CAS CS 440
Lec 19

Logic VIII & Supervised Learning I

1. Review
    a. Modern AI → AI has to figure out on its own
    b. Humans do less work and allows the model to learn on its own
2. Lifting & Unification
    a. Lifted inference rules
        i. Find substitutions that make different sentences look identical
        ii. Called unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

    b. How should this algorithm behave?
        i. ASKVARS(Knows(John, x))
        ii. Find all sentences in KB that unify with

$$\text{UNIFY}(Knows(John, x),\ Knows(John, Jane)) = \{x/Jane\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Bill)) = \{x/Bill, y/John\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$$

    c. $\text{UNIFY}(Knows(John, x),\ Knows(x_{17}, Elizabeth)) = \{x/Elizabeth, x_{17}/John\}$

$$\text{UNIFY}(Knows(John, x),\ Knows(x, Elizabeth)) = fail$$

x cannot take on two values at the same time

Problem! What if the rule is $\forall x\ \ Knows(x,\ Elizabeth)$?
two rules just happen to use the same variable

Standardize apart sentences

    d.
3. Most General Unifier
    a. $\text{UNIFY}(Knows(John, x), Knows(y, z))$
    b. Could return {y/John, x/z} or {y/John, x/John, z/John}
        i. 2nd result can be obtained from the first (with additional substitution)
        ii. 1st result (unifier) is more general than 2nd
            1. Places fewer restrictions on variables
        iii. There is always a single most general unifier for very unifiable pair of sentences

```
function UNIFY-VAR(var, x, θ) returns a substitution
  if {var/val} ∈ θ then return UNIFY(val, x, θ)
  else if {x/val} ∈ θ then return UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) then return failure
  else return add {var/x} to θ
```

S(x) cant unify with S(S(x))

c.

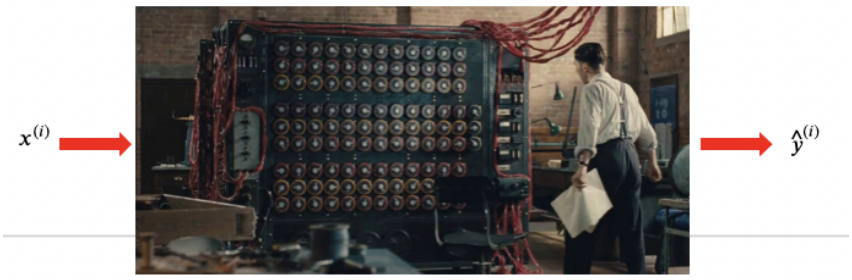
4. Unify

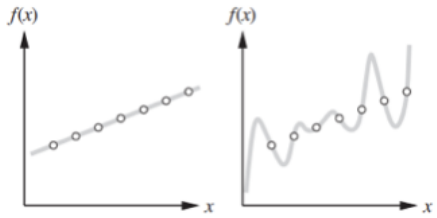
```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound expression
          y, a variable, constant, list, or compound expression
          θ, the substitution built up so far (optional, defaults to empty)

  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
      return UNIFY(x.ARGS, y.ARGS, UNIFY(x.OP, y.OP, θ))
  else if LIST?(x) and LIST?(y) then
      return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, θ))
  else return failure
```

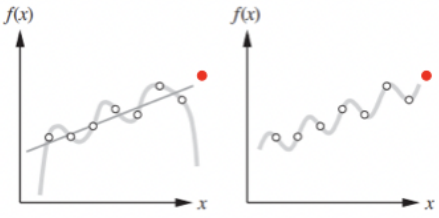
a.

5. The Problem
    a. This time we have the inputs and the ground truth given to us
    b. Supervised learning provides the correct answer along the data
        i. In (reinforcement learning) RL: ground truth came in a few forms:
            1. Bellman Equation w/ Rewards (were you rewarded or punished based on the answer you gain → but do not have the correct answer labeled, just the points)
            2. Policy Gradients
    c. Here ground truth is known and fixed
        i. Pro: we don't have to worry about generating it
        ii. Con: what is the "quality" of the dataset that we have?
            1. Person collecting the data can make mistakes → can unintentionally collect wrong answers

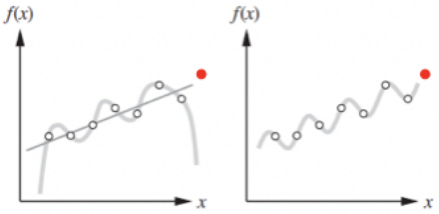
6. Example
    a. Task: Classify natural images into one of 1000 categories
        i. 
        ii. Each data comes with the respective category (somebody labels mushrooms as mushrooms)
        iii. (ImageNet 1k)
        iv. The dataset:
            1. 14.2M images (standard size, subject centered)
            2. Each image has a label (the category of that image)
                a. Each image can have multiple materials in it but does not exist in this kind dataset
                b. However, in that case, there can be mistakes in classifying
        v. Is this representative of natural images?
            1. How many natural images could exist?
            2. What are “hard” images from “easy” images?
7. The Data
    a. Theoretically we have a data space
        i. Want:
            1. Dataset contains every possible sample (and label)
            2. Don’t know how to distinguish “hard” examples from “easy” examples
        ii. Settle for:
            1. Dataset that is representative of the space
            2. How do we know this?
    b. Assumption:
        i. Data we have is drawn i.i.d from data space
        ii. Dataset = $\left\{\left(x^{(i)},\ y_{gt}^{(i)}\right)\right\}_{i=1}^{N}$
            1. gt = ground truth
8. The Goal
    a. Learn the function $f_\theta: X \rightarrow Y$ which maps examples to their ground truth
    b. Typically parameterize f: goal becomes learn the parameters

c. 

9. The Hypothesis Space
    a. If we have N data points, what functions could we learn?
    b. 

    c. Goal: search through possible functions to select the best one
10. The Goal Pt 2
    a. One important factor we forgot to mention:
        i. Since our data is (most likely) not the entire data space
        ii. AND we have no way to check if it is representative
        iii. We want our function to generalize
            1. The function needs to perform well on data it hasn't seen before
    2. 

11. How to get Unseen Data?
    a. Split your dataset into a training set D_{train} and a testing set D_{test}
    b. How?
        i. Lots of ways:
            1. Cross validation
            2. Random split
            3. Fixed partitons
            4. Someone else did it for you
    c. For now, let us assume random splits
        i. Something like 90% of your examples go into D_{train} and 10% into D_{test}
            1. Common choice
            2. Also 95%/5%, etc.

12. Generalization
    a. How can we expect to do well on unseen data?
        i. The function cannot learn from the unseen data…that's cheating
        ii. 

        iii. Ockham's razor? (Prefer simpler functions over more complicated ones?)
        iv. Can we estimate the likelihood of our model (given our data)?
13. The Learning Objective
    a. Call our hypothesis space $\mathcal{H}$
        i. How bigs the hypothesis space?
            1. If there are 15 booleans, there are 2^(2^15)
            2. Each boolean can give true/false and we can put operators in between
            3. Crazy amount of them
    b. Evaluate the "quality" of a function $h \in \mathcal{H}$:
        i. $quality(h) = \Pr[h \mid D]$
    c. Therefore:

$$h^* = \underset{h \in \mathcal{H}}{\text{argmax}} \Pr[h \mid D]$$
$$= \underset{h \in \mathcal{H}}{\text{argmax}} \Pr[D \mid h] \Pr[h]$$

        i. Data is given
14. How to Evaluate the Learning Objective
    a. Since we have D_{train} and D_{test}:
    b. Train on D_{train} a little bit to get h (by gradient descent)
        i. Eval h on D_{test} and keep the best h?
        ii. Cheating
            1. Learning (indirectly) from D_{test}
        iii. So what do we do?
    c. If our training paradigm is iterative (i.e. improve h incrementally)
        i. Give h a quiz!
15. Training/Validation/Testing Data
    a. Instead of splitting data into D_{train} and D_{test}
    b. Make three groups: D_{train}, D_{val} and D_{test}

i. D_{val} is quiz data: data that h doesn't directly train on but we use to check learning

c. When we claim we have found the "best" h, then give it a test (D_{test})

d. How to split?

i. Typically split like 90%/5%/5%

ii. Cross validation, etc.

16. The problem

a. ML is ill posed:

i. The more we train, the less likely we will generalize

ii. Need to train though!

b. Why?

i. If we only have training data:

1. What is inside $\mathcal{H}$?
2. If our task is realizable, then $h_{true} \in \mathcal{H}$
3. What else is in there?
    a. Avoid $h_{memorize}$ at all costs
    b. Can't tell the difference with training data!

c.