CSE 417T
Introduction to Machine Learning

Lecture 13

Instructor: Chien-Ju (CJ) Ho

- Homework 3: Due **Mar 19 (Friday).** Keep track your late days

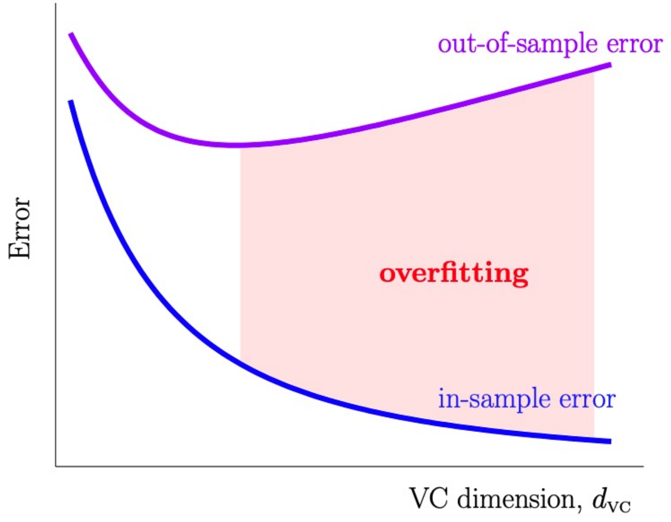
- Exam 1: **Mar 23 (Tuesday)**
  - Duration: 75+5 Minutes
  - Content: LFD Chapters 1 to 5
  - Time: By default, everyone is expected to take it during lecture time
    - Let me know **by this Friday (via private Piazza post)** if you can't do it during lecture time
  - Format: Gradescope online exam + turning on Zoom camera
    - A dummy exam is on Gradescope: Try file uploading and test various scenarios
  - Information access during exam:
    - Allowed: Textbook, slides, hardcopy materials (e.g., your own notes)
    - Not allowed: search for information online during exam, talk to any other persons

  - Other notes
    - **Follow Piazza announcements**
    - I'll give some practice questions next week
    - Next Thursday lecture will be a review lecture

Recap

Overfitting and Its Cures

- Overfitting
  - Fitting the data more than is warranted
  - Fitting the noise instead of the pattern of the data
  - Decreasing E_{in} but getting larger E_{out}
  - When H is too strong, but N is not large enough



- Regularization
  - Intuition: Constraining H to make overfitting less likely to happen

- Validation
  - Intuition: Reserve data to estimate E_{out}

Regularization

- Constraining H
  - Example: Weight decay $H(C) = \{h \in H_Q \text{ and } \vec{w}^T\vec{w} \leq C\}$
  - Finding g => Constrained optimization

minimize $E_{in}(\vec{w})$

subject to $\vec{w}^T\vec{w} \leq C$

- Defining augmented error
  - $E_{aug}(h, \lambda, \Omega) = E_{in}(\vec{w}) + \frac{\lambda}{N}\Omega(h)$
  - Finding g => Unconstrained optimization

minimize $E_{in}(\vec{w}) + \frac{\lambda_C}{N}\vec{w}^T\vec{w}$

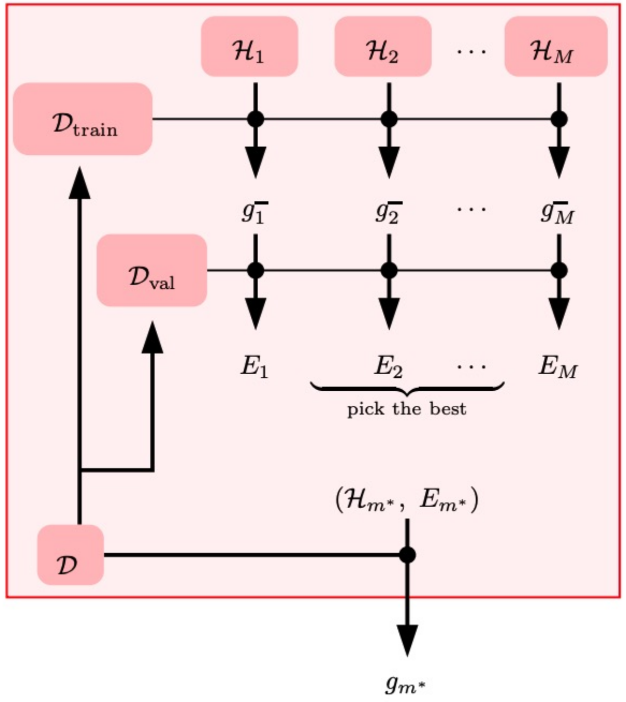
- The two interpretations are conceptually equivalent in a lot of cases.

- Understand the impacts of choosing Ω and λ

Validations

- Reserving data to estimate E_{out}

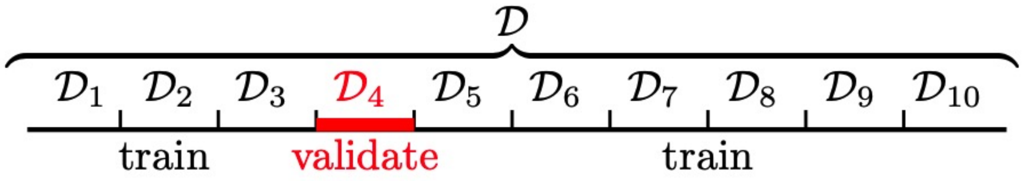
Model Selection



Note that the outlook comparisons are "in expectation"
If you only get one "draw" of $D_{train}, D_{val}, D_{test}$,
you cannot say anything "for certain"

	Outlook	Relationship to E_{out}
E_{in}	Incredibly optimistic	VC-bound
E_{val} (when used for model selection)	Slightly optimistic	Hoeffding's bound (multiple hypotheses)
E_{test}	Unbiased	Hoeffding's bound (single hypothesis)

- Cross Validation



Today’s Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

Let me know if you spot errors.

Three Learning Principles

Occam’s Razor

Sampling Bias

Data Snooping

Occam’s Razor

The simplest model that fits the data is also the most plausible

Sampling Bias

If the data is sampled in a biased way, learning will produce a similarly biased outcome.

Credit card example

- Determine whether to approve credit cards given applicants' financial information
- Banks have lots of data:
  - Customer information
  - Whether they are good customers or not
- Are there any issues here?

age	32 years
gender	male
salary	40,000
debt	26,000
years in job	1 year
years at home	3 years
...	...

Approve for credit?

BUSINESS NEWS OCTOBER 9, 2018 / 10:12 PM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

8 MIN READ

The New York Times

Facial Recognition Is Accurate, if You're a White Guy

By Steve Lohr

Feb. 9, 2018

SONIA PAUL BACKCHANNEL 03.20.2017 12:00 AM

Voice Is the Next Big Platform, Unless You Have an Accent

It's super funny that Alexa can't understand my mom — until we need Alexa to use the web, drive a car, and do pretty much anything else.

We will spend 1~2 lectures towards the end of the semester to talk about various ethical considerations of ML.

Occam's Razor

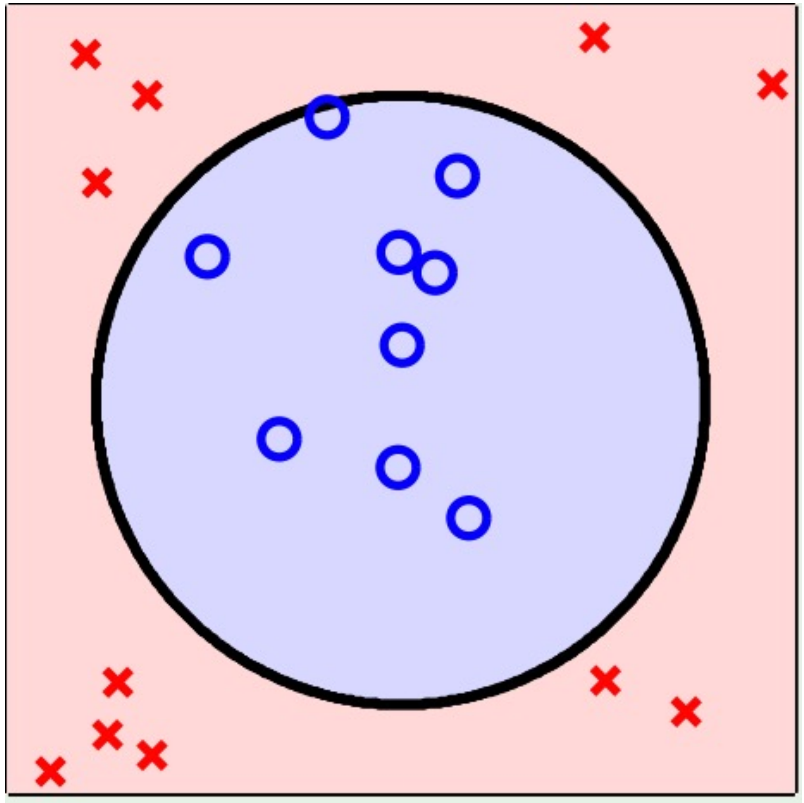
Sampling Bias

Data Snooping

Data Snooping

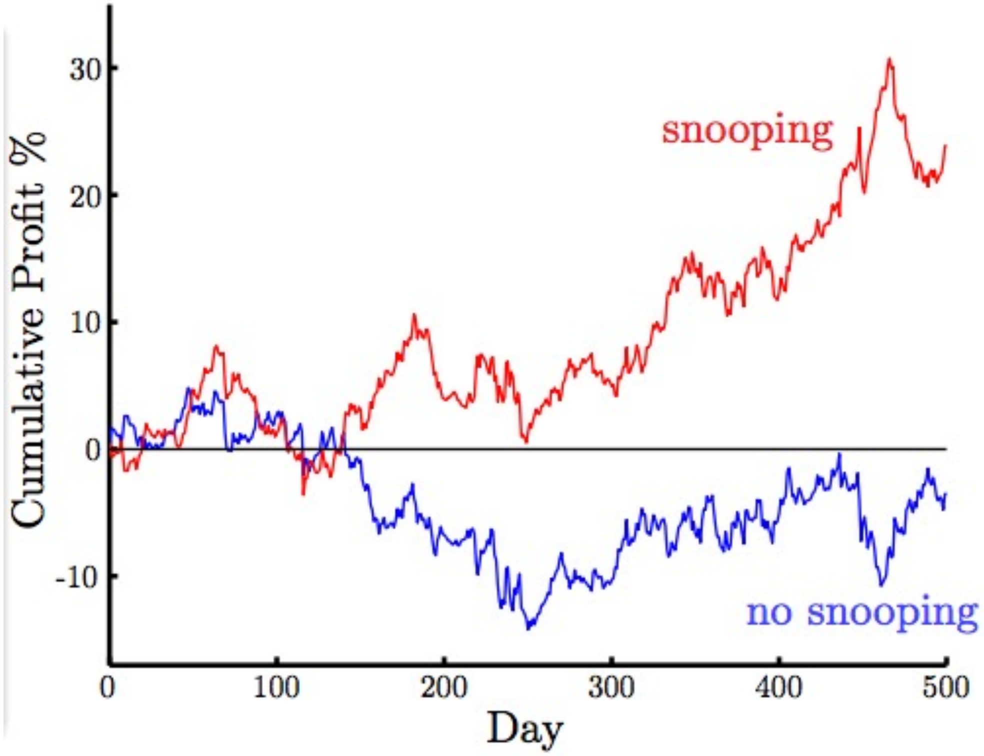
If a data set has affected any step in the learning process, its ability to assess the outcome has been compromised.

Shouldn't look at the data before selecting H



A Subtle Example

- Predict US Dollar vs. British Pound
  - $\vec{x}$: the change for the previous 20 days
  - y: the change in the 21th day
- Normalize data
- Split data $D = D_{train} \cup D_{test}$



- Where does snooping happen?
  - The normalization "looks at" D_{test}

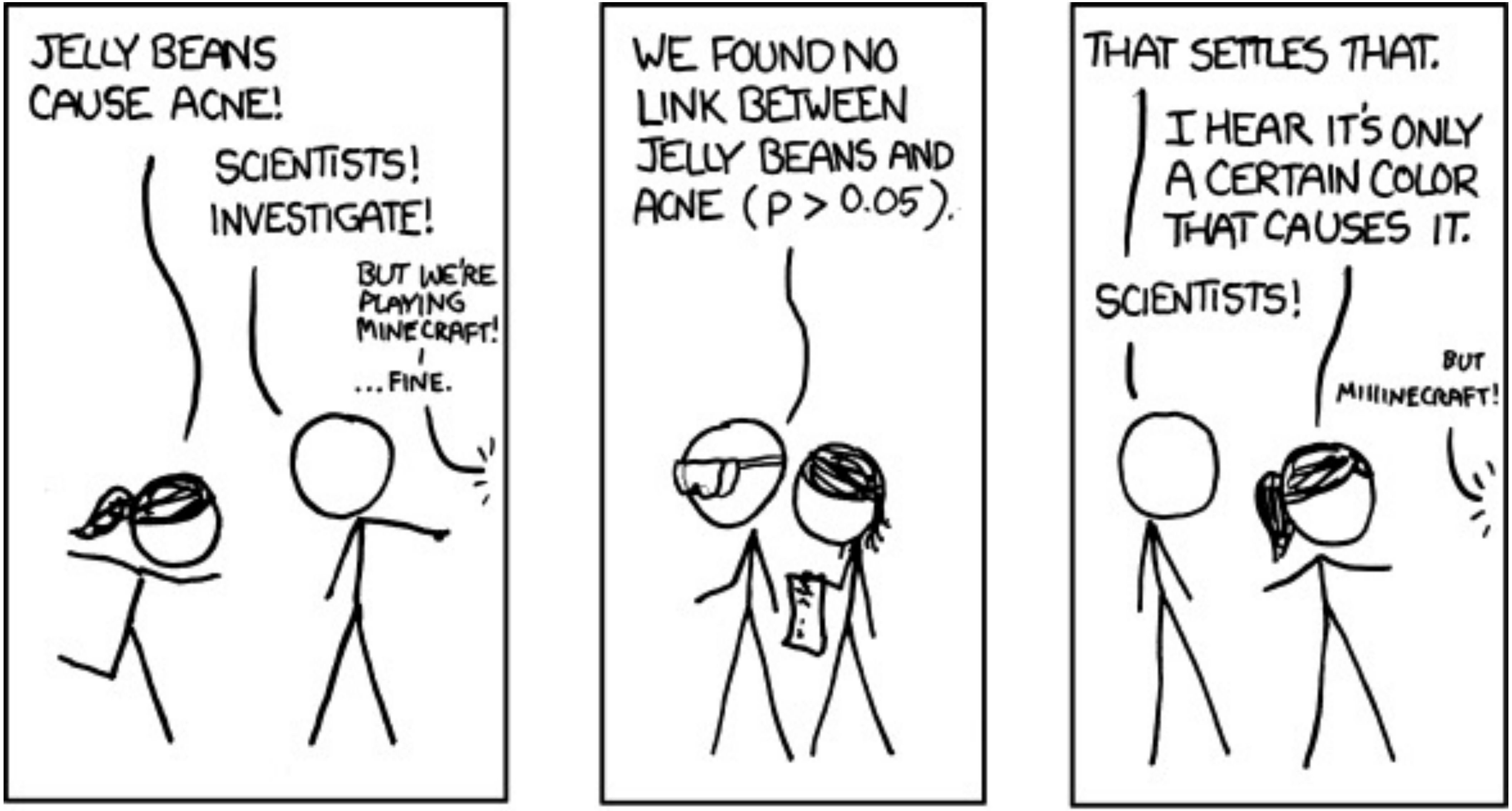
- How should you perform normalization in Q1 of HW2?

Reuse of a Data Set

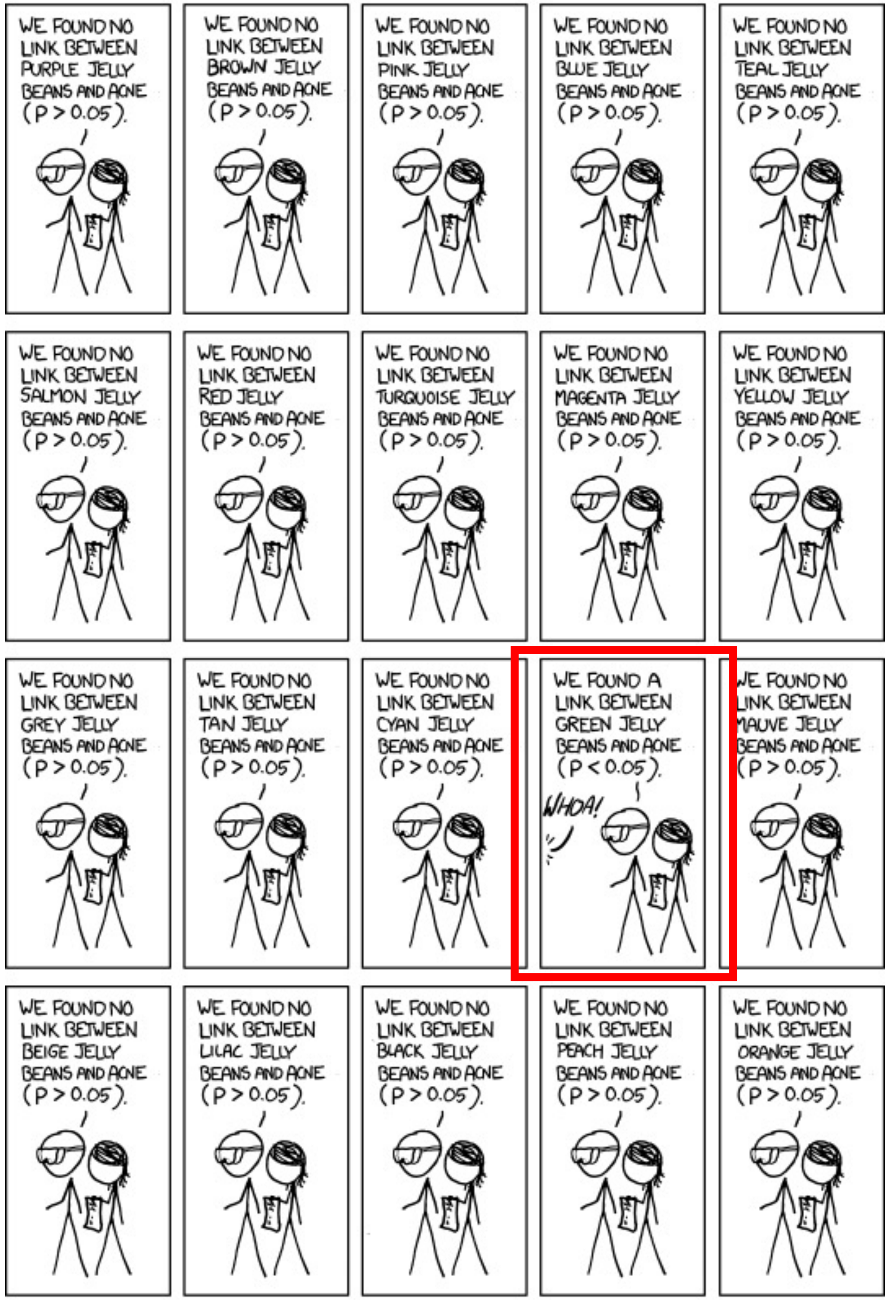
- Try one model after another **on the same data set**, you will eventually succeed.

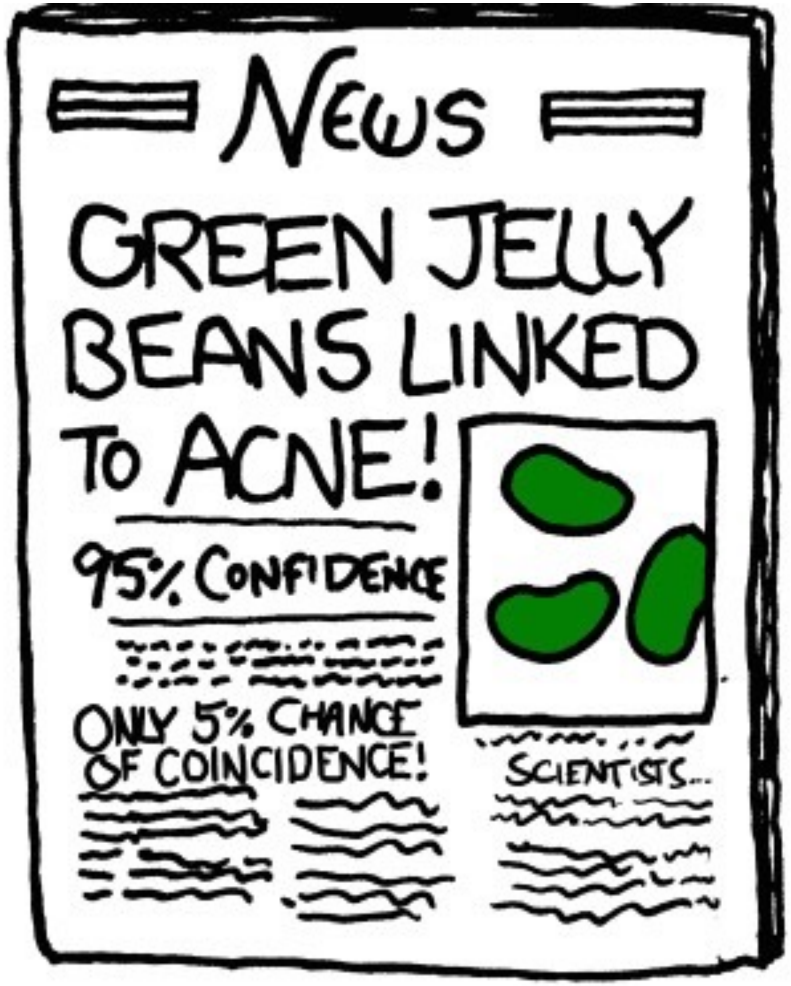
"If you torture the data long enough, it will confess"

- VC dimension of the total learning models
- May even include what others tried (e.g., if you read their paper…)
- p-hacking…



From xkcd, by Randall Munroe: http://xkcd.com/882





From xkcd, by Randall Munroe: http://xkcd.com/882

What Should We Do…

Avoid data snooping

- Strict discipline
- E.g., be **honest** and lock the test data

Account for data snooping

- Measure how much data is contaminated
- E.g., what we discussed in validation

Occam’s Razor

Sampling Bias

Data Snooping

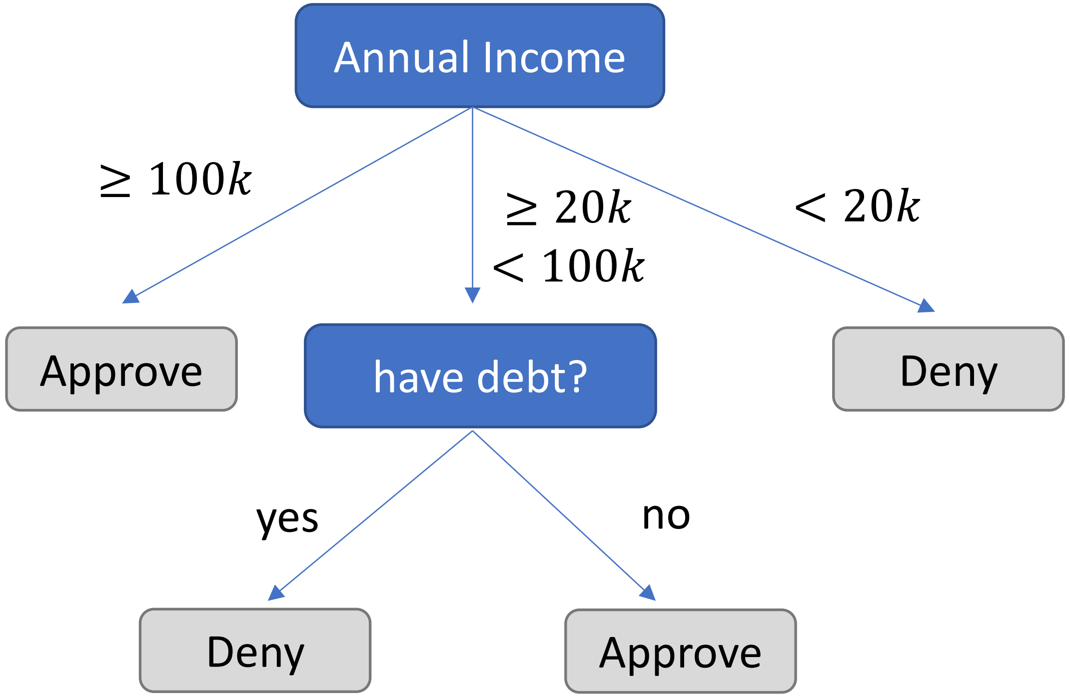
Content of Exam 1 Till Here

Course Plan

- Foundations
  - What's machine learning
  - Feasibility of learning
  - Generalization
  - Linear models
  - Non-linear transformations
  - Overfitting and how to avoid it
    - Regularization
    - Validation
- Techniques
  - Decision tree
  - Ensemble learning
    - Bagging and random forest
    - Boosting and Adaboost
  - Nearest neighbors
  - Support vector machine
  - Neural networks
  - …

Decision Tree

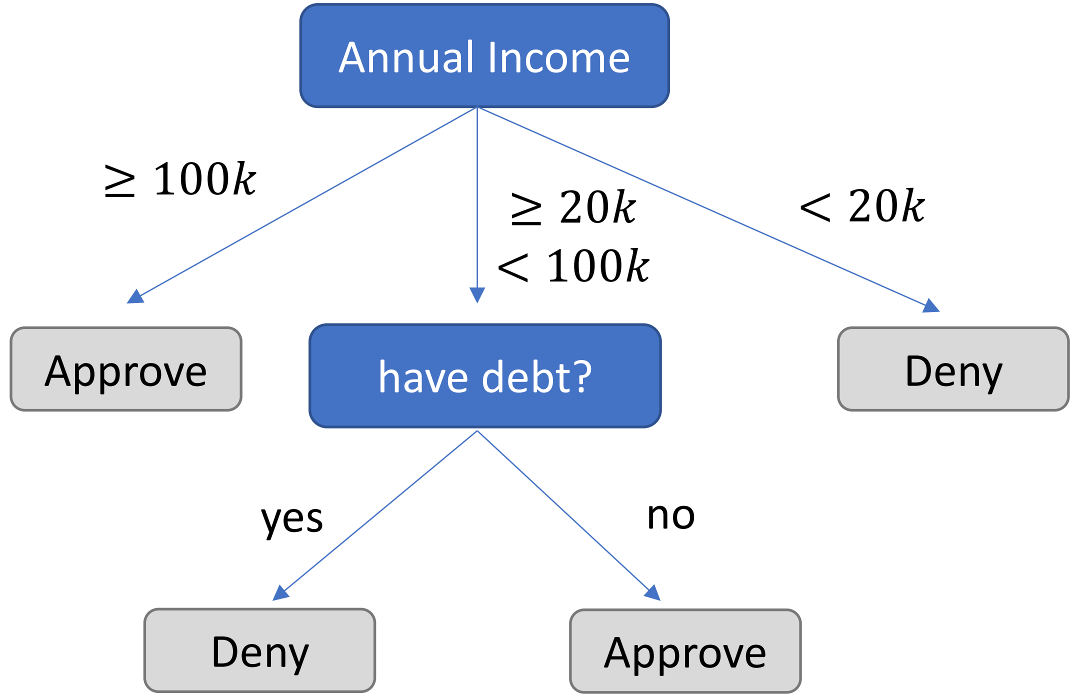
Decision Tree Hypothesis



Credit Card Approval Example

- $\vec{x} = (\text{annual income, have debt})$
- $y \in \{\text{approve, deny}\}$

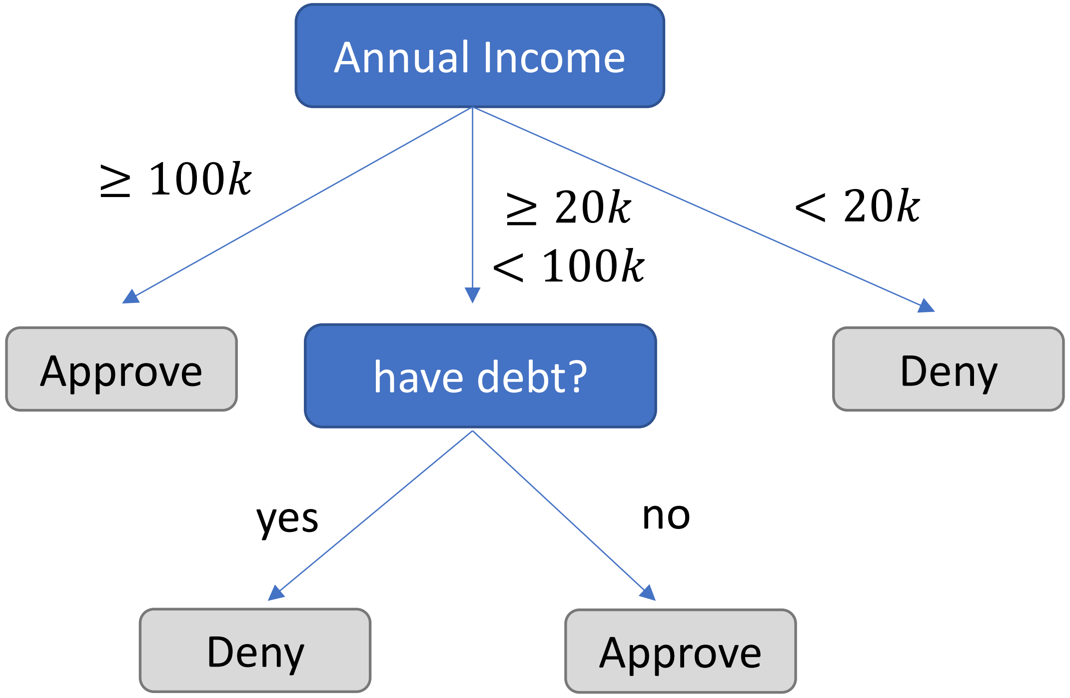
Decision Tree Hypothesis



Credit Card Approval Example

- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. …)
  - Easy to implement (Bunch of if-else rules)
- Cons

Decision Tree Hypothesis



Credit Card Approval Example

- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. …)
  - Easy to implement (Bunch of if-else rules)
- Cons
  - Generally speaking, bad generalization
  - VC dimension is infinity
  - High variance (small change of data leads to very different hypothesis)
  - Easily overfit
- Why we care?
  - One of the classical models
  - Building block for other models (e.g., random forest)

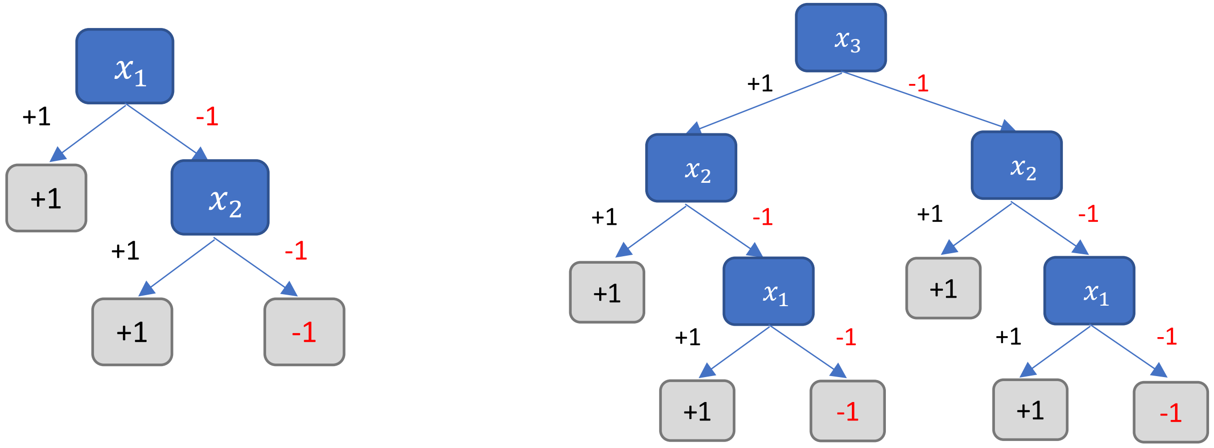
Learning Decision Tree from Data

- Given dataset D, how to learn a decision tree hypothesis?

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

- Potential approach
  - Find $g = argmin_{h \in H} E_{in}(h)$

- Multiple decision trees with zero E_{in}



Which one do you think might generalize better?

Learning Decision Tree from Data

- Conceptual intuition to deal with overfitting
  - Regularization: Constrain H

- Informally,

minimize E_{in}

subject to $size(tree) \leq C$

- This optimization is generally computationally intractable.
- Most decision tree learning algorithms rely on ***heuristics*** to approximate the goal.

Template of Greedy-Based Decision Tree Algorithm

- DecisionTreeLearn(D): Input a dataset D, output a decision tree hypothesis
  - Create a root node
  - If termination conditions are met
    - return a single node tree with leaf prediction based on D
  - Else: Greedily find a feature A (assigned as root) to split according to split criteria
    - For each possible value v_i of A
      - Let D_i be the dataset containing data with value v_i for feature A
      - Create a subtree DecisionTreeLearn(D_i) that being the child of root

- Most decision tree learning algorithms follow this template, but with different choices of heuristics

Example

DecisionTreeLearn(D)
  Create a root node
  If termination conditions are met
    return a single node tree with leaf prediction based on D
  Else: Greedily find a feature A (assigned as root) to split according to split criteria
  For each possible value v_i of A
    Let D_i be the dataset containing data with value v_i for feature A
    Create a subtree DecisionTreeLearn(D_i) that being the child of root

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

DecisionTreeLearn →

Termination conditions not net
Find a feature to split

x_1

+1 -1

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1

DecisionTreeLearn

terminate

Leaf prediction +1

x_1	x_2	x_3	y
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

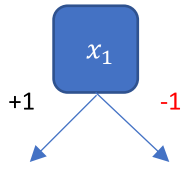
DecisionTreeLearn

Don’t terminate

Find next feature to split

Example Heuristics

DecisionTreeLearn(D)
    Create a root node
    If termination conditions are met
        return a single node tree with leaf prediction based on D
    Else: Greedily find a feature A to split according to split criteria
    For each possible value v_i of A
        Let D_i be the dataset containing data with value v_i for feature A
        Create a subtree DecisionTreeLearn(D_i) that being the child of root



x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1

DecisionTreeLearn

terminate

Leaf prediction +1

x_1	x_2	x_3	y
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

DecisionTreeLearn

Don't terminate

Find next feature to split

- Termination conditions
  - When the dataset is empty
  - When all labels are the same
  - when all features are the same
  - When the depth of the tree is too deep
  - …
- Leaf predictions
  - Majority voting
  - Average (for regression)
  - …
- Split criteria?

Split Criteria

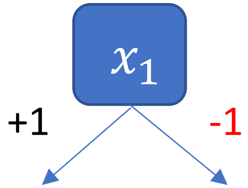
- Which feature would you choose to split?

x_1	x_2	y
+1	+1	+1
+1	-1	+1
-1	+1	-1
-1	-1	-1

Split Criteria

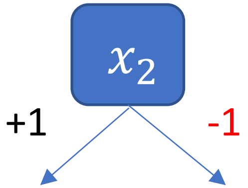
- Which feature would you choose to split?

x_1	x_2	y
+1	+1	+1
+1	-1	+1
-1	+1	-1
-1	-1	-1



x_1	x_2	y
+1	+1	+1
+1	-1	+1

x_1	x_2	y
-1	+1	-1
-1	-1	-1



x_1	x_2	y
+1	+1	+1
-1	+1	-1

x_1	x_2	y
+1	-1	+1
-1	-1	-1

- Want the tree to be "smaller"
  - Intuition: choose the one that the labels are more "pure"
  - Example: choose the one maximizing information gain => ID3 Algorithm

Brief Intro to Information Entropy

- Assume there are K possible labels
- Entropy:
  - $H(D) = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i}$
  - p_i: ratio of points with label i in the data

By definition

$0 \log_2 \frac{1}{0} = 0\,;\ 1 \log_2 \frac{1}{1} = 0$

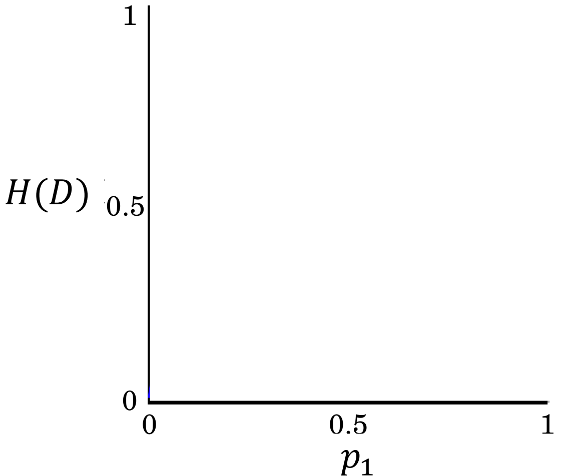
Brief Intro to Information Entropy

- Assume there are K possible labels
- Entropy:
  - $H(D) = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i}$
  - p_i: ratio of points with label i in the data

By definition

$0 \log_2 \frac{1}{0} = 0\,;\ 1 \log_2 \frac{1}{1} = 0$

- Binary case with $K = 2$



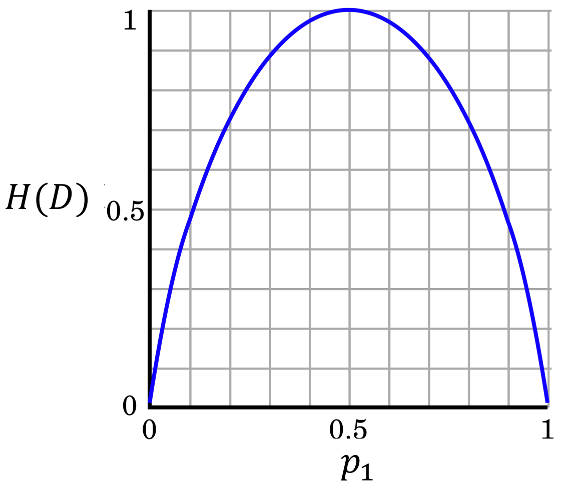
Brief Intro to Information Entropy

- Assume there are K possible labels
- Entropy:
  - $H(D) = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i}$
  - p_i: ratio of points with label i in the data

By definition

$0 \log_2 \frac{1}{0} = 0\,;\; 1 \log_2 \frac{1}{1} = 0$

- Binary case with $K = 2$



- Interpretations of entropy
  - Expected # bit to encode a distribution
- Higher entropy
  - data is less "pure"
- "pure" data => all labels are +1 or -1 => entropy = 0
- Want to choose splits that lead to pure data, i.e., lower entropy

ID3: Using Information Gain as Selection Criteria

- Information gain of choosing feature A to split
  - $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$ [The amount of decrease in entropy]
- ID3: Choose the split that maximize $Gain(D, A)$

Notation:
$|D|$ is the number of points in D

```
DecisionTreeLearn(D)
    Create a root node
    If termination conditions are met
        return a single node tree with leaf prediction based on D
    Else: Greedily find a feature A to split according to split criteria
    For each possible value v_i of A
        Let D_i be the dataset containing data with value v_i for feature A
        Create a subtree DecisionTreeLearn(D_i) that being the child of root
```

- ID3 termination conditions
  - If all labels are the same
  - If all features are the same
  - If dataset is empty
- ID3 leaf predictions
  - Most common labels (majority voting)
- ID3 split criteria
  - Information gain

ID3: Using Information Gain as Selection Criteria

- Information gain of choosing feature A to split
  - $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$
- ID3: Choose the split that maximize $Gain(D, A)$

x_1	x_2	y
+1	+1	+1
+1	-1	+1
-1	+1	-1
-1	-1	-1

$H(D) = 0.5 \log_2 2 + 0.5 \log_2 2 = 1$

x_1: +1 / -1

x_1	x_2	y
+1	+1	+1
+1	-1	+1

$H(D_{x_1=1}) = 0$

x_1	x_2	y
-1	+1	-1
-1	-1	-1

$H(D_{x_1=-1}) = 0$

$Gain(D, x_1) = 1$

x_2: +1 / -1

x_1	x_2	y
+1	+1	+1
-1	+1	-1

$H(D_{x_2=1}) = 1$

x_1	x_2	y
+1	-1	+1
-1	-1	-1

$H(D_{x_2=-1}) = 1$

$Gain(D, x_2) = 0$

ID3 will choose x_1 as the next split attribute

Further Addressing Overfitting

- More Regularization (Constrain H)
  - Do not split leaves past a fixed depth
  - Do not split leaves with fewer than c labels
  - Do not split leaves where the maximal information gain is less than τ
- Pruning (removing leaves)
  - Evaluate each split using a validation set and compare the validation error with and without that split (replacing it with the most common label at that point)
  - Use statistical test to examine whether the split is "informative" (leads to different enough subtrees)

More Discussions

- Real-valued features (continuous x)
  - Need to select threshold for branching
- Regression (continuous y)
  - Change leaf prediction: e.g., average instead of majority vote
  - Change measure for "purity" of data: e.g., squared error of data