Foundations of Machine Learning

Standard Framework: Players

- Distributions & data
- Outcomes & predictions
- Models & model classes
- Training & testing
- Learning = generalization

Distributions & Data

- Instance/input space X
- E.g. loan apps, images, med. records, ...
- Outcome/output space Y
- E.g. loan status, cats, diagnosis, ...
- Examples $\langle x, y \rangle : x \in X, y \in Y$
- Distribution/population P over possible $\langle x, y \rangle$
- All we see is a sample S:
  $S = \{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots, \langle x_n, y_n \rangle\}$
  drawn from P

Important Remark: x

- Often we **choose** what info or "features" are in x
- Usually info **predictive** of y
- E.g. financial/employment history in lending
- Some info may **favor** or disfavor groups
- E.g. clubs or jobs
- Even "obvious" info may encode **biases**
- E.g. pixels in images

Important Remark: y

- Sometimes the outcomes y are objectively measurable and "unbiased"
- E.g. S&P 500 up/down; winner of last night's game; inches of snow @ PHL
- Sometimes "ground truth" more subjective
- E.g. are these two faces of the same person?
- Sometimes our y's are biased or filtered
- E.g. only know GPAs of students we admit to Penn

Models & Predictions

- Our goal is to use sample S to make predictions $\hat{y}$ for y in new $\langle x, y \rangle$
- Our predictions will be in form of a model or function $\hat{y} = h(x)$
- We allow that P might be arbitrary but h will be "simple" by necessity

Error: Train & True

- Training error of h on S:

$$\hat{\varepsilon}_S(h) = \hat{\varepsilon}(h) \triangleq \frac{1}{n}\sum_{i=1}^{n} I[h(x_i) \neq y_i]$$

("indicator function")

= fraction of mistakes of h on S (classification setting)

- True/test error of h on P:

$$\varepsilon_P(h) = \varepsilon(h) \triangleq \mathop{E}_{\langle x,y\rangle \sim P} [I[h(x) \neq y]]$$

= probability h makes a mistake on $\langle x,y\rangle \sim P$

Model Classes

- Even if P is arbitrarily complicated, we must build "simple" models
- Choose $h \in H$ where H "simple"
- E.g. H = decision trees,
  H = linear classifiers,
  H = neural networks, ...
- Hope that chosen H has an $h \in H$ with small true error $\mathcal{E}(h)$

Standard ML Workflow

- Gather sample S from P
- Choose/design model class H
- Use algo/heuristic to find $h \in H$ with small $\hat{\varepsilon}(h)$
- Estimate $\varepsilon(h)$ on new data also from P
- (Repeat...)

What justifies this methodology?

Example:

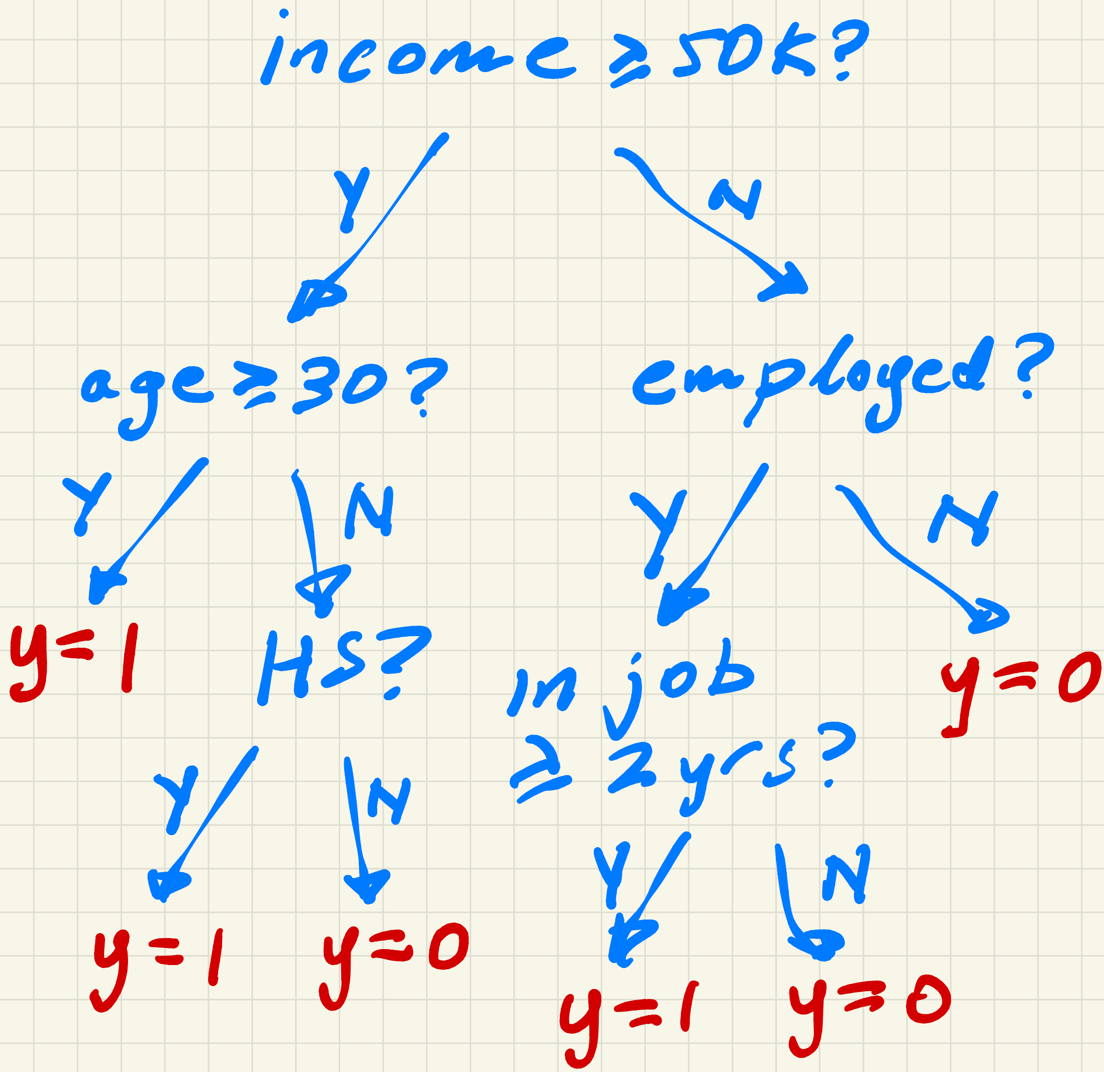
Decision Trees for Lending

- **Goal:** predict who will repay a loan
- So $y = \begin{cases} 1 & \text{will repay} \\ 0 & \text{won't repay} \end{cases}$
- Then x might contain:
  - + credit history
  - + current salary
  - + employment history
  - + savings
  - + age, gender, race...
  - + social media activity?
  - + religion? politics?
  - + GPA?

⋮

- P is some pop./distribution over $\langle x, y \rangle$ pairs
- We have sample $S = \{\langle x_1, y_1 \rangle, \ldots \langle x_n, y_n \rangle\}$ (assume unbiased/unfiltered)
- Let's build a (small) decision tree to predict y from x

Example: a depth 3 DT



- So e.g. could let H be class of all possible depth 3 (or whatever) decision trees

- Design algo to find best DT in H on S (usually hard)

Example:

"Deep Learning"

in 4 slides

Goal: Determine if there is a cat in a photo.

So x's are digital images, e.g. $256 \times 256 \approx 65K$

Each pixel has R, G, B

And $y\text{'s} = \begin{cases} 1 & \text{if cat} \\ 0 & \text{if no cat} \end{cases}$

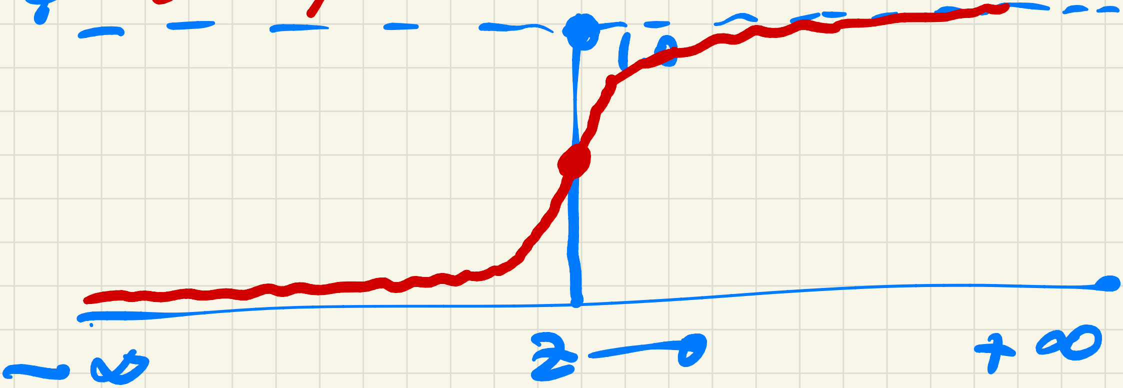
E.g. scraped from web, crowdsourced/captions

Treshold Gates

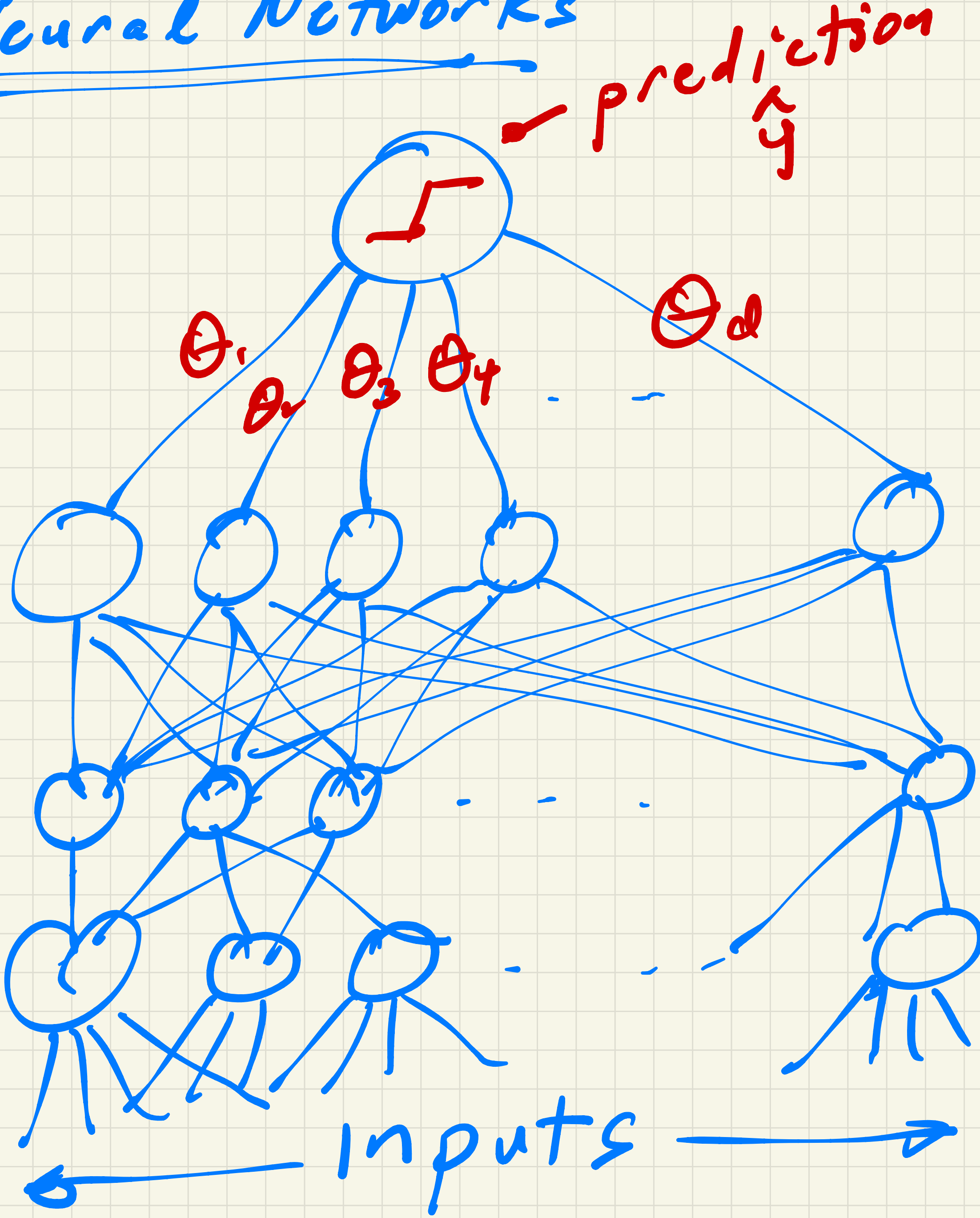
- Take linear sum of inputs, run through a nonlinear function
- I.e. if inputs are $x_1, x_2, \ldots x_d \in \mathbb{R}$ then let

$$z = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

& output $= f(z)$:



Neural Networks



- If we use squared error:

$$(\hat{y} - y)^2$$

instead of 0/1 error, whole mess is differentiable in the parameters θ.

- Try to minimize error by gradient descent. (again "hard")

OK. But again,

why is this a good idea?

Fundamental Theorem of ML

- No matter what P looks like...
- ... and for any "reasonable" H...
- ... if we have "enough" data S...
- ... then for every $h \in H$, we have

$$\hat{\mathcal{E}}_S(h) \approx \mathcal{E}_P(h)$$

∴ minimizing error on data ≈ minimizing true (future) error

"Enough" data: n large compared to complexity of H.

ML Research

- Design of natural/expressive $\mathcal{H}$
- Design of fast algos for (approx) minimizing $\hat{\mathcal{E}}(h)$ in $\mathcal{H}$
- Refinements of fundamental theorem
- Experiments

No mention (yet) of:

fairness, privacy, explainability, safety, robustness...

Up Next:

Bias and Discrimination in ML (and "solutions")