Machine Learning

CS 165B

Prof. Matthew Turk

Monday, April 18, 2016

Today

- Concept learning
- Decision trees (Ch. 5)

Notes

- HW#2 due on Friday, 4:30pm
  - Version 2 (v. 2) posted – problem 2 (classifiers C0 through C6), stated which side of the discriminant line is "positive"
  - Reminder:

    *"Justify every answer you give – **show the work** that achieves the answer or **explain** your response."*

    Just giving a numeric answer without showing how you arrived at that value will be marked incorrect.

Concept learning + hypothesis space

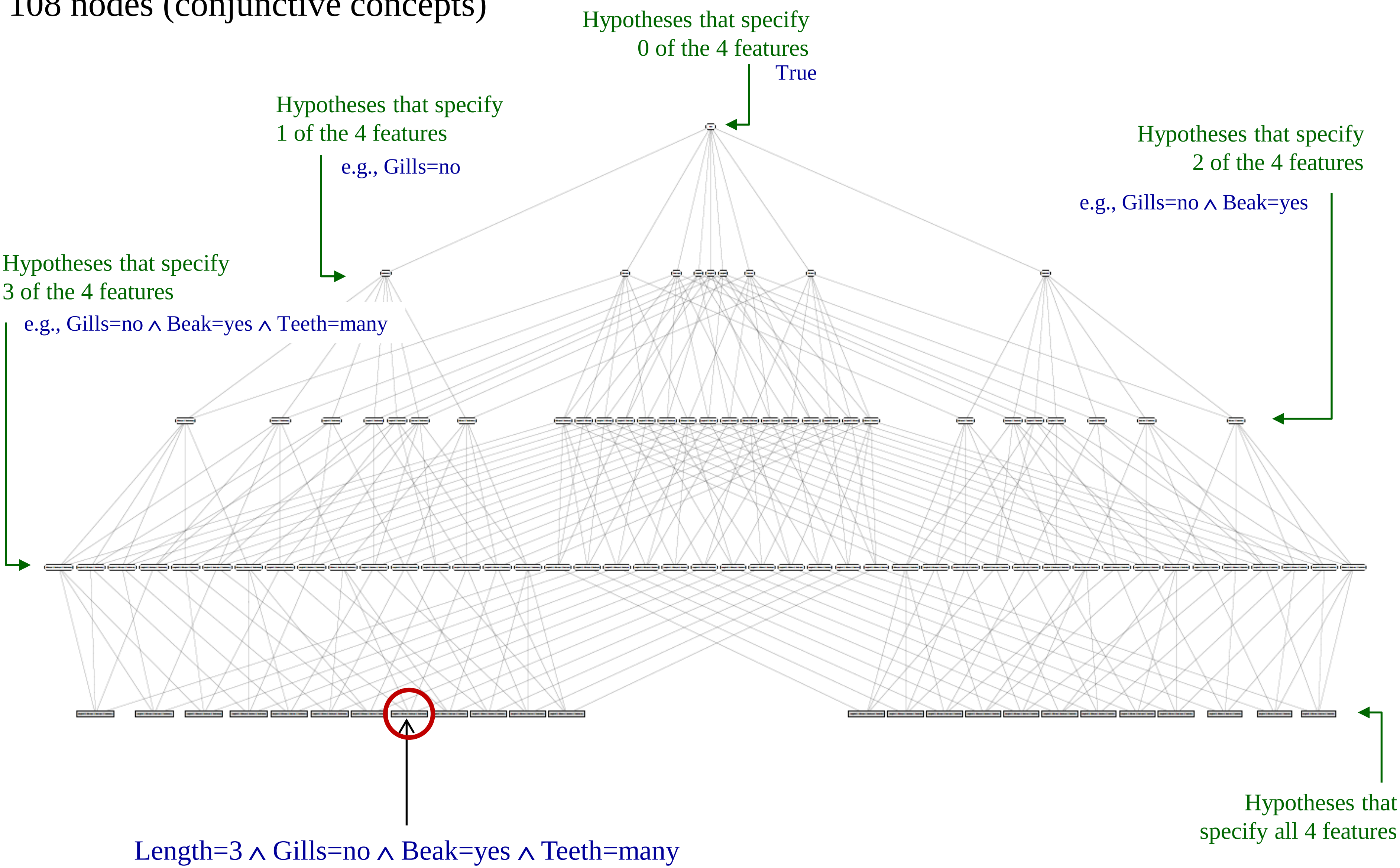
- In concept learning, we want to learn a Boolean function (our hypothesis) over a set of attributes+values
  - E.g.: Temperature = high $\wedge$ Coughing = yes $\wedge$ Spots = yes
  - Some combinations are in the concept/hypothesis, others are not
- The target concept ***c*** is the true concept – we'd like to choose a hypothesis ***h*** such that $\boldsymbol{h} \approx \boldsymbol{c}$
- The hypothesis space is the space of all possible concepts (hypotheses) over the attributes
- For N attributes, each with with F_i values, there are $2^{(F_1 \times F_2 \times \ldots \times F_N)}$ possible hypotheses
- Our problem: given the training data, which hypothesis should we choose to represent the concept?
  - It should generalize well to new, unseen instances

The conjunctive hypothesis space

- We'll limit our hypothesis space to conjunctive concepts, where hypotheses are represented as assigning a value (including "don't care") to each attribute
  - E.g., Quarter=Fall $\wedge$ Dept=X $\wedge$ courselevel=ugrad $\wedge$ topic=X represents the concept "Fall undergraduate courses"
    - Or Quarter=Fall $\wedge$ courselevel=ugrad or (Fall, X, ugrad, X)
  - (Spring, Psychology, grad, Perception) – e.g., most specific hypothesis
  - (X, X, X, X) – most general hypothesis
- Then rule out all the hypotheses (concepts) that don't include all of the instances in our training data
- And finally choose the least general of these as our result – the concept defined by our training data
  - This is the least general generalization (LGG)

The Conjunctive Hypothesis Space

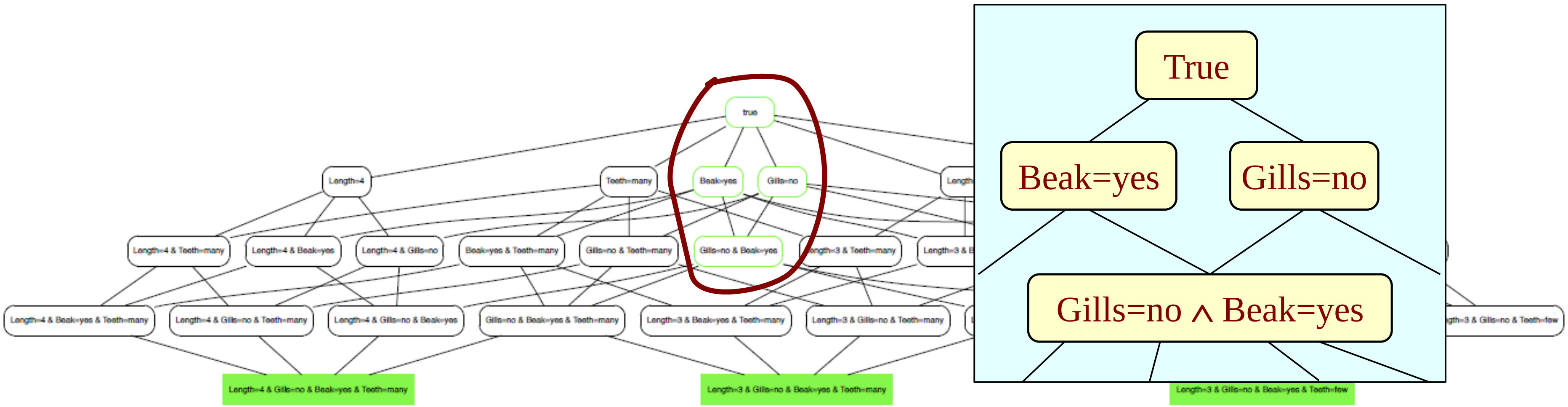
108 nodes (conjunctive concepts)



This connects upward to every more general hypothesis that includes it

Reducing the hypothesis space

- If we require hypotheses to cover all three training examples, we're left with only four concepts



- Let's choose the least general of these as our result – the concept defined by our training data

  Gills = no $\wedge$ Beak = yes

Least general generalization (LGG) procedure

Algorithm LGG-Set(D) – find least general generalisation of a set of instances.

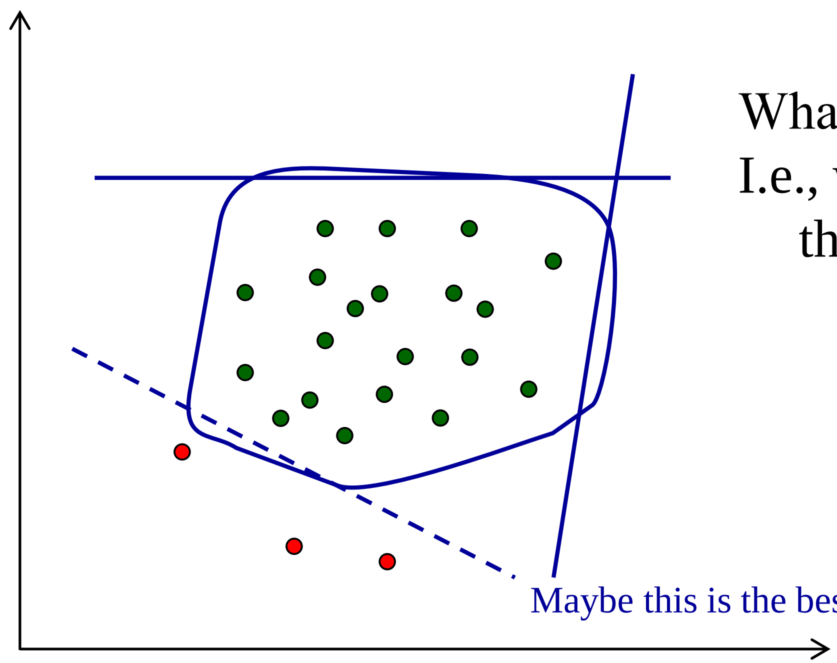
Input : data D.
Output : logical expression H.
$x \leftarrow$first instance from D;
$H \leftarrow x$;
while instances left **do**
  $x \leftarrow$next instance from D;
  $H \leftarrow$LGG(H, x) ; // e.g., LGG-Conj (Alg. 4.2) or LGG-Conj-ID (Alg. 4.3)
end
return H

Algorithm LGG-Conj(x, y) – find least general conjunctive generalisation of two conjunctions.

Input : conjunctions x, y.
Output : conjunction z.
$z \leftarrow$conjunction of all literals common to x and y;
return z

Negative examples

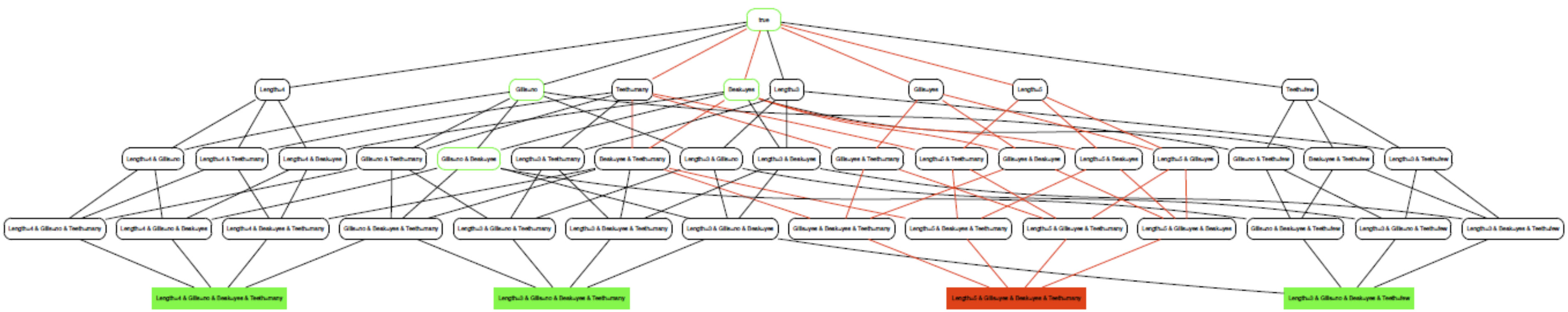
- So far, we’ve only looked at CHS learning from positive training examples (examples of the concept)
- Negative examples are very useful in learning concepts (for people and machines!)
  - Negative examples help to prune the hypothesis space

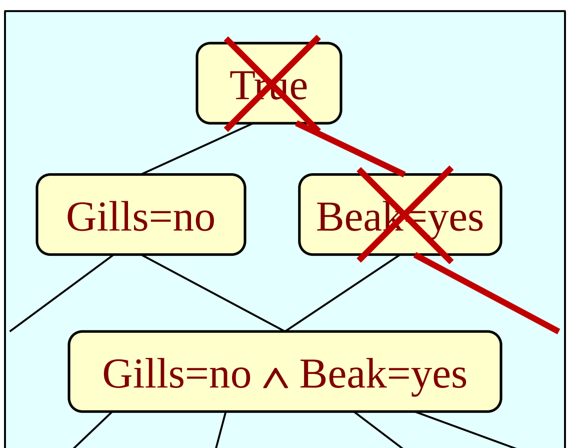


What’s the hypothesis?
I.e., where would we draw
the concept boundary?

Negative examples

- We'd like to refine our hypothesis (guide our search through hypothesis space) using both positive and negative examples
  - Rule out hypotheses that include negative examples





Negative sample: Length=5 ∧ Gills=yes ∧ Beak=yes ∧ Teeth=many

Adding internal disjunction to CHS

- We can make our representation somewhat richer by allowing internal disjunctions (ORs within a feature)
- So instead of positive examples of length=4 and length=5 causing a conflict and thus resulting in our concept containing length=X, we can use length=[4,5] in our concept
  - This means length=4 $\vee$ length=5
- F attribute values → $2^F - 1$ combinations w/internal disjunctions
- Allowing internal disjunction increases the size of our hypothesis space
  - Rather than $|H| = (3+1)(2+1)(2+1)(2+1) = 108$, we'll have $|H| = (2^3 - 1)(2^2 - 1)(2^2 - 1)(2^2 - 1) = 189$ hypotheses

Adding internal disjunction to CHS

- E.g., for Quarter, F = 4 so there are $2^4 - 1 = 15$ combinations:

  Quarter=Fall, Quarter=Winter, Quarter=Spring, Quarter=Summer

  Quarter=[Fall, Spring], Quarter=[Winter, Spring], (4 more)

  Quarter=[Fall, Spring, Summer], (3 more)

  ➡ Quarter=[Fall, Spring, Summer, Winter]

- When all values of a feature are included, the feature becomes X (*don't care*)

  Equivalent
  Quarter=[Fall, Spring, Summer, Winter]
  Quarter=Fall $\vee$ Quarter=Winter $\vee$ Quarter=Spring $\vee$ Quarter=Summer
  Quarter=X

Adding internal disjunction to our example

We (again) take a specific-to-general approach in coming up with a hypothesis:

Instances: Hypotheses:

(1) Length = 3 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many

Length = 3 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many

(2) Length = 4 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many

Length = [3,4] $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many

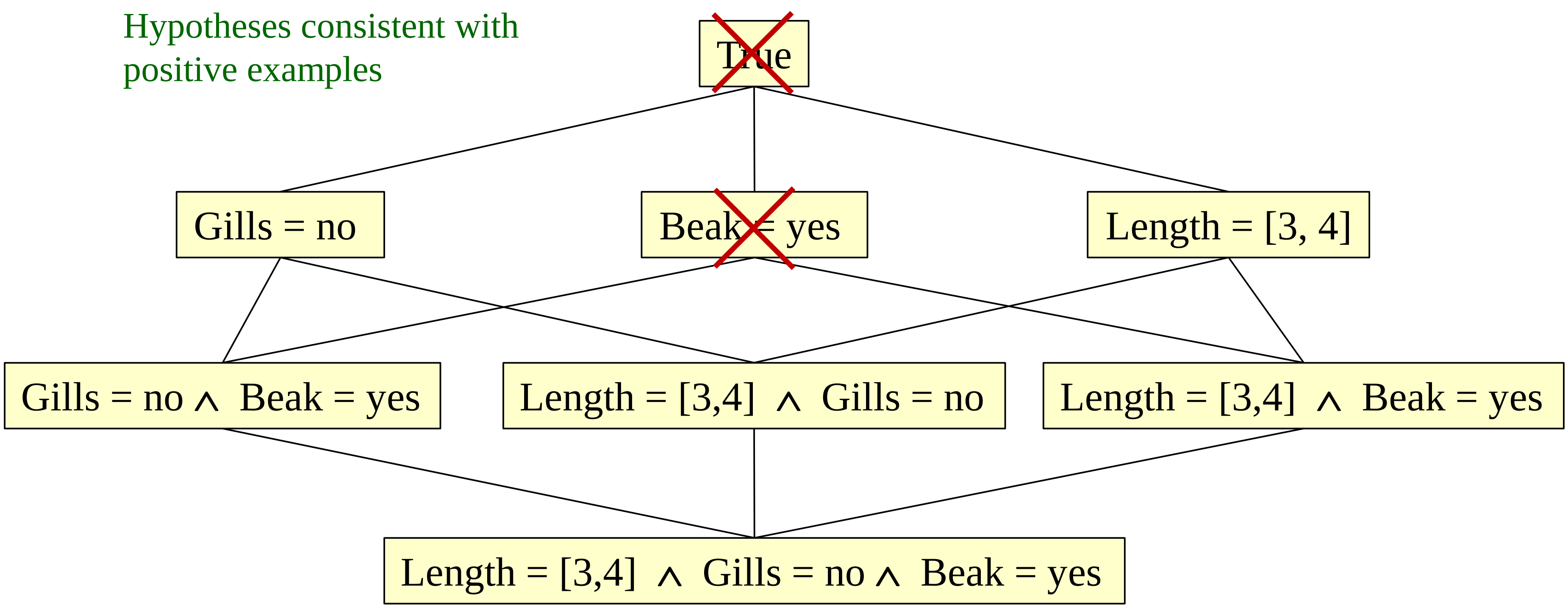
(3) Length = 3 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = few

Length = [3,4] $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = X

(4) [Negative] Length = 5 $\wedge$ Gills = yes $\wedge$ Beak = yes $\wedge$ Teeth = many

Length = [3,4] $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = X

Internal disjunction – adding negative example



Add negative example: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many

Which hypotheses are no longer consistent with the data?

Additional notes

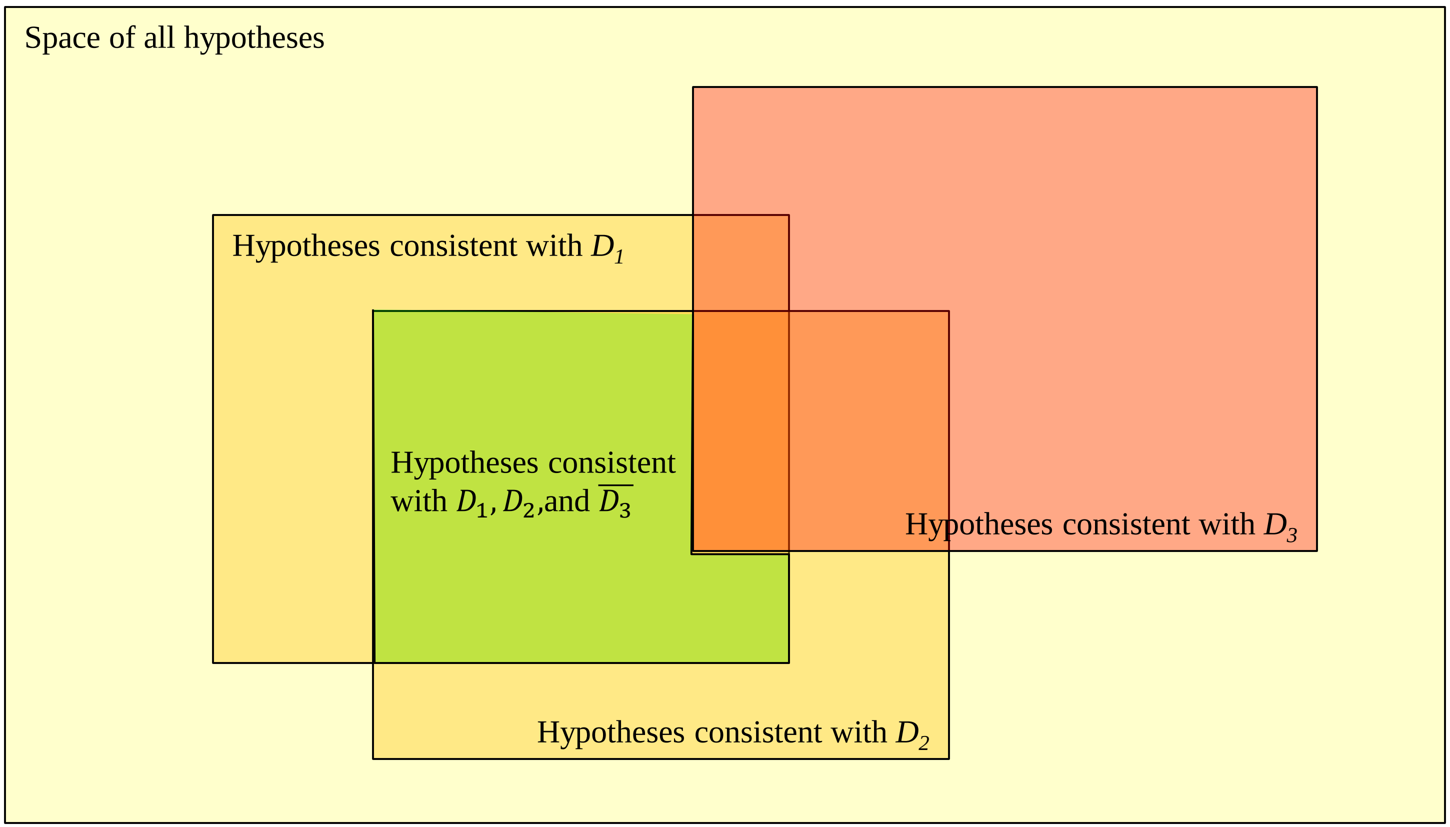
- There are two different issues here:
  1. The number of hypotheses consistent with the data
     - For a given training data example, which is a leaf node at the lowest level in the graph, all connected nodes above it (including the data point itself) are hypotheses consistent with that data point
     - Adding a new, different data point prunes the nodes that represent hypotheses consistent with both data points
       - For positive and negative data points
  2. The generality of various hypotheses
     - Higher/lower nodes represent more/less general hypotheses
     - Pruning a hypothesis is a separate issue from the generality of the hypothesis
- Our algorithm of finding a hypothesis from training data (starting with the first data point as the initial hypothesis and generalizing with more data points) prunes hypotheses at each step, choosing the least general consistent hypothesis as the current hypothesis

Additional notes

- Adding a positive training example prunes the space of consistent hypotheses by eliminating hypothesis that do not have a link to the new example
  - I.e., candidate hyptheses must be connected to all positive examples
  - The less similar it is to the previous training examples, the more it will prune
- Adding a negative training example prunes the space of consistent hypotheses by eliminating hypotheses that do have a link to the new example
  - I.e., candidate hypotheses must not be connected to any negative examples
  - The more similar it is to the previous training examples, the more it will prune
- A negative example will always prune the top node (True)

Additional notes

A Venn diagram to show how both positive and negative examples prune the space of consistent hypotheses:



Hypothesis languages for concept learning

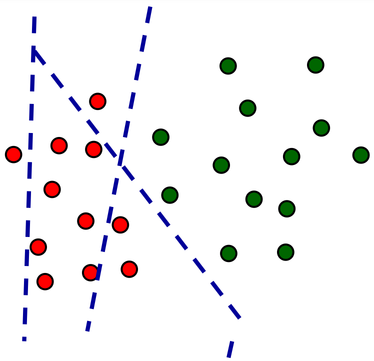
- This idea can be extended to more realistic data by modifying the "all or nothing" nature of positive/negative training data
- We can make richer hypothesis representation languages:
  - Conjunctive hypothesis space
  - Conjunctive hypothesis space with internal disjunctions
  - Conjunctions of Horn clauses
  - Clauses in first-order logic
  - Etc.
- The richer the representation and thus the more expressive the hypothesis language, the more difficult the learning problem
  - Learnability – how hard it is to learn a concept?
- Let's look at some important learning concepts and terms:

Complete and consistent concepts

“Cover” – classify as positive

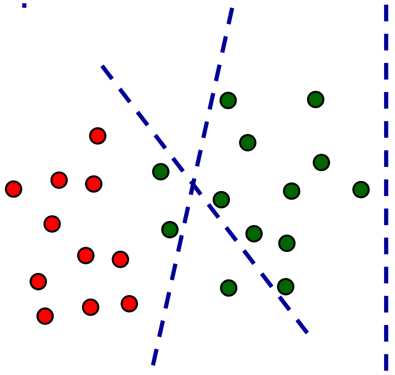
A concept is **complete** if it covers all positive examples

- FNR = 0

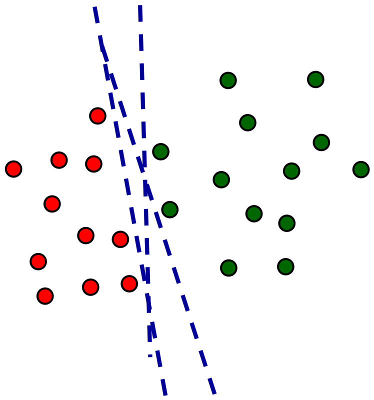


A concept is **consistent** if it covers none of the negative examples

- FPR = 0



The **version space** is the set of all concepts that are both complete and consistent



PAC learning

- Let's consider an important learning model:
  Probably Approximately Correct (PAC) learning
- If a concept is PAC-learnable, then there exists a learning algorithm that gets it mostly right, most of the time
- Terms:
  - Hypothesis: $\boldsymbol{h}$, hypothesis space: $\boldsymbol{H}$
  - Distribution of the (true) data: $\boldsymbol{D}$
  - Error rate of $\boldsymbol{h}$ for data distribution $\boldsymbol{D}$: $\boldsymbol{err_D}$
  - Allowable error rate: $\boldsymbol{\varepsilon}$
  - Allowable failure rate: $\boldsymbol{\delta}$
- PAC learning outputs, with probability at least $1-\boldsymbol{\delta}$, a hypothesis $\boldsymbol{h}$ such that $\boldsymbol{err_D} < \boldsymbol{\varepsilon}$

"most of the time"

"mostly right"

Low generalization error

PAC learning

We may get back to this later, but wanted to mention it now since it's briefly discussed in Chapter 4….

- Even with noise-free data and a complete and consistent hypothesis ***h*** (i.e., no errors on the training data), the training data may not have been perfectly representative of the instance space, and the hypothesis might have a "large" error ($\boldsymbol{err_D} > \boldsymbol{\varepsilon}$) over the instance space.
  - This should happen infrequently, with probability less than $\boldsymbol{\delta}$
- It turns out that we can guarantee this by choosing a large enough training set, $m = |D|$
  - With various assumptions, this is:

$$m \geq \frac{1}{\varepsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

- This leads to the concept of VC dimension, which is a key theoretical concept in machine learning

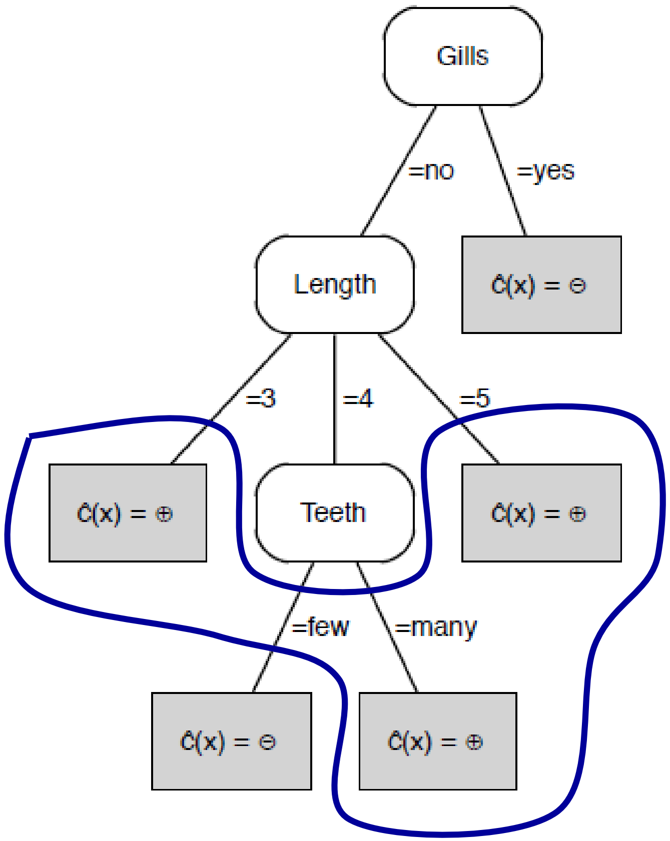
Decision Trees

Chapter 5 in the textbook

Tree models and decision trees

Decision trees

- A decision tree partitions the instance space by branching on feature values (literals), with leaves representing the learned concept
- Each leaf represents a conjunction of literals on its path
- The learned concept is the disjunction of the positive leaves
  - $L_1 \vee L_2 \vee L_3 \vee \ldots$
- Decision trees are maximally expressive – they can separate any consistently labeled data
  - Thus more powerful than the conjunctive hypothesis space we just discussed



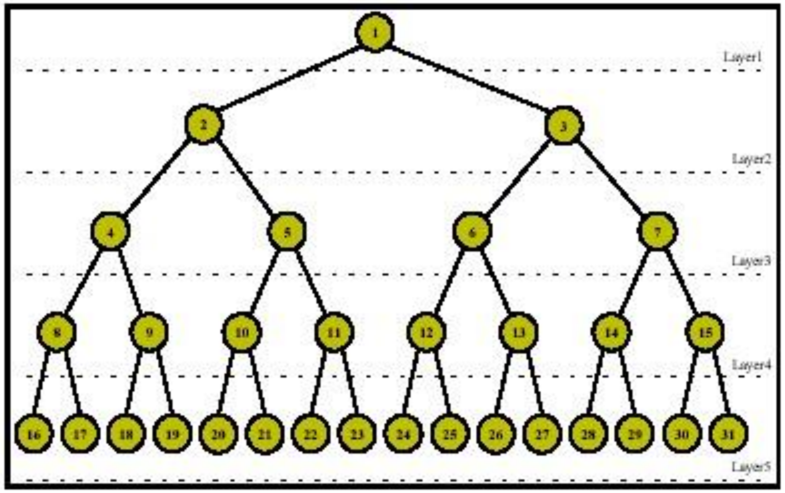
Ideally, each leaf contains only positives or only negatives from the training data

Key question: Which features (and in what order) will accomplish this best?

Decision trees

- The drawback of this is that they may not generalize well – i.e., overfitting can be a problem
  - So we have to employ mechanisms to enforce generalizations beyond the examples and avoid overfitting
  - These are referred to as the inductive bias of the learning algorithm

Overfitting?



- A typical inductive bias is towards less complex hypotheses
  - A linear discriminant in classification, a line for a regression function, a restrictive hypothesis language for concept learning, etc.
  - What’s the inductive bias in decision trees? (We’ll see….)

Decision trees

Note: This is very similar to ID3, a well-known DT algorithm – in tomorrow's discussion session!

- Tree models can be used for classification, ranking, probability estimation, regression, and clustering
- Recursive generic tree learning procedure:

100%?
99%?
80%

Most useful feature

Algorithm GrowTree(D, F) – grow a feature tree from training data.

Input : data D; set of features F.
Output : feature tree T with labelled leaves.
if Homogeneous(D) **then return** Label(D);
$S \leftarrow$BestSplit(D, F) ; // e.g., BestSplit-Class (Algorithm 5.2)
split D into subsets D_i according to the literals in S;
for each i **do**
  if $D_i \neq \emptyset$ **then** $T_i \leftarrow$GrowTree(D_i, F) ;
  else T_i is a leaf labelled with Label(D);
end
return a tree whose root is labelled with S and whose children are T_i

Divide-and-conquer approach: build a tree for each subset of the data, then merge into a single tree