\ \vdots \\ y_n - (w_0 + w_1 x_n) \end{bmatrix}$$

These formulas are equivalent!

We need to find the $\vec{w}$ that minimizes

$$R_{sq}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2$$

$\vec{y} - X\vec{w}$: error vector!

How? We already solved this problem via projections!

The "best" $\vec{w}$ is the $\vec{w}$ such that

$X\vec{w}^*$ is the closest vector in colsp(X) to $\vec{y}$

→ If X's cols are linearly independent, then

$$\vec{w}^* = \begin{bmatrix} w_0^* \\ w_1^* \end{bmatrix} = (X^T X)^{-1} X^T \vec{y}$$

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y} = \begin{bmatrix} \bar{y} - w_1^* \bar{x} \\ r \frac{\sigma_y}{\sigma_x} \end{bmatrix}$$

These formulas are the same!

Q: Why is it guaranteed that the model's errors add up to 0?

A: Remember, when projecting $\vec{y}$ onto colsp(X), the error vector $\vec{e}$ is orthogonal to colsp(X).

In regression, the design matrix X has a column of all 1's, $\vec{1}$. So, it must be that

$$\vec{e} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = 0 \Rightarrow e_1 + e_2 + \ldots + e_n = 0 !$$

The point was to unlock the ability to add more features!

	departure_hour	day	day_of_month	minutes
0	10.816667	Mon	15	68.0
1	7.750000	Tue	16	94.0
2	8.450000	Mon	22	63.0
3	7.133333	Tue	23	100.0
4	9.150000	Tue	30	69.0

Let's start by adding just one new feature.

not an exponent!

$$h\left(x_i^{(1)}, x_i^{(2)}\right) = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)}$$

$x_i^{(1)}$: departure hour; $x_i^{(2)}$: day of month

departure hours

days of month

$$X_{n\times 3} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} \\ 1 & x_2^{(1)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} \end{bmatrix}$$

key idea: new feature ⇒ new col in design matrix!

How to find w_0^*, w_1^*, w_2^* ?

Just use

$$\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$$

Aside: what if some features are linear combinations of other features?

⇒ Then, $(X^TX)^{-1}$ doesn't exist

⇒ "Multicollinearity"

⇒ Doesn't impact model performance, but interpretability and uniqueness of $\vec{w}^*$

A more general notation for multiple linear regression

In general, suppose we have a dataset with

n rows, d features $\vec{x}_i$ "feature vector"

Model:

$$h(\vec{x}_i) = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + \dots + w_d x_i^{(d)}$$

$$= \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x_i^{(1)} \\ \vdots \\ x_i^{(d)} \end{bmatrix} = \vec{w} \cdot \text{Aug}(\vec{x}_i)$$

$\vec{w} \in \mathbb{R}^{d+1}$ $\quad \text{Aug}(\vec{x}_i) \in \mathbb{R}^{d+1}$

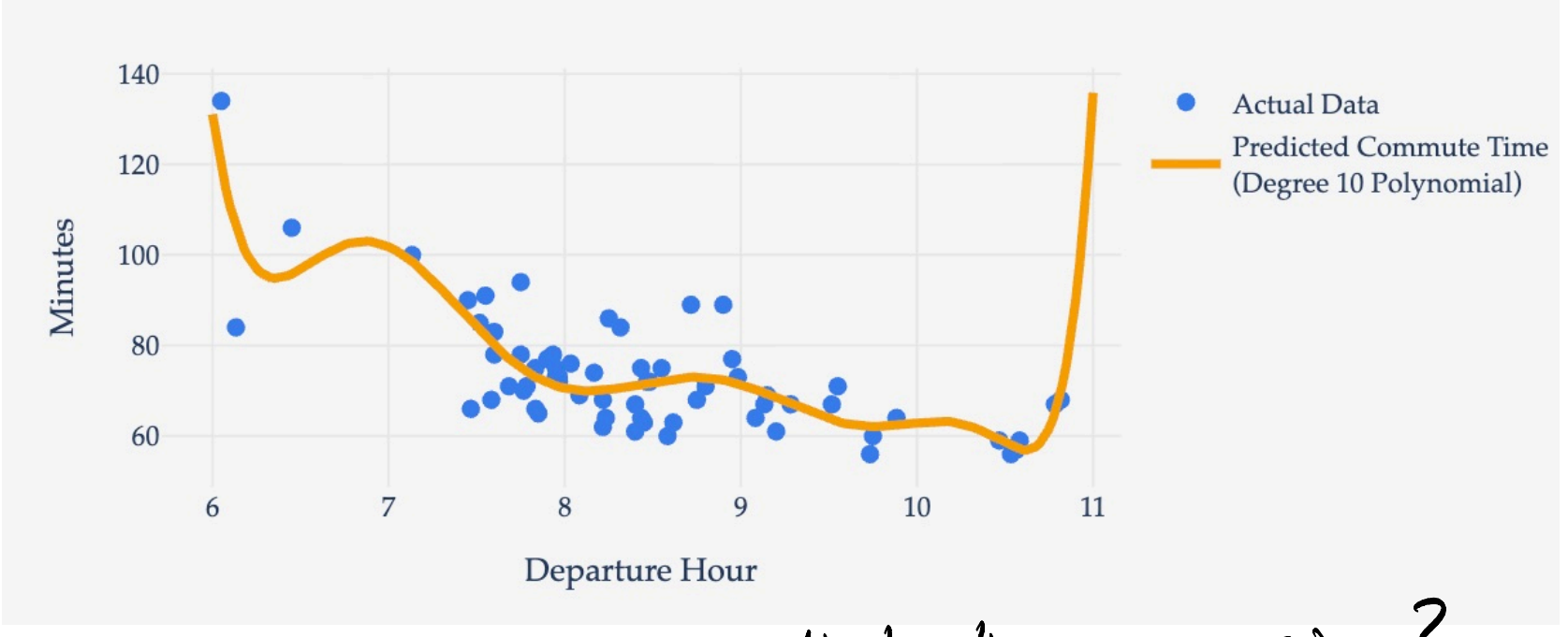
$\vec{x}_i \in \mathbb{R}^d$

General design matrix has

n rows and $d+1$ columns

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(d)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(d)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(d)} \end{bmatrix}$$

Find optimal parameters, $\vec{w}^*$, using normal equations (which is the same as minimizing mean squared error!)



Can we make this using multiple linear regression?

Sure can! Just make an $n \times 11$ design matrix.

	departure_hour	day	day_of_month	minutes
0	10.816667	Mon	15	68.0
1	7.750000	Tue	16	94.0
2	8.450000	Mon	22	63.0
3	7.133333	Tue	23	100.0
4	9.150000	Tue	30	69.0

"feature engineering"
→ new features from old

degree 3 poly, x_i = dept hour$_i$

Goal: $h(x_i) = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3$

How? Write the first 2 rows of design matrix.
No variables!

$$X = \begin{bmatrix} 1 & 10.81 & 10.81^2 & 10.81^3 \\ 1 & 7.75 & 7.75^2 & 7.75^3 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

→ Increasing polynomial degree either decreases or stays same mean squared error cannot increase on the training data.

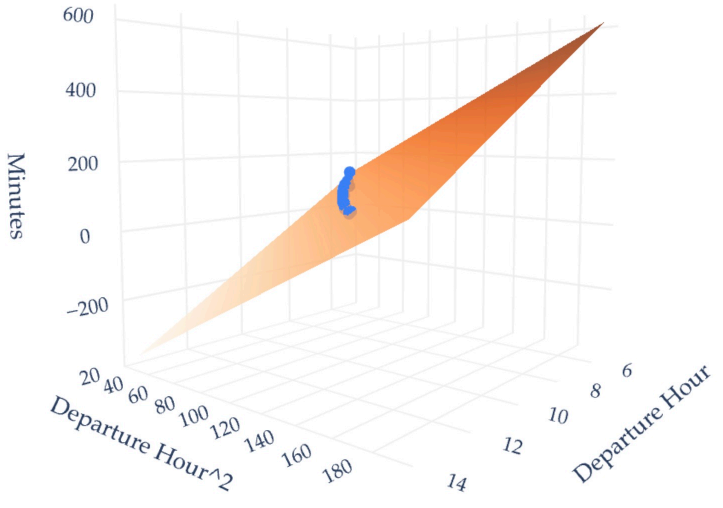
→ But, we care about models generalizing to unseen data.

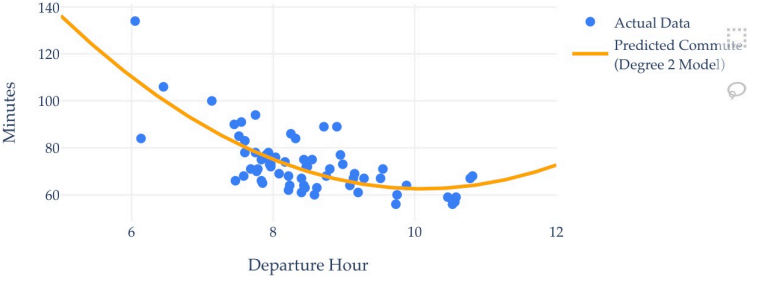
→ We need a technique for selecting model complexity

→ see HW 9/10

How is this still linear regression, despite not being a line?

⇒ If we drew the model in $\mathbb{R}^n$, it would look like a flat surface ("affine" subspace – not guaranteed to go through origin)


600
400
200
0
−200
Minutes
20
40
60
80
100
120
140
160
180
Departure Hour^2
14
12
10
8
6
Departure Hour


140
120
100
80
60
Minutes
6
8
10
12
Departure Hour
Actual Data
Predicted Commute
(Degree 2 Model)

What sorts of models can we not use linear regression for?

"linear in the parameters"

$$h(x_i) = w_0 + w_1 \sin(x_i)$$

good: $\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \sin(x_i) \end{bmatrix}$

$$h(x_i) = w_0 + w_1 \sin(w_2 x_i + w_3)$$

bad: has w's inside of sin!

need this model format: $\sum w_j \cdot \boxed{\text{no } w\text{'s}}^{(j)}$

$$h(\vec{x}_i) = w_0 + w_1 x_i^{(1)} + w_2 \frac{x_i^{(1)}}{\log x_i^{(7)}}$$

$$+ w_3 \frac{x_i^{(12)} x_i^{(1)} x_i^{(19)}}{72}$$

this is legal!

$$= \vec{w} \cdot \text{Aug}(\vec{x}_i)$$

	departure_hour	day	day_of_month	minutes
0	10.816667	Mon	15	68.0
1	7.750000	Tue	16	94.0
2	8.450000	Mon	22	63.0
3	7.133333	Tue	23	100.0
4	9.150000	Tue	30	69.0

could map Mon → 1, Tue → 2, ---

but this emphasizes later days more than earlier days

Solution: one hot encoding

day	day == Mon	day == Tue	day == Wed	day == Thu	day == Fri
Mon	1	0	0	0	0
Tue	0	1	0	0	0
Mon	1	0	0	0	0
...	...	...	...	...	...
Mon	1	0	0	0	0
Tue	0	1	0	0	0
Thu	0	0	0	1	0

5 unique values

Issue: Mon + Tue + --- + Fri = 

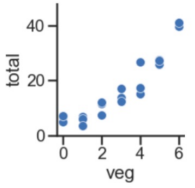
Solution: drop one column

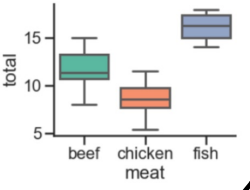
Activity 2: Chicken, Beef, or Fish?

Every week, Lauren goes to her local grocery store and buys exactly one pound of meat (either beef, fish, or chicken) but varying amounts of vegetables. We've collected a dataset containing the pounds of vegetables bought, the type of meat bought, and the total bill. Below we display the first few rows of the dataset and two plots generated using the entire (training) dataset.

Lab 8

veg	meat	total
1	beef	13
3	fish	19
2	beef	16
0	chicken	9





In each part below, we provide you with a model that predicts `total` (her total grocery bill), fit to the dataset by minimizing mean squared error. For each model, determine whether **each optimal parameter w^* is positive, negative or exactly 0**. For example, in part (iv), you'll need to provide 3 answers: one for w_0^*, one for w_1^*, and one for w_2^*.

1 $h(\vec{x}_i) = w_0$

2 $h(\vec{x}_i) = w_0 + w_1 \cdot \text{veg}_i$

3 $h(\vec{x}_i) = w_0 + w_1 \cdot \text{meat=chicken}_i$ (one hot encoded feature for chicken)

4 $h(\vec{x}_i) = w_0 + w_1 \cdot \text{meat=beef}_i + w_2 \cdot \text{meat=chicken}_i$

5 $h(\vec{x}_i) = w_0 + w_1 \cdot \text{meat=beef}_i + w_2 \cdot \text{meat=chicken}_i + w_3 \cdot \text{meat=fish}_i$

← Ch. 7

Ch. 8 →

$R_{sq} : \underbrace{\mathbb{R}^{d+1}}_{\text{domain}} \to \underbrace{\mathbb{R}}_{\text{codomain}}$

"vector-to-scalar function"

So far, we've minimized

$$R_{sq}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2$$

using linear algebra and orthogonality.

→ Could we use calculus?

Vector-to-scalar function example

$x^2 + y^2 - 3xy$

$f(\vec{x}) = x_1^2 + x_2^2 - 3x_1x_2$ $\quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$f: \mathbb{R}^2 \to \mathbb{R}$ (vector-to-scalar)

$\frac{\partial f}{\partial x_1} = 2x_1 - 3x_2$ $\quad \frac{\partial f}{\partial x_2} = 2x_2 - 3x_1$

New idea: package all partial derivatives of $f(\vec{x})$ into a vector

$$\nabla f(\vec{x}) = \begin{bmatrix} 2x_1 - 3x_2 \\ 2x_2 - 3x_1 \end{bmatrix}$$

gradient of $f(\vec{x})$

$$\nabla f(\vec{x}) = \begin{bmatrix} 2x_1 - 3x_2 \\ 2x_2 - 3x_1 \end{bmatrix}$$

e.g. if $\vec{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ($x_1 = 2$, $x_2 = 1$)

$$\nabla f\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ -4 \end{bmatrix}$$

The gradient, $\nabla f(\vec{x})$, is a vector-to-vector function!

Gradient vector

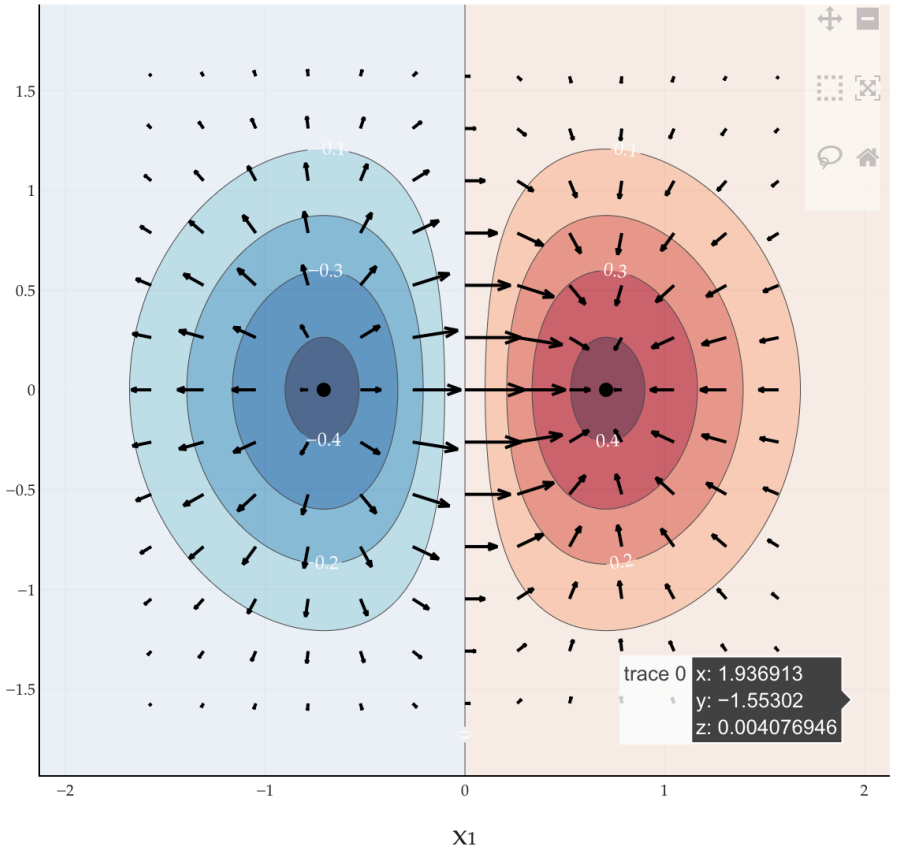
Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is a vector-to-scalar function.

The gradient of f at input $\vec{x}$ is

$$\nabla f(\vec{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

~ describes the direction of steepest ascent!

Gradient field of $f(\vec{x}) = x_1 e^{-x_1^2 - x_2^2}$



Ideally, to minimize $f(\vec{x})$, set

$$\nabla f(\vec{x}) = \vec{0}$$

"Big 3 Rules" of Chapter 8.2

(1) $f(\vec{x}) = \vec{a} \cdot \vec{x}$ where $\vec{a}, \vec{x} \in \mathbb{R}^n$

$= a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$

$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$, $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$$\nabla f(\vec{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \vec{a}$$

$\vec{a} \cdot \vec{x} = \vec{x}^T \vec{a}$

② $\vec{x} \in \mathbb{R}^n$

$$f(\vec{x}) = \|\vec{x}\|^2 = x_1^2 + x_2^2 + \cdots + x_n^2$$

$$\frac{\partial f}{\partial x_i} = 2x_i$$

$$\nabla f(\vec{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_n \end{bmatrix} = 2\vec{x}$$

Rule (3) "quadratic form"

$$f(\vec{x}) = \vec{x}^T A \vec{x}$$

$\vec{x} \in \mathbb{R}^n$

A : $n \times n$ matrix

$$\nabla f(\vec{x}) = (A + A^T)\vec{x}$$

⇒ special case: what if A is symmetric?

$A = A^T$, so then

$$f(\vec{x}) = \vec{x}^T A \vec{x} \Rightarrow \nabla f(\vec{x}) = 2A\vec{x}$$

⇒ see Lab 8