10-301/601: Introduction to Machine Learning Lecture 2 – ML as Function Approximation

Matt Gormley & Henry Chai

8/28/24

Front Matter

- Announcements:
  - HW1 released 8/26, due 9/4 (next Wednesday) at 11:59 PM
  - Two components: written and programming
    - Separate assignments on Gradescope
  - Unique policies specific to HW1:
    - Two opportunities to submit the written portion (see write-up for details)
    - Unlimited submissions to the autograder for the (really, just keep submitting until you get 100%)
    - **We will grant (almost) any extension request**

1. Taylor series ...

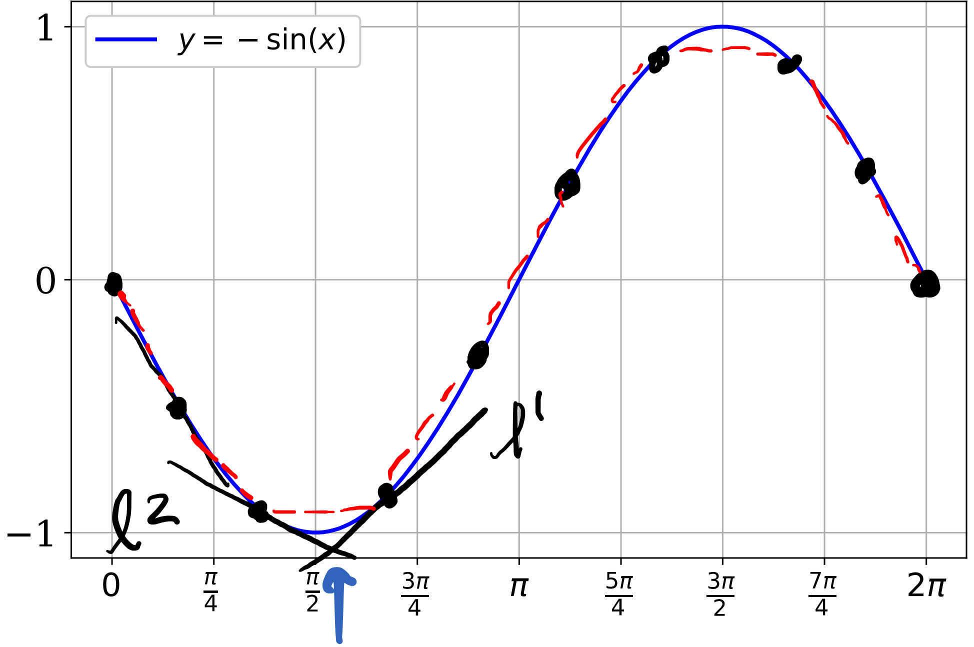
$$-\sin(x) \approx -\sin(0) + x\,\frac{-\frac{d}{dx}\sin(0)}{1!} + \dots$$

Warm-up Activity

2. Call the ref. implementation
   a) tangent lines
   b) linear interpolation

- Challenge: implement a function that computes

$$-\sin(x) \text{ for } x \in [0, 2\pi]$$

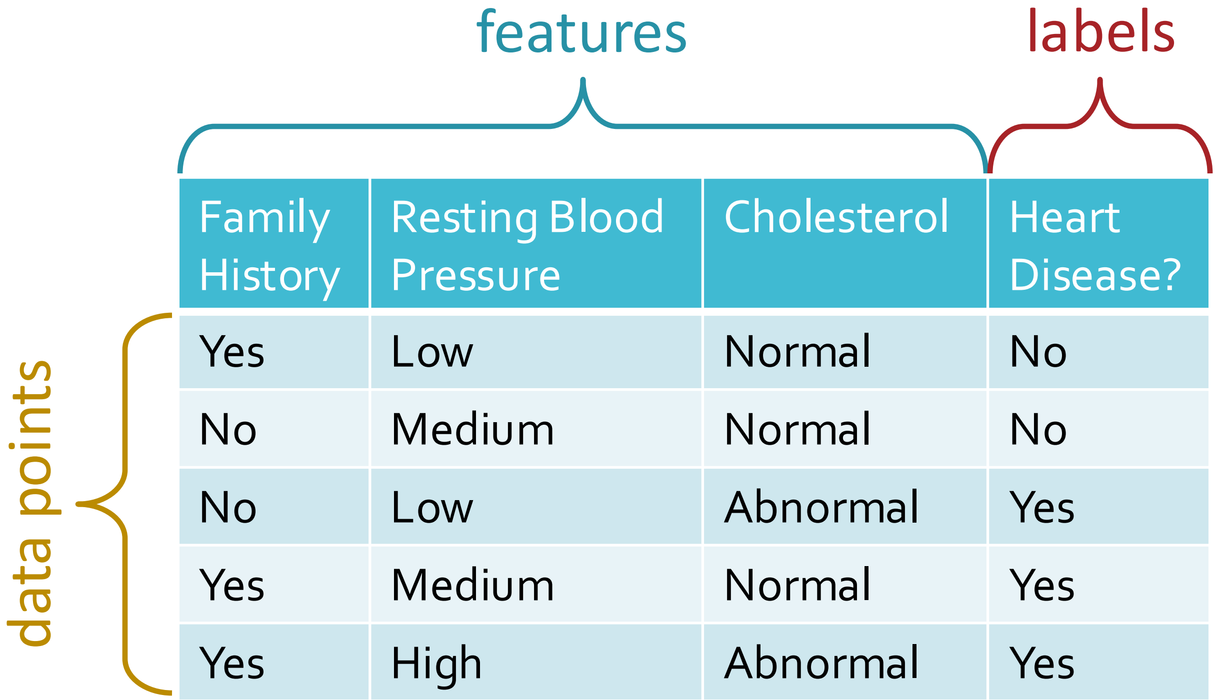


- You may not call any trigonometric functions
- You may call an existing implementation of $\sin(x)$ a few times (e.g., 10~~0~~) to check your work

Our second Machine Learning Task

- Learning to diagnose heart disease

  as a **(supervised) binary classification task**



Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

Our second Machine Learning Task

- Learning to diagnose heart disease

  as a **(supervised) binary classification task**

features

labels

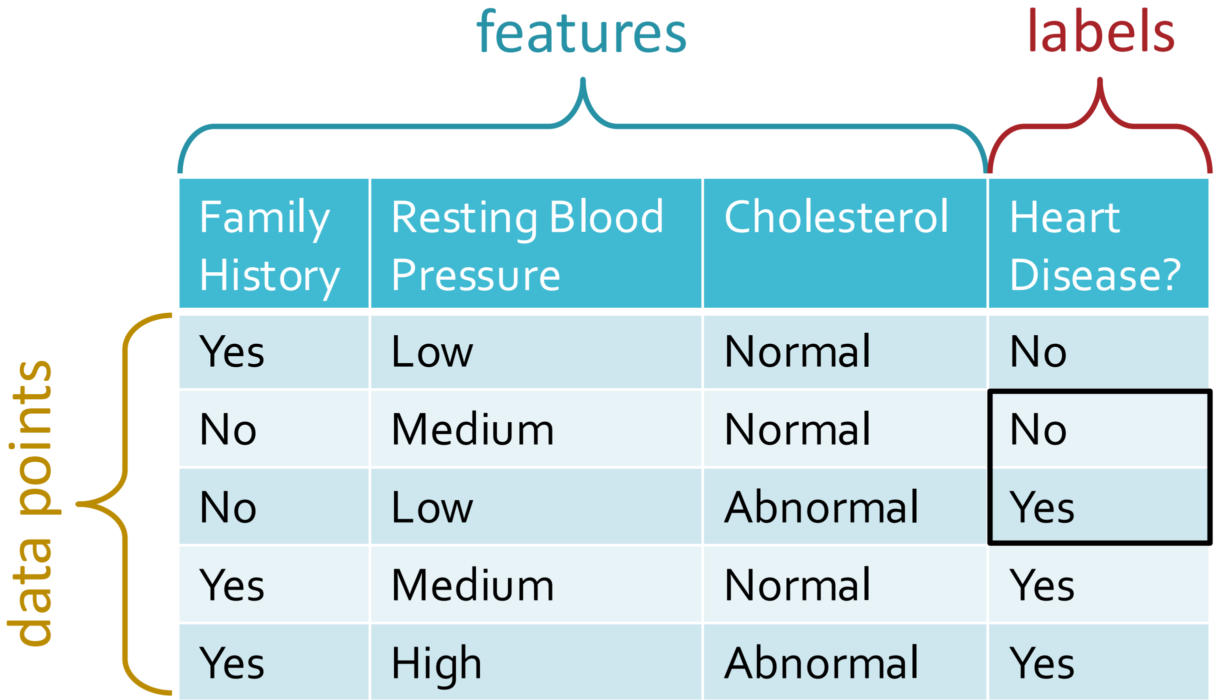
Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

data points

Our second Machine Learning Task

- Learning to diagnose heart disease

  as a **(supervised)** **<u>binary classification</u> task**



Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

Our second Machine Learning Task

- Learning to diagnose heart disease

as a **(supervised)** **<u>classification</u> task**

features labels

data points

Family History	Resting Blood Pressure	Cholesterol	Risk
Yes	Low	Normal	Low Risk
No	Medium	Normal	Low Risk
No	Low	Abnormal	Medium Risk
Yes	Medium	Normal	High Risk
Yes	High	Abnormal	High Risk

Our second Machine Learning Task

- Learning to diagnose heart disease

  as a **(supervised)** **regression task**

features

targets

data points

Family History	Resting Blood Pressure	Cholesterol	Medical Costs
Yes	Low	Normal	$0
No	Medium	Normal	$20
No	Low	Abnormal	$30
Yes	Medium	Normal	$100
Yes	High	Abnormal	$5000

Learning $-\sin(x)$

Notation

- Feature space, $\mathcal{X}$ $[0, 2\pi]$
- Label space, $\mathcal{Y}$ $\mathbb{R}$ or $[-1, 1]$
- (Unknown) Target function, $c^*: \mathcal{X} \rightarrow \mathcal{Y}$ $-\sin(x)$
- Training dataset:

$$\mathcal{D} = \{(\boldsymbol{x}^{(1)}, c^*(\boldsymbol{x}^{(1)}) = y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}) \dots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$$

$$\{(x^{(1)} = 0, y^{(1)} = -\sin(0)), (x^{(2)} = \frac{2\pi}{15}, y^{(2)} = -\sin(\frac{2\pi}{15})) \dots\}$$

- Example: $(\boldsymbol{x}^{(n)}, y^{(n)}) = (x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}, y^{(n)})$
- Hypothesis space: $\mathcal{H}$

{all possible piecewise linear functions over $[0, 2\pi]$}

{all possible fifth polynomial}

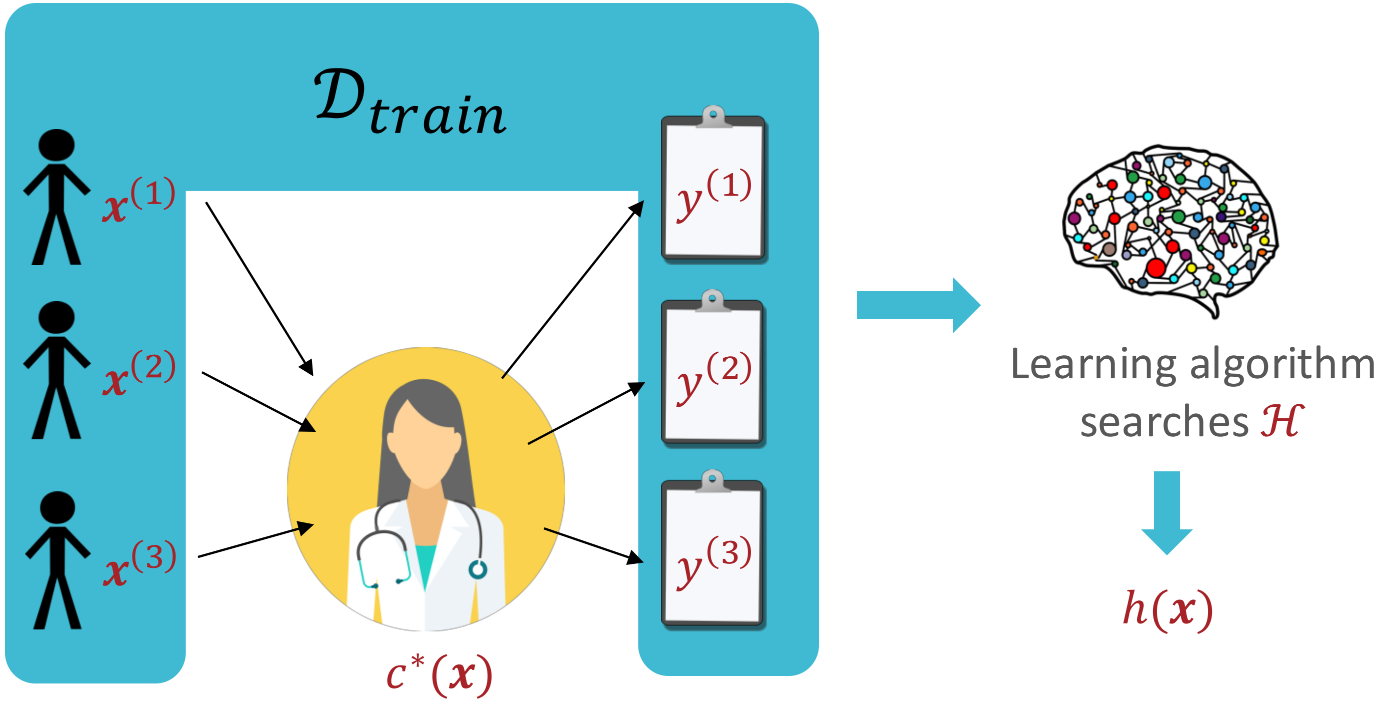
- Goal: find a classifier, $h \in \mathcal{H}$, that best approximates c^*

Notation: Example

- $N = 5$ and $D = 3$
- $\boldsymbol{x}^{(2)} = \left(x_1^{(2)} = \text{“No”}, x_2^{(2)} = \text{“Medium”}, x_3^{(2)} = \text{“Normal”}\right)$

	x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
	Yes	Low	Normal	No	Yes
$\boldsymbol{x}^{(2)}$	No	Medium	Normal	No	Yes
	No	Low	Abnormal	Yes	Yes
	Yes	Medium	Normal	Yes	Yes
	Yes	High	Abnormal	Yes	Yes

Our second Machine Learning Task



Learning algorithm searches $\mathcal{H}$

$h(\boldsymbol{x})$

Figure courtesy of Matt Gormley

Evaluation

- Loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
    - Defines how "bad" predictions, $\hat{y} = h(\boldsymbol{x})$, are compared to the true labels, $y = c^*(\boldsymbol{x})$
    - Common choices

1. Squared loss (for regression): $\ell(y, \hat{y}) = (y - \hat{y})^2$
2. Binary or 0-1 loss (for classification):

$$\ell(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$$

Evaluation

- Loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
  - Defines how "bad" predictions, $\hat{y} = h(\boldsymbol{x})$, are compared to the true labels, $y = c^*(\boldsymbol{x})$
  - Common choices

1. Squared loss (for regression): $\ell(y, \hat{y}) = (y - \hat{y})^2$
2. Binary or 0-1 loss (for classification):

$$\ell(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$$

$$\mathbb{1}(\text{logical statement, } L) = \begin{cases} 1 & \text{if } L \text{ is TRUE} \\ 0 & \text{otherwise} \end{cases}$$

- Error rate:

$$err(h, \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(y^{(n)} \neq \hat{y}^{(n)}\right)$$

$$\uparrow \; h(x^{(n)}) = \hat{y}^{(n)}$$

$$D = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N}$$

Different Kinds of Error

- Training error rate = $err(h, \mathcal{D}_{train})$

- Test error rate = $err(h, \mathcal{D}_{test})$

- True error rate = $err(h)$

  = the error rate of h on all possible examples

  - In machine learning, this is the quantity that we care about but, in most cases, it is unknowable.

Our second Machine Learning Task

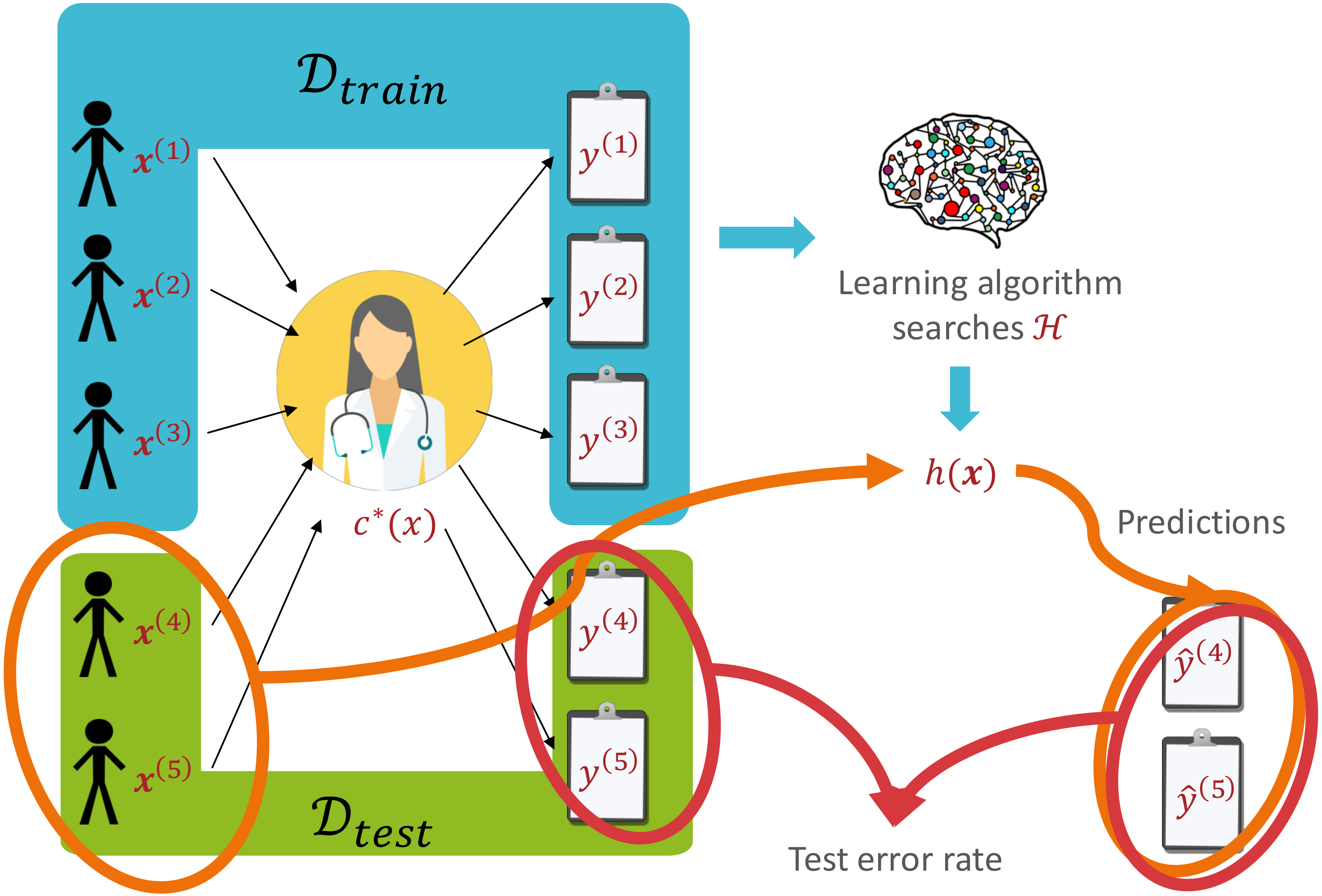


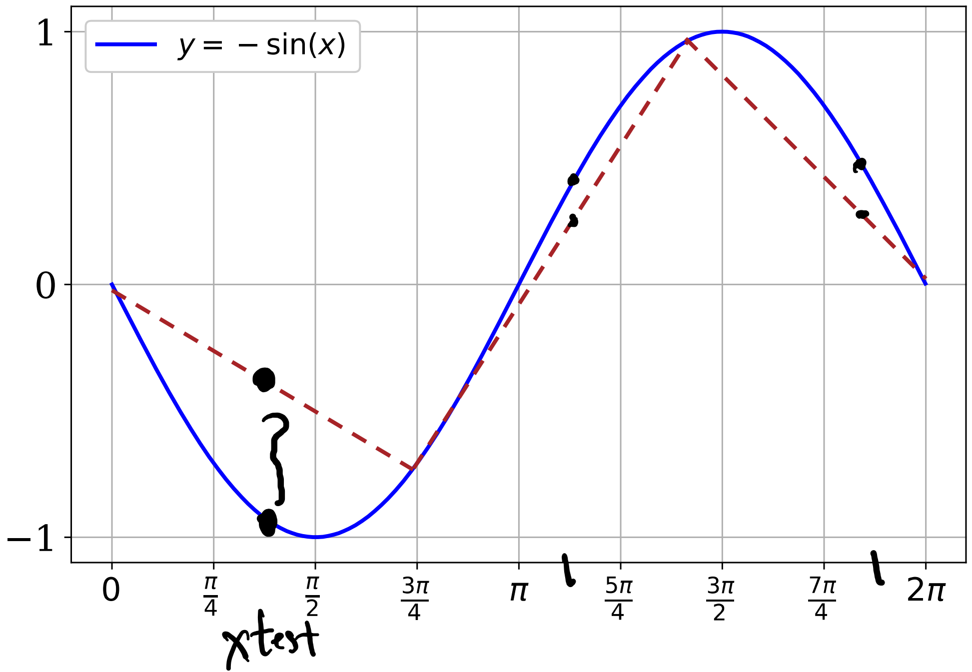
Figure courtesy of Matt Gormley

$$\ell(y, \hat{y}) = \left(h(x^{test}) - (-\sin(x^{test})) \right)^2$$

Warm-up Activity

- Challenge: implement a function that computes

  $-\sin(x)$ for $x \in [0, 2\pi]$



- You may not call any trigonometric functions
- You may call an existing implementation of $\sin(x)$ a few times (e.g., 100) to check your work

Recall: Our first Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Majority vote classifier: always predict the most common label in the **training** dataset

features

labels

data points

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	Yes
No	Low	Abnormal	Yes	Yes
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

Our first Machine Learning Classifier: Pseudocode

classifier specific

```
def train($D_{train}$):
        store $v$ = mode($y^{(1)}, y^{(2)}, \ldots, y^{(N)}$)

def h($\vec{x}$):
        return $v$
```

general

```
def predict($D_{test}$):
    for ($\vec{x}^{(n)}, y^{(n)}$) $\in D_{test}$:
        $\hat{y}^{(n)} = h(\vec{x}^{(n)})$
        $\vdots$
```

Test your understanding

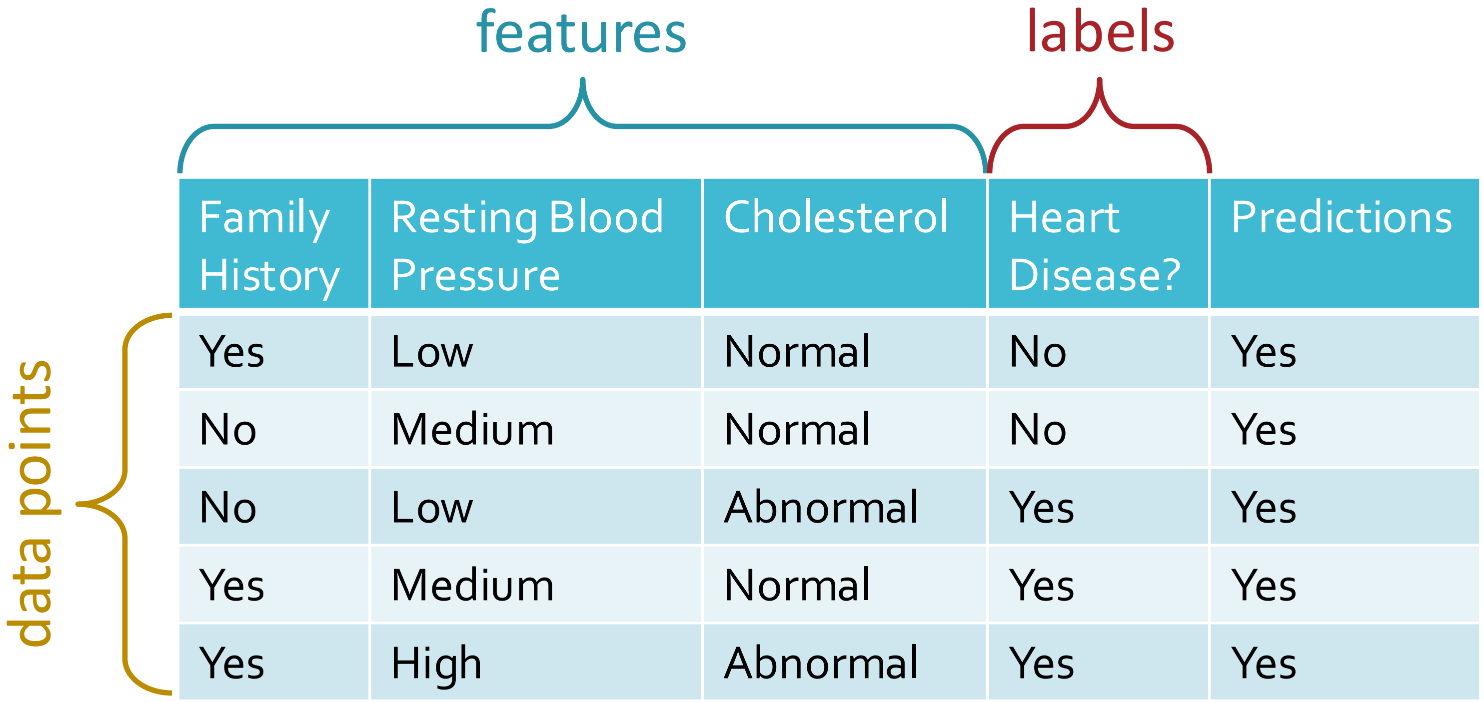
x_1	x_2	y	
1	0	-	X
1	0	-	X
1	0	+	
1	0	+	
1	1	+	
1	1	+	
1	1	+	
1	1	+	

- What is the **training error** of the **majority vote classifier** on this dataset?

$\frac{2}{8} = 1/4$

Our first Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Majority vote classifier: always predict the most common label in the **training** dataset



Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	Yes
No	Low	Abnormal	Yes	Yes
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

- This classifier completely ignores the features…

Our second Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

Our second Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	Predictions
Yes	Low	Normal	No	No
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	Yes
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

- The training error rate is 0!

Our second Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	Predictions
Yes	Low	Normal	No	No
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	Yes
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

- The training error rate is 0…

Is the memorizer ”learning”?

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?	Predictions
Yes	Low	Normal	No	No
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	Yes
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

- The training error rate is 0…

Our second Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote
- The memorizer (typically) does not **generalize** well, i.e., it does not perform well on unseen data points
- In some sense, good generalization, i.e., the ability to make accurate predictions given a small training dataset, is the whole point of machine learning!

Our second Machine Learning Classifier: Pseudocode

```
def train($D_{train}$):
    store $D_{train}$

def $h(\vec{x})$:
    if $\exists\ (\vec{x}^{(n)}, y^{(n)}) \in D_{train}$ s.t. $\vec{x}^{(n)} = \vec{x}$:
        return $y^{(n)}$
    else
        return mode$(y^{(1)}, \ldots, y^{(N)})$
```

Our third Machine Learning Classifier

- Alright, let’s actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump: based on a single feature, x_d, predict the most common label in the **training** dataset among all data points that have the same value for x_d

Our third Machine Learning Classifier

- Alright, let’s actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x_1', \dots, x_D') = \begin{cases} ??? & \boxed{\text{if } x_1' = \text{“Yes”}} \\ ??? & \text{otherwise} \end{cases}$$

Our third Machine Learning Classifier

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x'_1, \ldots, x'_D) = \begin{cases} \text{“Yes” if } x'_1 = \text{“Yes”} \\ ??? \boxed{\text{ otherwise }} \end{cases}$$

Our third Machine Learning Classifier

- Alright, let’s actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	No
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes

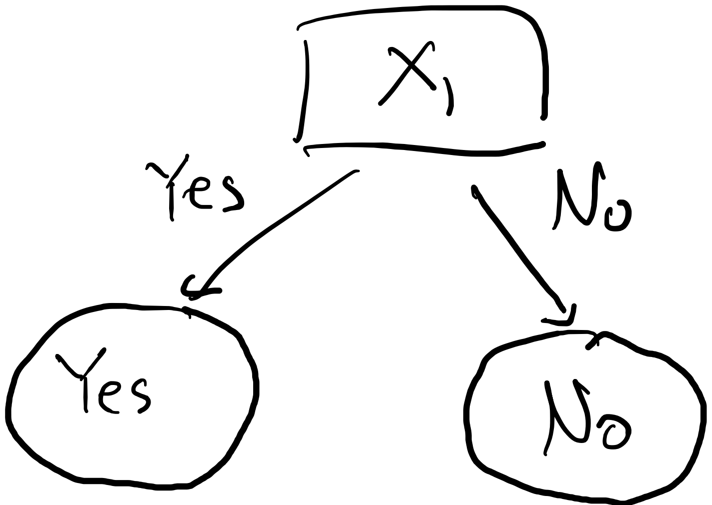
- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x_1', \ldots, x_D') = \begin{cases} \text{“Yes” if } x_1' = \text{“Yes”} \\ \text{“No” otherwise} \end{cases}$$

Our third Machine Learning Classifier

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	No
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes



Decision Stumps: Pseudocode

```
def train($D_{train}$):
    1. pick a feature to split on, $x_d$ (???)
    2. split $D_{train}$ on $x_d$
       for v in $V(x_d) = \{$all possible values $x_d$ takes on$\}$:
           $D_v = \{(\vec{x}^{(n)}, y^{(n)}) \in D_{train} \mid x_d^{(n)} = v\}$
    3. Compute the prediction for each subset
       for v in $V(x_d)$:
           store $\hat{y}_v$ = mode(labels in $D_v$)
def $h(\vec{x})$:
    for v in $V(x_d)$:
        if $x_d = v$: return $\hat{y}_v$
```

Decision Stumps: Questions

1. How can we pick which feature to split on?

2. Why stop at just one feature? Don't!