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

Henry Chai & Matt Gormley

8/31/22

Q & A

- In Lecture 1, why did we use the term experience instead of just data?
- Because our concern isn't just the data itself, but also where the data comes from (e.g., an agent interacting with the world vs. knowledge from a book). As well, the word experience better aligns with the notion of what humans require in order to learn.

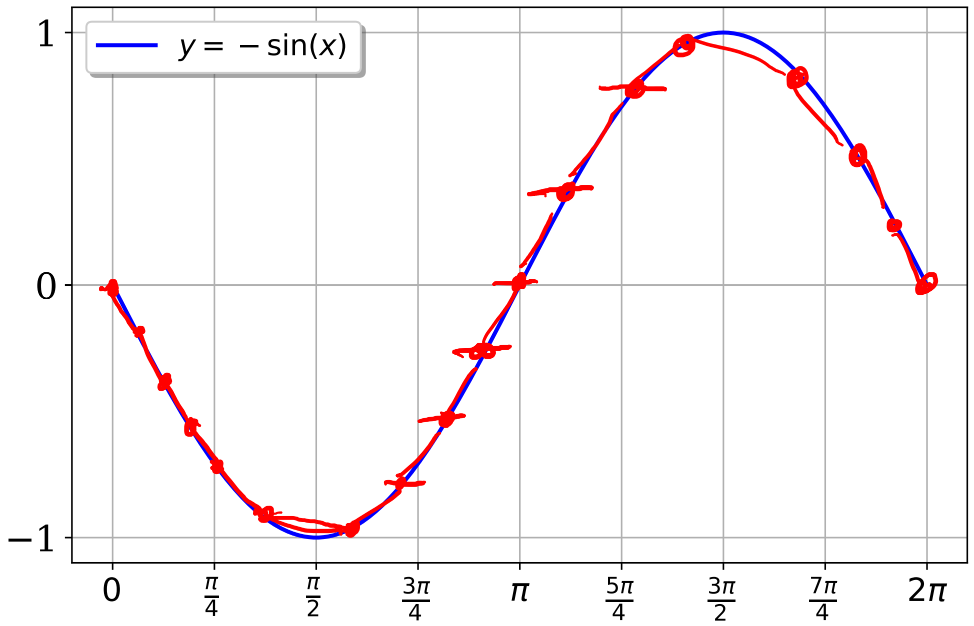
Front Matter

- Announcements:
  - HW1 released 8/29, due 9/7 at 11:59 PM
  - Two components: written and programming
    - Separate submissions on Gradescope
  - Unique policies specific to HW1:
    - Two submissions for the written portion (see write-up for details)
    - Unlimited submissions for the programming portion (really, just keep submitting until you get 100%)
    - We will grant (almost) any extension request

Function Approximation: Example

- Challenge: implement a function that computes

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



– Taylor series
– use the like classical definition

- 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

Our 2nd Machine Learning Task

- Learning to diagnose heart disease

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

features

labels

examples

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 2nd Machine Learning Task

- Learning to diagnose heart disease

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

features | labels

examples

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 2nd Machine Learning Task

- Learning to diagnose heart disease

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

features — labels

examples

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 2nd Machine Learning Task

- Learning to diagnose heart disease

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

	features			labels
	Family History	Resting Blood Pressure	Cholesterol	Risk
examples	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 2nd Machine Learning Task

- Learning to diagnose heart disease

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

features | targets

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

examples

Our 2^{nd} 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

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

examples

Our 2nd 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

features

labels

examples

Example: $-\sin(x)$

Notation

- Feature space, $\mathcal{X}$ $[0, 2\pi]$
- Label space, $\mathcal{Y}$ $\mathbb{R}$, $[-1, 1]$
- (Unknown) Target function, $c^*: \mathcal{X} \to \mathcal{Y}$ function definition

  $-\sin(x)$
- Training dataset: – calling the reference implementation

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

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

  $(x, -\sin(x))$
- Hypothesis space: $\mathcal{H}$

  – piecewise linear functions – step functions
- Goal: find a classifier, $h \in \mathcal{H}$, that <u>best approximates c^*</u>

Our 2nd Machine Learning Task

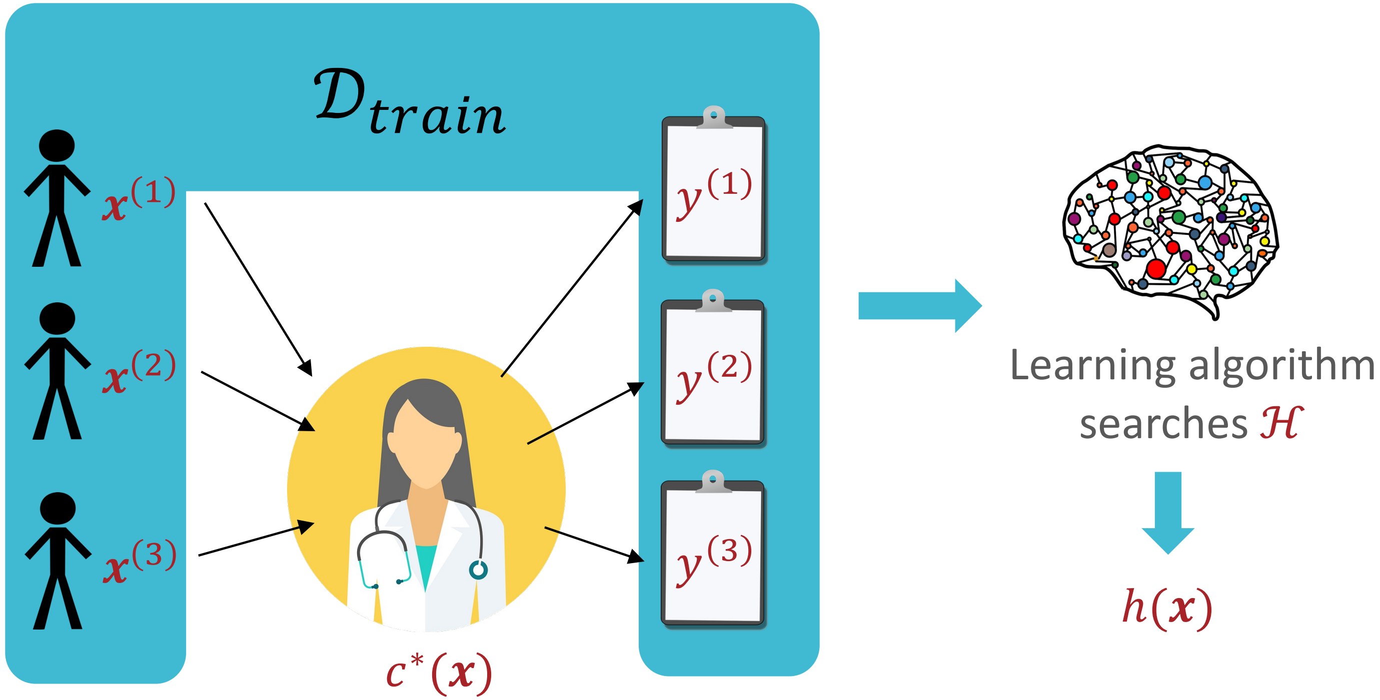


Figure courtesy of Matt Gormley

Our 2nd Machine Learning Classifier

- 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

$\boldsymbol{x}^{(2)}$

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

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

absolute loss : $|y - \hat{y}|$

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):

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

indicator function

- Error rate:

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

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 2nd Machine Learning Task

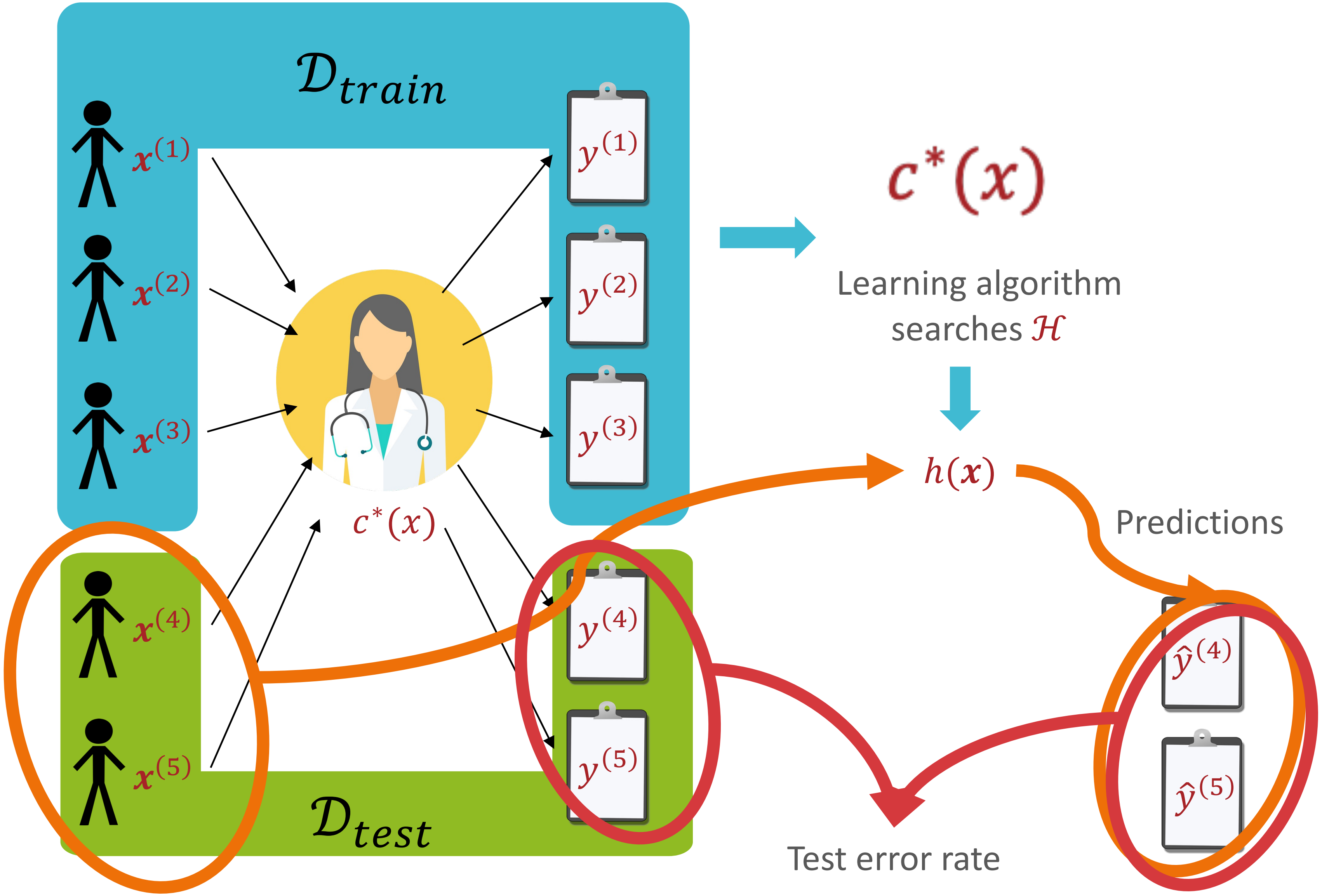


Figure courtesy of Matt Gormley

Our 2nd Machine Learning Classifier

- Majority vote classifier:

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

def predict(x'):
    return v

Test your understanding

x_1	x_2	y
1	0	-
1	0	-
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?

Our 2nd 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

labels

examples

Heart Disease?	Predictions
No	Yes
No	Yes
Yes	Yes
Yes	Yes
Yes	Yes

- This classifier completely ignores the features…

Our 3rd Machine Learning Classifier

- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict a random label.

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 3rd Machine Learning Classifier

- Memorizer:

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

def predict(x'):
    if $\exists\ x^{(n)} \in D_{train}$ s.t. $x^{(n)} == x'$:
        return $y^{(n)}$
    else:
        return $y \in \mathcal{Y}$ at random

Our 3rd Machine Learning Classifier

- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict a random label.

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 always 0!

No	Yes
- bad at unseen data	- By the book

Is the Memorizer learning?

- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict a random label.

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 always 0!

Our 3rd Machine Learning Classifier

- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict a random label.
- 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 4th 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 4th Machine Learning Classifier: Example

- 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 4th Machine Learning Classifier: Example

- 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:

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

Our 4th Machine Learning Classifier: Example

- 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

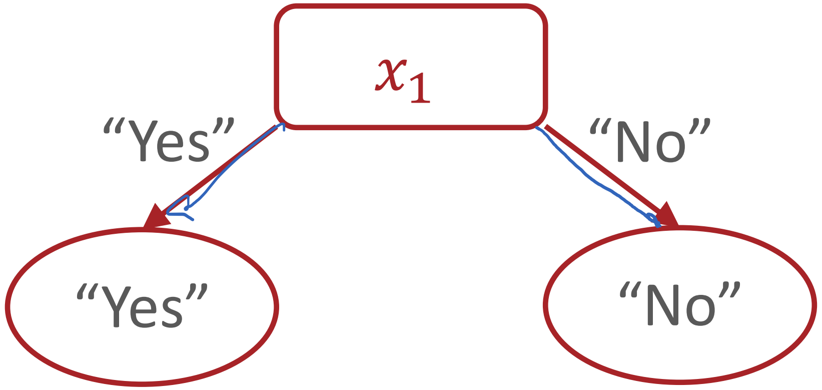
- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x'_1, \dots, x'_D) = \begin{cases} \text{“Yes” if } x'_1 = \text{“Yes”} \\ \text{“No” otherwise} \end{cases}$$

Our 4th Machine Learning Classifier: Example

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

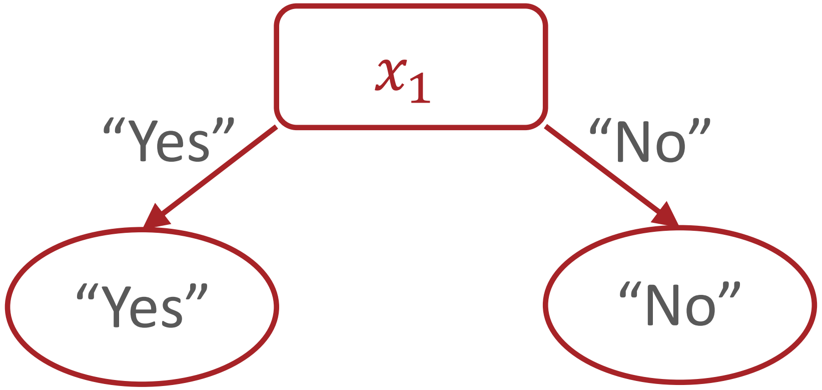
x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes



Our 4th Machine Learning Classifier: Example

- 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, x_d (???)
2. split D_{train} according to x_d
   - for v in $V(x_d)$, all possible values of x_d:
     $D_v = \{(x^{(n)}, y^{(n)}) \in D_{train} \mid x_d^{(n)} = v\}$
3. Compute some majority votes
   - for v in $V(x_d)$:
     store $\hat{y}_v$ = majority_vote(D_v)

def predict(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?