# Machine Learning

## CS 165B

Prof. Matthew Turk

Wednesday, April 13, 2016

Today

- Classification (cont.)
  - Chapters 2-3
- Concept learning
  - Chapter 4

# Notes

- HW#2
  - Posted on Friday, due next Friday (April 22)

# Multi-class classification

- Many classification problems involve multiple classes
- Performance can be described with the multi-class contingency table
  - Not including the marginals, also known as the **confusion matrix**
  - We can compute accuracy, per-class precision, per-class recall…

| | Predicted | | | |
|---|---|---|---|---|
| | 15 | 2 | 3 | 20 |
| *Actual* | 7 | 15 | 8 | 30 |
| | 2 | 3 | 45 | 50 |
| | 24 | 20 | 56 | 100 |

Accuracy = (15+15+45)/100 = 0.75

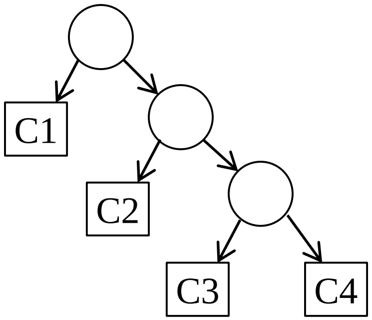
Class 1 precision = 15/24 = 0.63

Class 1 recall = 15/20 = 0.75

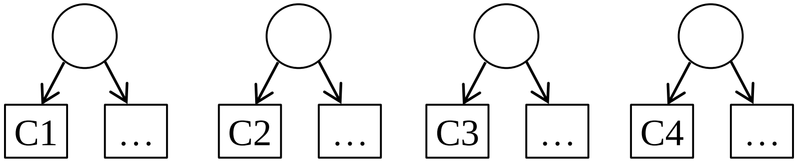
Etc.

# *K*-class classifiers

- How to build a *k*-class classifier?
  - We can combine several binary classifiers, e.g.:
    - One-versus-rest scheme – learn *k*-1 models, apply in sequence
      - C1 vs. { C2, C3, C4 }
      - C2 vs. { C3, C4 }
      - C3 vs. C4

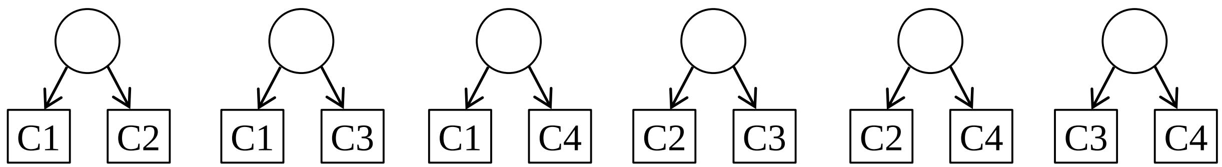



    - One-versus-rest scheme – learn a *one-class* model for each class
      - C1 vs. { C2, C3, C4 }
      - C2 vs. { C1, C3, C4 }
      - C3 vs. { C1, C2, C4 }
      - C4 vs. { C1, C2, C3 }

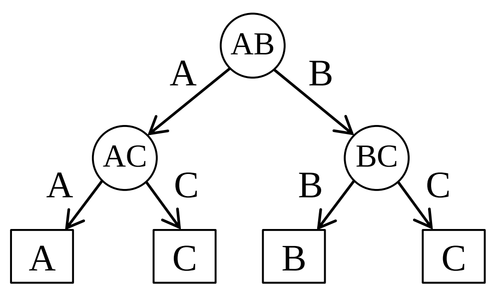


# $K$-class classifiers

- One-versus-one scheme – learn a model for each pair of classes
  - Train $k(k-1)/2$ binary classifiers, apply them all to $x$ and **vote**

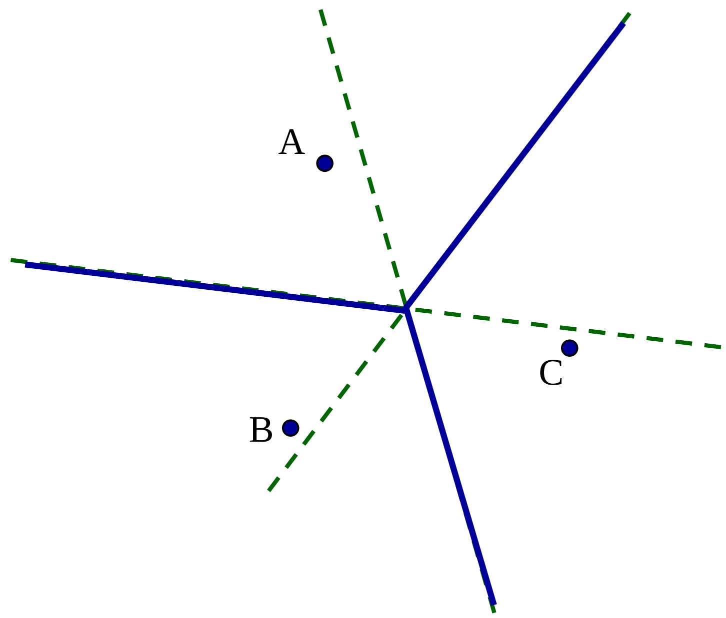



- One-versus-one scheme with a decision tree:



# Example: A 3-class linear classifier

Classify instances into three classes {A, B, C}
using three linear discriminant functions
classifying (A vs. B), (A vs. C), and (B vs. C)



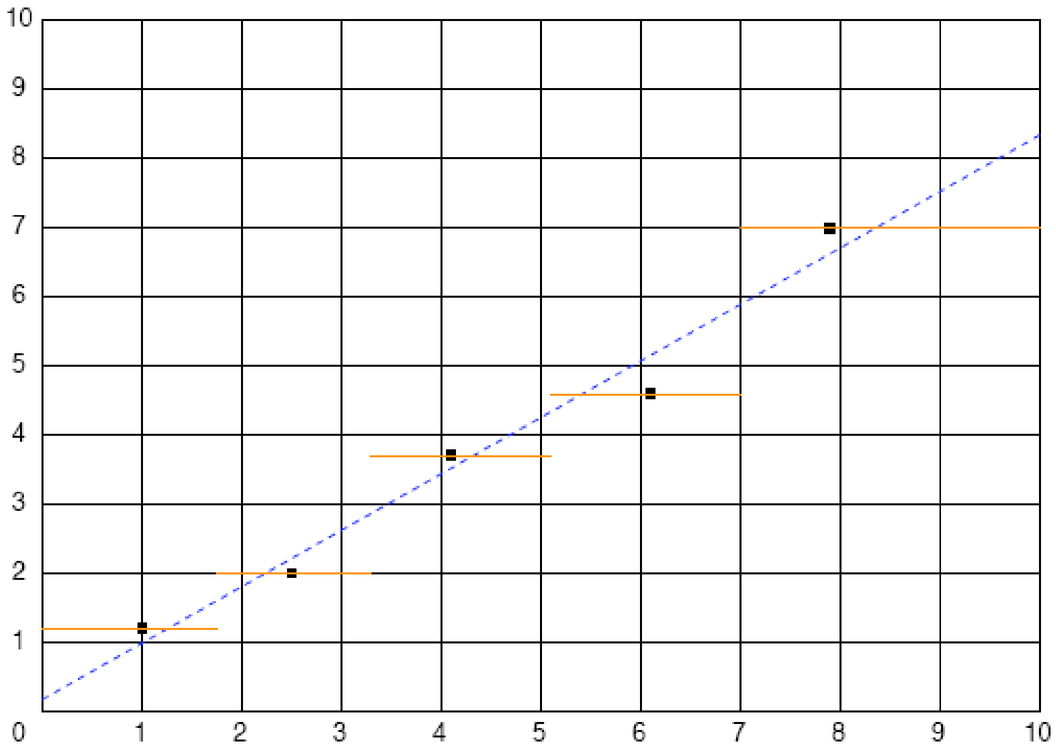
# Regression – another predictive ML task

- In the classification tasks we've been discussing, the label space was a discrete set of classes
  - Classification, scoring, ranking, probability estimation
- Regression learns a function (the regressor) that is a mapping $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$ from examples – $f(x_i)$
  - I.e., the target variable (output) is real-valued
- Assumption: the examples will be noisy, so watch out for overfitting – need to capture the general trend or shape of the function, not exactly match every data point
  - E.g., if fitting an N-degree polynomial to the training data (thus N+1 parameters to estimate), choose one as low degree as possible
- The number of data points should be much greater than the number of parameters to be estimated!
  - How much data is needed? An open question in ML….

# Regression example

Training data

| $x$ | $f(x)$ |
|---|---|
| 1.0 | 1.2 |
| 2.5 | 2.0 |
| 4.1 | 3.7 |
| 6.1 | 4.6 |
| 7.9 | 7.0 |




Piecewise linear fit

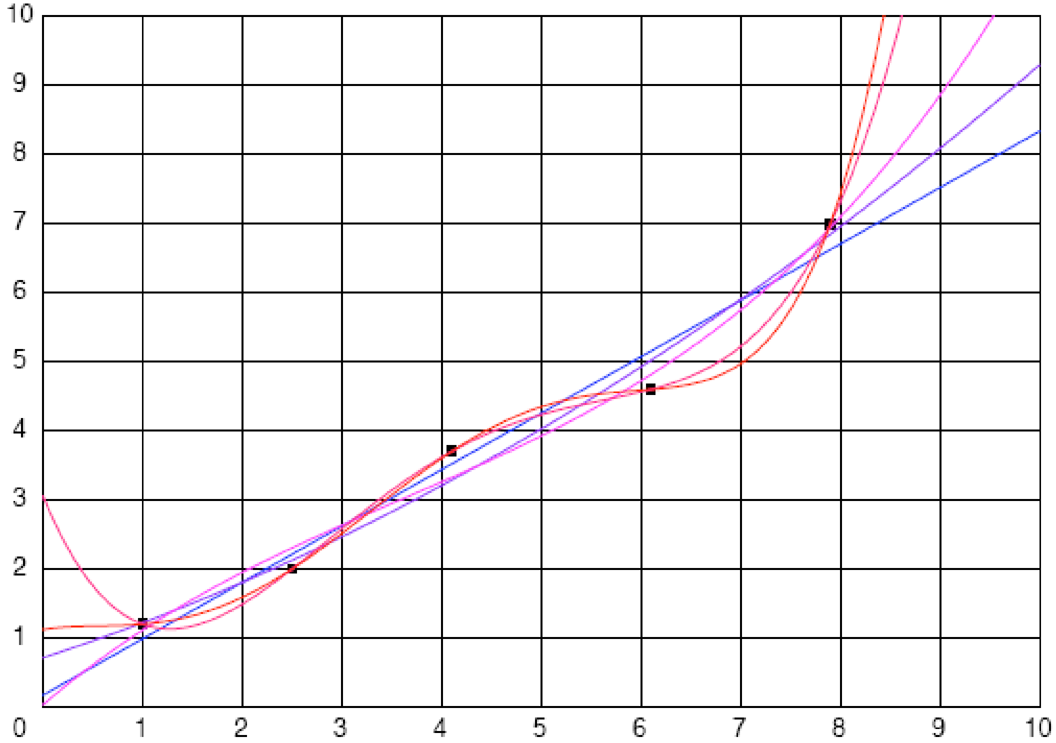
Globally linear fit

The regression function may or may not fit the training data exactly

# Regression example

Training data

| $x$ | $f(x)$ |
|---|---|
| 1.0 | 1.2 |
| 2.5 | 2.0 |
| 4.1 | 3.7 |
| 6.1 | 4.6 |
| 7.9 | 7.0 |




{1, 2, 3, 4}-order functions

# Regression

- We'll generally estimate a regression function based on some function of the *residual*, the different between the estimate and the label (the true value):

$$r(x) = f(x) - \hat{f}(x)$$

True function

Regression function

- That function is (again) the *loss function* L
  - The most common loss function for regression is the squared residual:

$$L(x) = r^2(x) = \left(f(x) - \hat{f}(x)\right)^2$$

  - However, this is sensitive to outliers (large errors have a disproportionately large effect), so often a function that minimizes large errors is used – the result is called a ***robust estimator***

# Concept Learning

Chapter 4 in the textbook

*Logical Models: tree models and rule models*

# Concept learning

- Concept learning means learning (typically binary) concepts from examples
  - The learned concept is the positive class
  - Everything else is the negative class

- We'll now use logical models – logical expressions describe concepts and divide the instance space appropriately
  - See "Background 4.1" on p. 105 in the textbook (or take CS 40 or CS 165A!) for an overview of the logical concepts and notation
  - Propositional logic
    - Logical manipulation of propositions (symbols that have values)
  - (First-order) predicate logic
    - Add variables, predicates (binary functions), functions, and variable quantification (for all x…, there exists an x such that…)

# Propositional (Boolean) logic

- Symbols represent propositions (statements of fact, sentences)
  - *P* means "San Francisco is the capital of California"
  - *Q* means "It is raining in Seattle"
  - *Length = 3* means "The value of the feature *Length* is 3"
  - *Teeth=many* means "The value of the feature *Teeth* is *many*
- Expressions are generated by combining proposition symbols with Boolean (logical) connectives
  - *True*, *false*, propositional symbols
  - *feature/value* relations – e.g., *feature* = *value*, *feature* < *value*, ...
  - ( ) , $\neg$ (not), $\wedge$ (and), $\vee$ (or), $\Rightarrow$ (implies), $\Leftrightarrow$ (equivalent)

# Propositional logic

- Semantics
  - Defined by clearly interpreted symbols and straightforward application of truth tables
  - Rules for evaluating truth: Boolean algebra
  - Simple method: truth tables

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| *False* | *False* | *True* | *False* | *False* | *True* | *True* |
| *False* | *True* | *True* | *False* | *True* | *True* | *False* |
| *True* | *False* | *False* | *False* | *True* | *False* | *False* |
| *True* | *True* | *False* | *True* | *True* | *True* | *True* |

**$2^N$ rows for N propositions**

# Concept learning

- In concept learning, we want to learn a Boolean function over a set of attributes+values
  - I.e., derive a Boolean function from training examples
    - Positive and negative examples of the concept
      - Positive: Temperature = high $\wedge$ Coughing = yes $\wedge$ Spots = yes
      - Negative: Temperature = medium $\wedge$ Coughing = no $\wedge$ Spots = yes
  - This is our hypothesis
- The target concept $\boldsymbol{c}$ is the true concept
  - We want the hypothesis to be a good estimate of the true concept
  - Thus we wish to find $\boldsymbol{h}$ (or $\hat{\boldsymbol{c}}$) such that $\boldsymbol{h} = \boldsymbol{c}$ (or $\hat{\boldsymbol{c}} = \boldsymbol{c}$)
- The hypothesis is a Boolean function over the features
  - E.g., some combinations of {Temperature, Coughing, Spots} are <u>in</u> the concept, and others are <u>not in</u> the concept

# The hypothesis space

- Using a set of features, what concepts can possibly be learned?
- The space of all possible concepts is called the hypothesis space
  - What is the hypothesis space for a given problem?
- First, how many possible instances are there for a given set of features?
  - In set theory, the Cartesian product of all the features
  - $F_1 \times F_2 \times \ldots \times F_N$
  - All combinations of feature values
  - UCSB courses: Quarter (4), Dept (40), courselevel (2), topic (500)
  - $4 \times 40 \times 2 \times 500 = 160{,}000$ possible instances
    - E.g., (spring, CS, ugrad, ML), (fall, Music, grad, StringTheory), …

# The hypothesis space

- The hypothesis space is the number of binary functions on these instances, which is… $2^{160,000}$!!
  - I.e., the number of different sets you can make from 160,000 elements
  - Or if you laid out all possible instances, the number of different contours you could draw separating some instances from the rest
  - Each of these hypotheses… sets… contours… defines a concept
- The challenge in concept learning is deciding which hypothesis is best, given the training data
  - As with all problems in machine learning, generalization is of key importance – we don't only want to memorize the training data (the overfitting problem)
  - We want to learn a concept that will generalize well to new, unseen instances

# The conjunctive hypothesis space

- Let's limit our hypothesis space to conjunctive concepts – i.e., hypotheses that can be expressed as a conjunction of literals (features)

  Quarter=? ∧ Dept=? ∧ courselevel=? ∧ topic=?

- We add "absence" or "don't care" to each feature, so now the total number of combinations is $5 \times 41 \times 3 \times 501 = 308{,}115$
  - That's a lot, but much better than $2^{160{,}000}$! (between $2^{18}$ and $2^{19}$)
- The most general hypothesis is (X, X, X, X), which includes all possible instances
  - (fall, X, X, X) is the concept of all fall quarter courses
  - (fall, CS, grad, X) is the concept of all CS graduate courses in the fall
- In this conjunctive hypothesis space, we can't represent concepts like "all courses in AI or Graphics"

# An example hypothesis space

**Target concept:**
***c*** = *Pays income taxes*

Two binary features:

| | | UCSB student | |
|---|---|---|---|
| | | Yes | No |
| Age < 21 | Yes | **A** | **B** |
| | No | **C** | **D** |

Instance space:
{Age × Student}

2 × 2 = 4 instances:

(Yes, Yes) – A (Yes, No) – B
(No, Yes) – C (No, No) – D

How many possible hypotheses are there?
$2^4$ = 16 possible hypotheses (concepts)

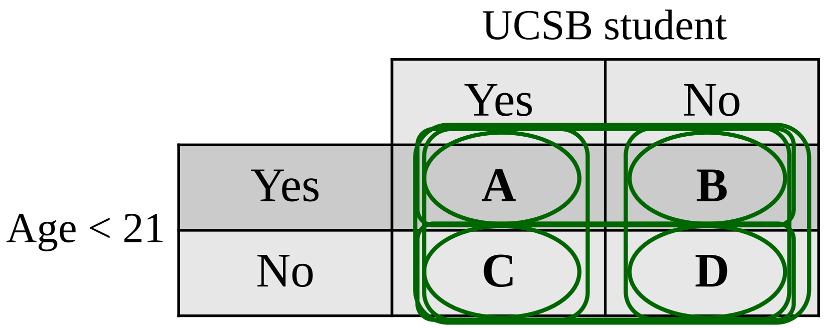
The training example (Yes, No) provides evidence for which hypotheses?

– All the ones that contain B

Now what if we observe a second training example (No, No)?

| | |
|---|---|
| {A} | {A, D} |
| ★ {B} | ★{B, C} |
| {C} | ★{B, C, D} |
| {D} | {A, C, D} |
| ★ {A, B} | ★{A, B, D} |
| {C, D} | ★{A, B, C} |
| {A, C} | ★{A, B, C, D} |
| ★ {B, D} | { } or $\phi$ |

# Our example using conjunctive hypothesis space




Instance space:
{Age × Student}

CHS: Hypotheses that can be represented as
Age={Yes, No, X} ∧ Student={Yes, No, X}

That’s 9 hypotheses:

| | | |
|---|---|---|
| (Yes, Yes) | (Yes, No) | (Yes, X) |
| (No, Yes) | (No, No) | (No, X) |
| (X, Yes) | (X, No) | (X, X) |

| | |
|---|---|
| {A} | {A, D} |
| {B} | {B, C} |
| {C} | {B, C, D} |
| {D} | {A, C, D} |
| {A, B} | {A, B, D} |
| {C, D} | {A, B, C} |
| {A, C} | {A, B, C, D} |
| {B, D} | { } or $\phi$ |

Conjunctive = combining rows and columns via AND (not by OR)

# An example of CHS learning

Suppose you come across a number of sea animals that you suspect belong to the same species. You observe their length in meters, whether they have gills, whether they have a prominent beak, and whether they have few or many teeth. The first animal can described by the following conjunction of features:

$$\text{Length} = 3 \;\wedge\; \text{Gills} = \text{no} \;\wedge\; \text{Beak} = \text{yes} \;\wedge\; \text{Teeth} = \text{many}$$

The next one has the same characteristics but is a meter longer, so you drop the length condition and generalize the conjunction to

$$\text{Gills} = \text{no} \;\wedge\; \text{Beak} = \text{yes} \;\wedge\; \text{Teeth} = \text{many}$$

The third animal is again 3 meters long, has a beak, no gills and few teeth, so your description becomes

$$\text{Gills} = \text{no} \;\wedge\; \text{Beak} = \text{yes}$$

All remaining animals satisfy this conjunction, and so your hypothesis is formed.

Someone tells you what these animals are called: Dolphins

# An example of CHS learning

We took a specific-to-general approach in coming up with a hypothesis here.

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 = X $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many

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

Length = X $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = X

# An example of CHS learning

Features and possible values:

Length = { 3, 4, 5 }

Gills = { yes, no }

Beak = { yes, no }

Teeth = { few, many }

In this problem, there are $3 \times 2 \times 2 \times 2 = 24$ possible instances and $2^{24}$ possible hypotheses over the instances (about 16.8 million)

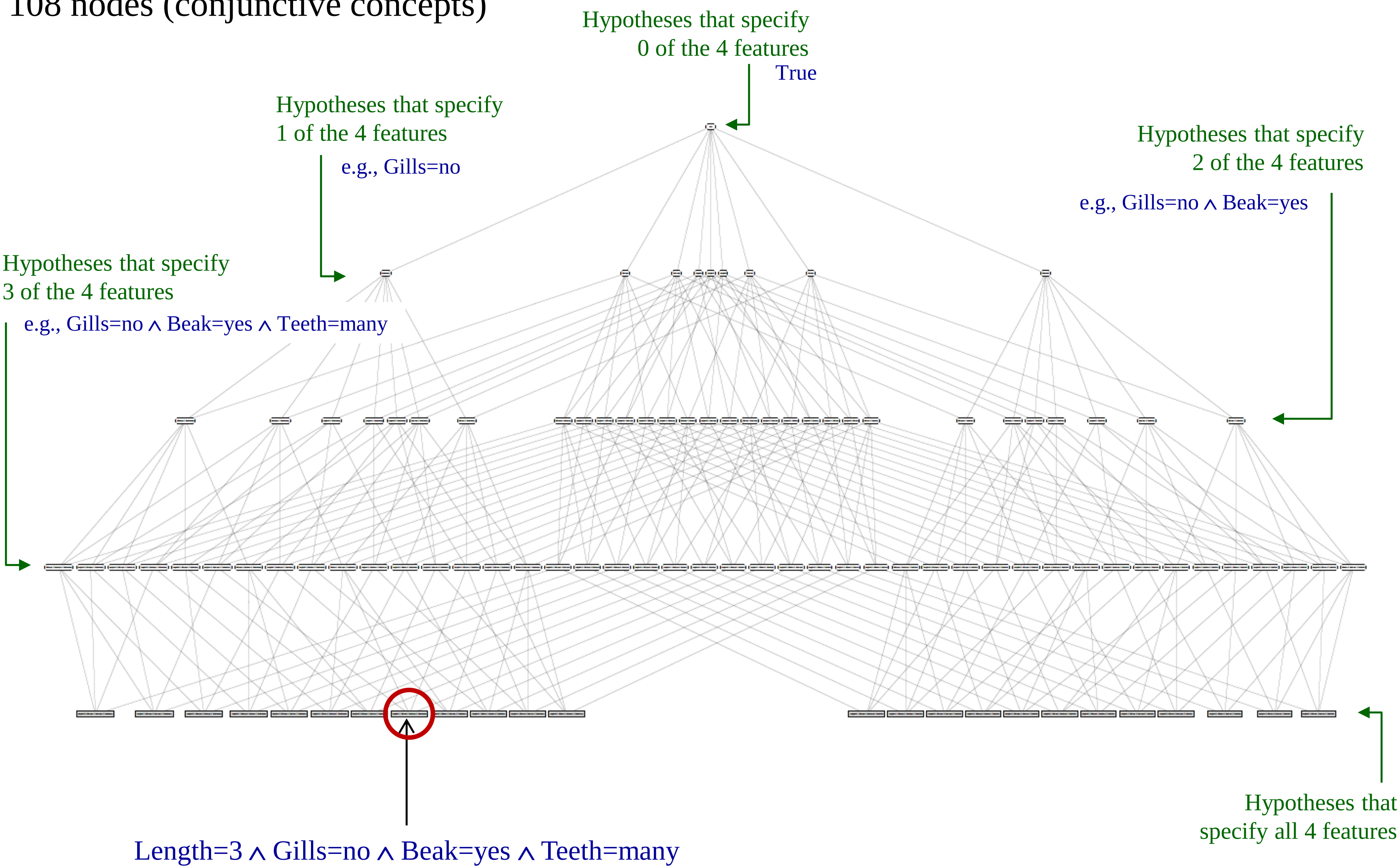
But with the conjunctive hypothesis space, we have only $4 \times 3 \times 3 \times 3 = 108$ possible conjunctive hypotheses

- In our earlier example, we went from 16 hypotheses to 9 using CHS
- Here we go from 16.8 million to 108

Let’s visualize the conjunctive hypothesis space for this problem:

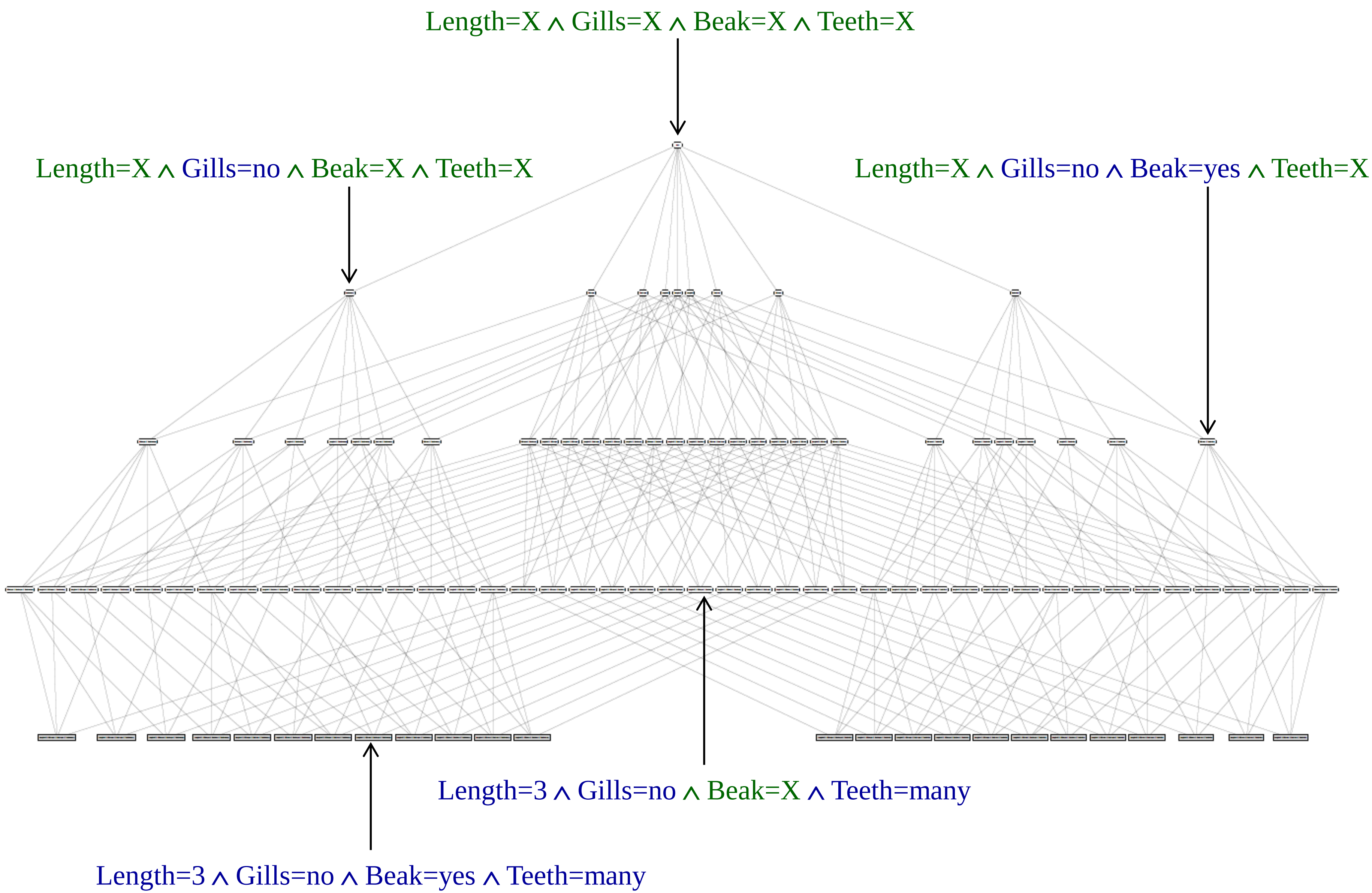
# The Conjunctive Hypothesis Space

108 nodes (conjunctive concepts)




This connects upward to every more general hypothesis that includes it

# The Conjunctive Hypothesis Space

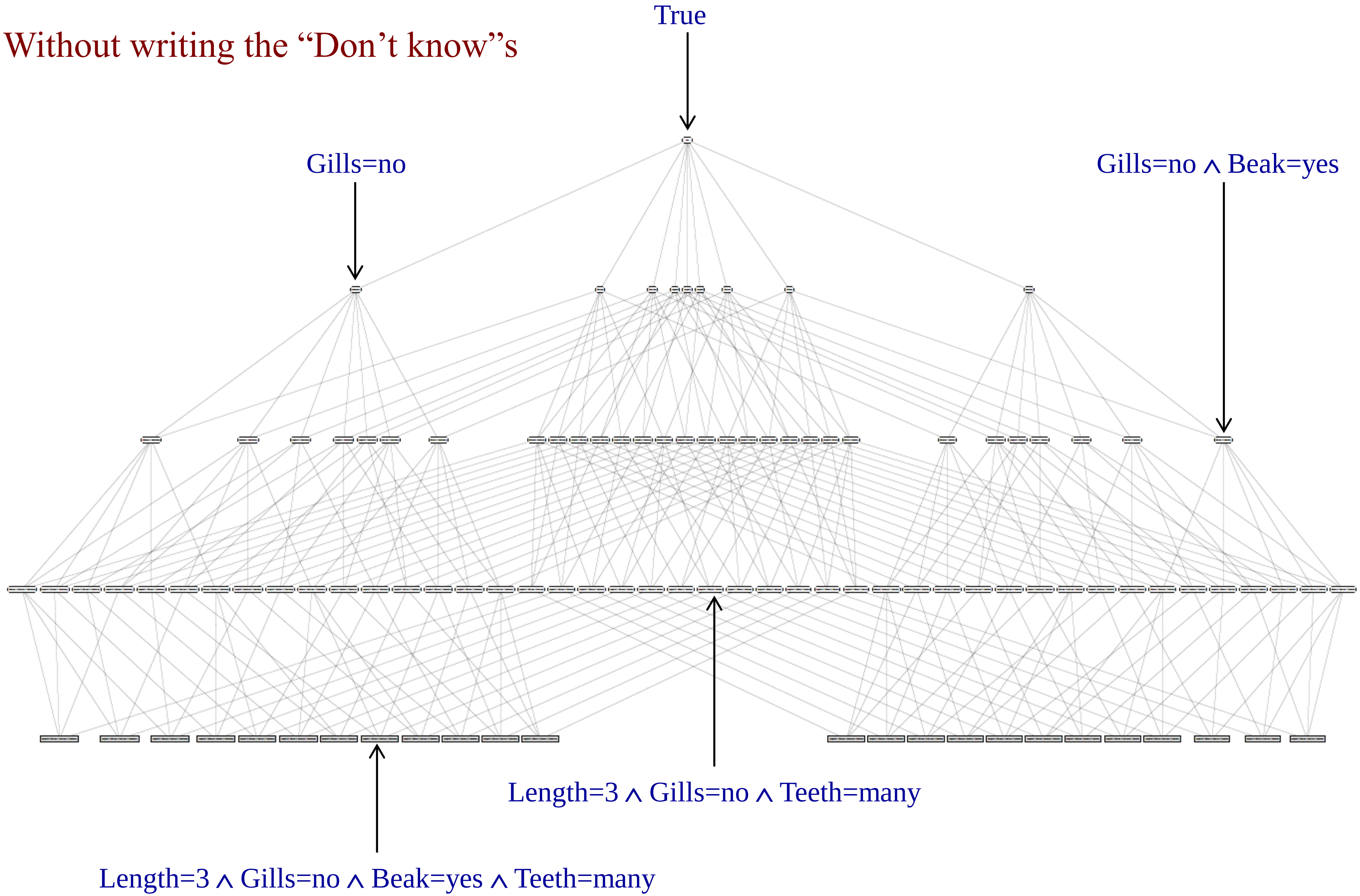



Number of “Don’t know” (X) increases by 1 each level

# The Conjunctive Hypothesis Space

Note that ∧ True can be appended to any proposition, so this is what's left when all the literals are removed

Without writing the "Don't know"s

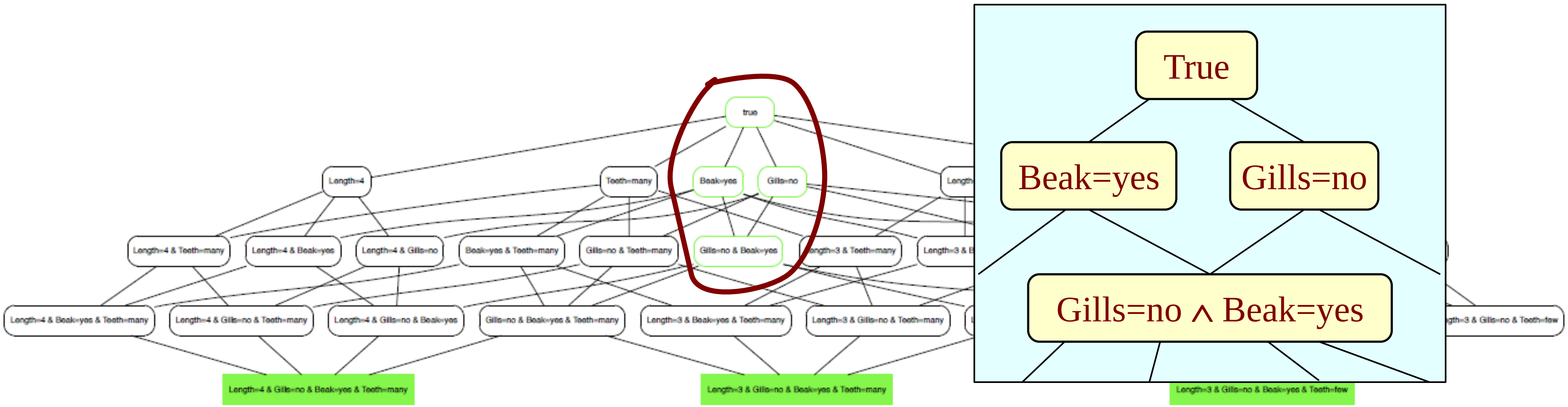


# Reducing the hypothesis space

- Given that hypothesis space and our training examples:
  - Length = 3 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many
  - Length = 4 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = many
  - Length = 3 $\wedge$ Gills = no $\wedge$ Beak = yes $\wedge$ Teeth = few
- Let's rule out all the hypotheses (concepts) that don't include at least one of the instances in our example
  - I.e., delete nodes that don't fit with at least one training example
  - This leaves us with just 32 conjunctive concepts (out of the original 108)

# Reducing the hypothesis space

- But if we require hypotheses to cover <u>all</u> three 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

$$\text{Gills} = \text{no} \ \wedge \ \text{Beak} = \text{yes}$$

# Least general generalization (LGG)

- We want to generalize beyond our specific training data, but not too much – the most general hypothesis is to accept everything
- Thus we'd like the *least general* generalization
  - General enough to include all of our training data, but no more general than that
- Referring to our classification terminology, the more general our hypothesis, the lower our…
  - False negative rate      H($x$) = True
- And the less general (more specific) our hypothesis, the lower our…
  - False positive rate      H($x$) = False
- So, our approach to generating a hypothesis/concept should depend on the needs of our application