CSEP 546
Data Mining

Instructor: Jesse Davis

Today's Program

- Logistics and introduction
- Inductive learning overview
- Instance-based learning
- Collaborative filtering (Homework 1)

Logistics

- **Instructor:** Jesse Davis
  - Email: jdavis@cs [Please include 546 in subject]
  - Office: CSE 356
  - Office hours: Mondays 5:30-6:20
- **TA:** Andrey Kolobov
  - Email: akolobov@cs [Please include 546 in subject]
  - Office: TBD
  - Office hours: Mondays 5:30-6:20
- **Web:** www.cs.washington.edu/p546
- **Mailing list**: csep546@cs

Assignments

- **Four homeworks**
  - Individual
  - Mix of questions and programming (to be done in either java or c++)
  - 10% penalty per each day late (max of 5 days late)

Assignments

- Homework 1: Due April 12th (100 points)
  - Collaborative filtering, IBL, d-trees and methodology
- Homework 2: Due April 26th (100 points)
  - NB for spam filtering, rule learning, BNs
- Homework 3: Due May 10th (100 points)
  - Perceptron for spam filtering, NNs, ensembles, GAs
- Homework 4: Due June 1st (135-150 points)
  - Weka for empirical comparison, clustering, learning theory, association rules

Source Materials

- Tom Mitchell, *Machine Learning,* McGraw-Hill, 1997.
- R. Duda, P. Hart & D. Stork, Pattern Classification (2nd ed.), Wiley, 2001 (recommended)
- Papers
  - Will be posted on the course Web page

Course Style

- Primarily algorithmic & experimental
- Some theory, both mathematical & conceptual (much on statistics)
- "Hands on" experience, interactive lectures/discussions
- Broad survey of many data mining/machine learning subfields

Course Goals

- Understand what a data mining or machine learning system should do
- Understand how current systems work
  - Algorithmically
  - Empirically
  - Their shortcomings
- Try to think about how we could improve algorithms

Background Assumed

- Programming languages
  - Java or C++
- AI Topics
  - Search, first-order logic
- Math
  - Calculus (i.e., partial derivatives) and simple probability (e.g., prob(A | B)
- Assume no data mining or machine learning background (some overlap with CSEP 573)

What is Data Mining?

- **Data mining** is the process of identifying valid, novel, useful and understandable patterns in data
- Also known as **KDD** (**K**nowledge **D**iscovery in **D**atabases)
- "We're drowning in information, but starving for knowledge." (John Naisbett)

Related Disciplines

- Machine learning
- Databases
- Statistics
- Information retrieval
- Visualization
- High-performance computing
- Etc.

Applications of Data Mining

- E-commerce
- Marketing and retail
- Finance
- Telecoms
- Drug design
- Process control
- Space and earth sensing
- Etc.

The Data Mining Process

- Understanding domain, prior knowledge, and goals
- Data integration and selection
- Data cleaning and pre-processing
- Modeling and searching for patterns
- Interpreting results
- Consolidating and deploying discovered knowledge
- Loop

Data Mining Tasks

- Classification
- Regression
- Probability estimation
- Clustering
- Association detection
- Summarization
- Trend and deviation detection
- Etc.

Requirements for a Data Mining System

- Data mining systems should be
  - Computationally sound
  - Statistically sound
  - Ergonomically sound

Components of a Data Mining System

- Representation
- Evaluation
- Search
- Data management
- User interface

Focus of this course

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Search

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained search
  - E.g.: Linear programming

Topics for this Quarter (Slide 1 of 2)

- Inductive learning
- Instance based learning
- Decision trees
- Empirical evaluation
- Rule induction
- Bayesian learning
- Neural networks

Topics for this Quarter (Slide 2 of 2)

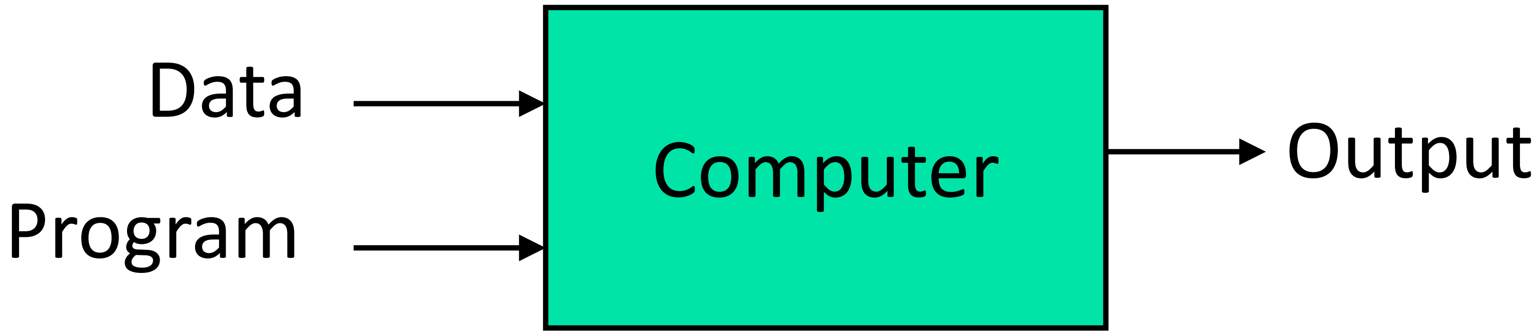
- Genetic algorithms
- Model ensembles
- Learning theory
- Association rules
- Clustering
- Advanced topics, applications of data mining and machine learning

Inductive Learning

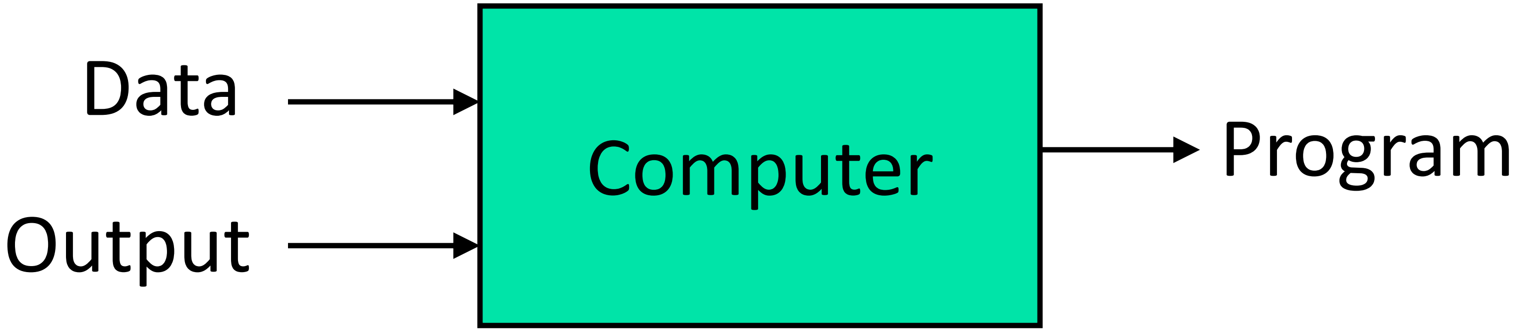
A Few Quotes

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)
- “Machine learning is the next Internet” (Tony Tether, Director, DARPA)
- Machine learning is the hot new thing” (John Hennessy, President, Stanford)
- “Web rankings today are mostly a matter of machine learning” (Prabhakar Raghavan, Dir. Research, Yahoo)
- “Machine learning is going to result in a real revolution” (Greg Papadopoulos, CTO, Sun)

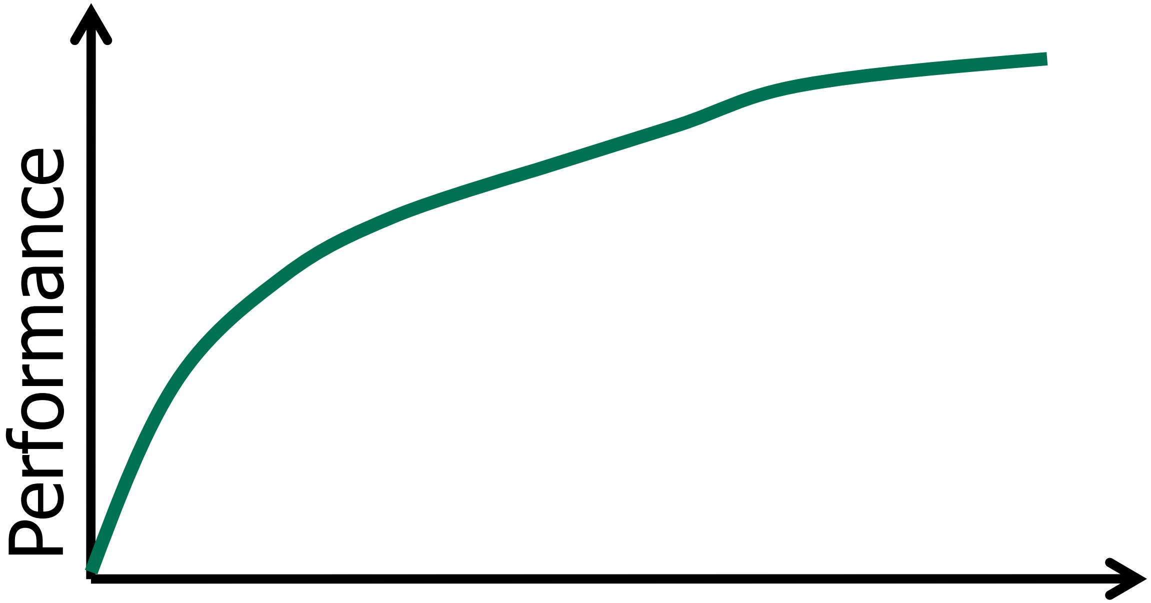
Traditional Programming



Machine Learning



What is Learning



Experience
e.g.: amount of training data, time, etc.

Defining a Learning Problem

- A program learns from experience E with respect to task T and performance measure P, if its performance at task T, as measured by P, improves with experience E

- Example:
  - Task: Play checkers
  - Performance: % of games won
  - Experience: Play games against itself

Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Inductive Learning

- **Inductive learning** or **Prediction**:
  - **Given:** Examples of a function *(X, F(X))*
  - **Predict:** Function *F(X)* for new examples *X*

- Discrete *F(X):* Classification
- Continuous *F(X):* Regression
- *F(X)* = Probability*(X):* Probability estimation

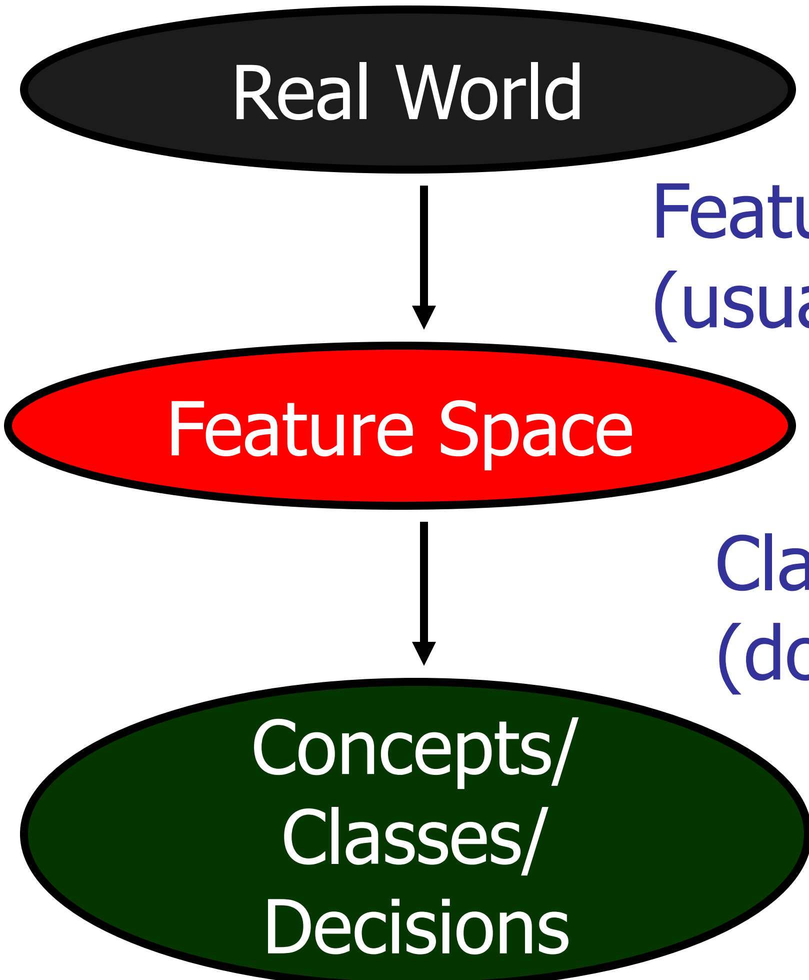
Example Applications

- Disease diagnosis
  - x: Properties of patient (e.g., symptoms, lab test results)
  - f(x): Predict disease
- Automated steering
  - x: Bitmap picture of road in front of car
  - f(x): Degrees to turn the steering wheel
- Credit risk assessment
  - x: Customer credit history and proposed purchase
  - f(x): Approve purchase or not

Widely-used Approaches

- Decision trees
- Rule induction
- Bayesian learning
- Neural networks
- Genetic algorithms
- Instance-based learning
- Etc.

Supervised Learning Task Overview



Feature construction and selection
(usually done by humans)

Classification rule construction
(done by learning algorithm)

Apply model to unseen data

Task Definition

- Given:
  - Set of positive examples of a concept/class/category
  - Set of negative examples (possibly)
- Produce:
  - A description that covers
    - All/many positive examples
    - None/few negative examples
  - Goal: Properly categorizes most future examples!

The Key Point!

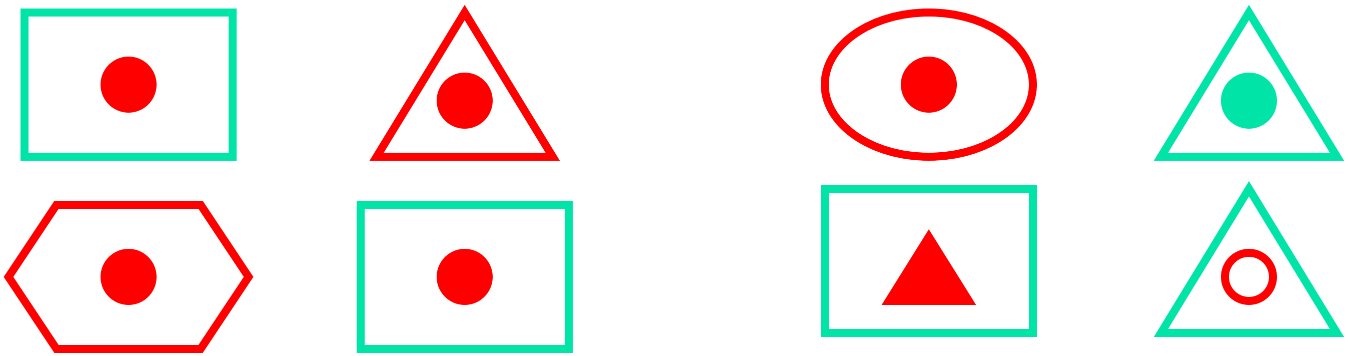
Note: one can easily extend this definition to handle more than two classes

Learning from Labeled Examples

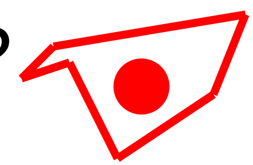
- Most successful form of inductive learning
- Given a set of data of the form: <x, f(x)>
  - x is a set of features
  - f(x) is the label for x
  - f is an unknown function
- Learn: f' which approximates f

Example

Positive Examples | Negative Examples



How do we classify this symbol?



- Concept
  - Solid Red Circle in a (Regular?) Polygon
- What about?
  - Figures on left side of page
  - Figures drawn before 5pm 3/29/89 <etc>

Assumptions

- We are assuming examples are IID: *independently identically distributed*

- We are ignoring *temporal* dependencies (covered in *time-series learning*)

- We assume the learner has no say in which examples it gets (covered in *active learning*)

Design Choices for Inductive Learners

- Need a language to represent each example (i.e., the training data)
- Need a language to represent the learned "concept" or "hypothesis"
- Need an algorithm to construct a hypothesis consistent with the training data
- Need a method to label new examples

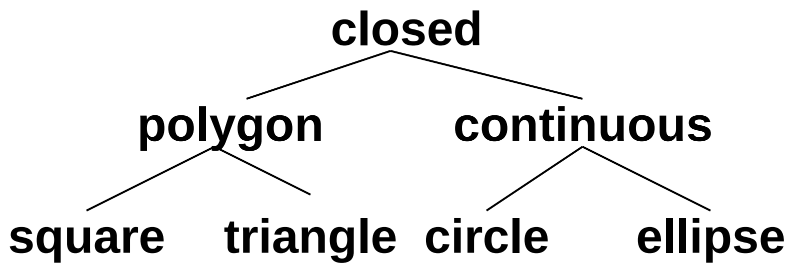
Focus of much of this course. Each choice effects the expressivity/efficiency of the algorithm

Constructing a Dataset

- Step 1: Choose a feature space
  - Common approach: Fixed length feature vector
    - Choose N features
    - Each feature has V_i possible values
    - Each example is represented by a vector of *N* feature values (i.e., is a point in the feature space)
      e.g.: <red, 50, round>
      color weight shape
  - Feature types
    - Boolean
    - Nominal
    - Ordered
    - Hierarchical
- Step 2: Collect examples (i.e., "I/O" pairs)

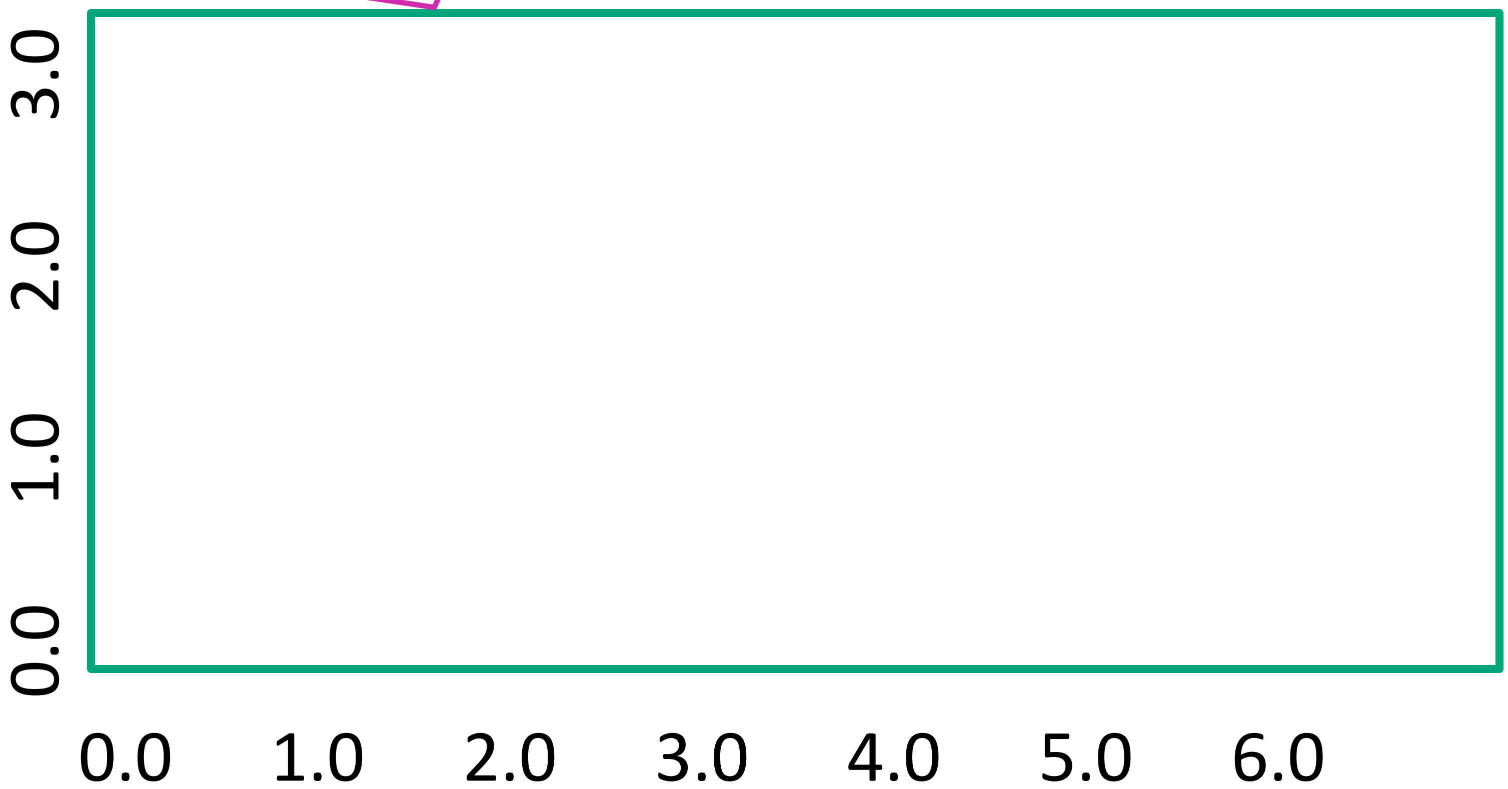
Types of Features

- Nominal: No relationship between values
  - For example: color = {red, green, blue}
- Linear/Ordered: Feature values are ordered
  - Continuous: Weight = {1,…,400}
  - Discrete: Size = {small, medium, large}
- Hierarchical: Partial ordering according to an ISA relationship



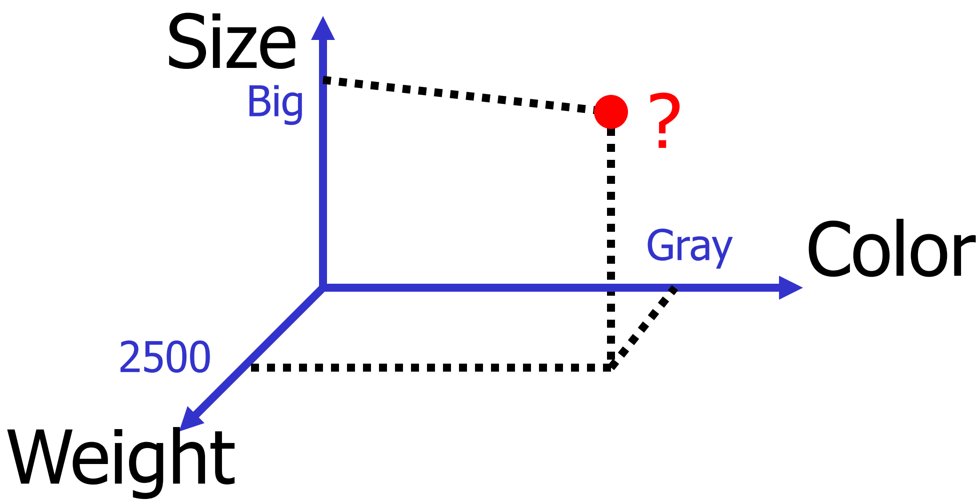
Terminology

Feature Space:
Properties that describe the problem



Another View of Feature Space

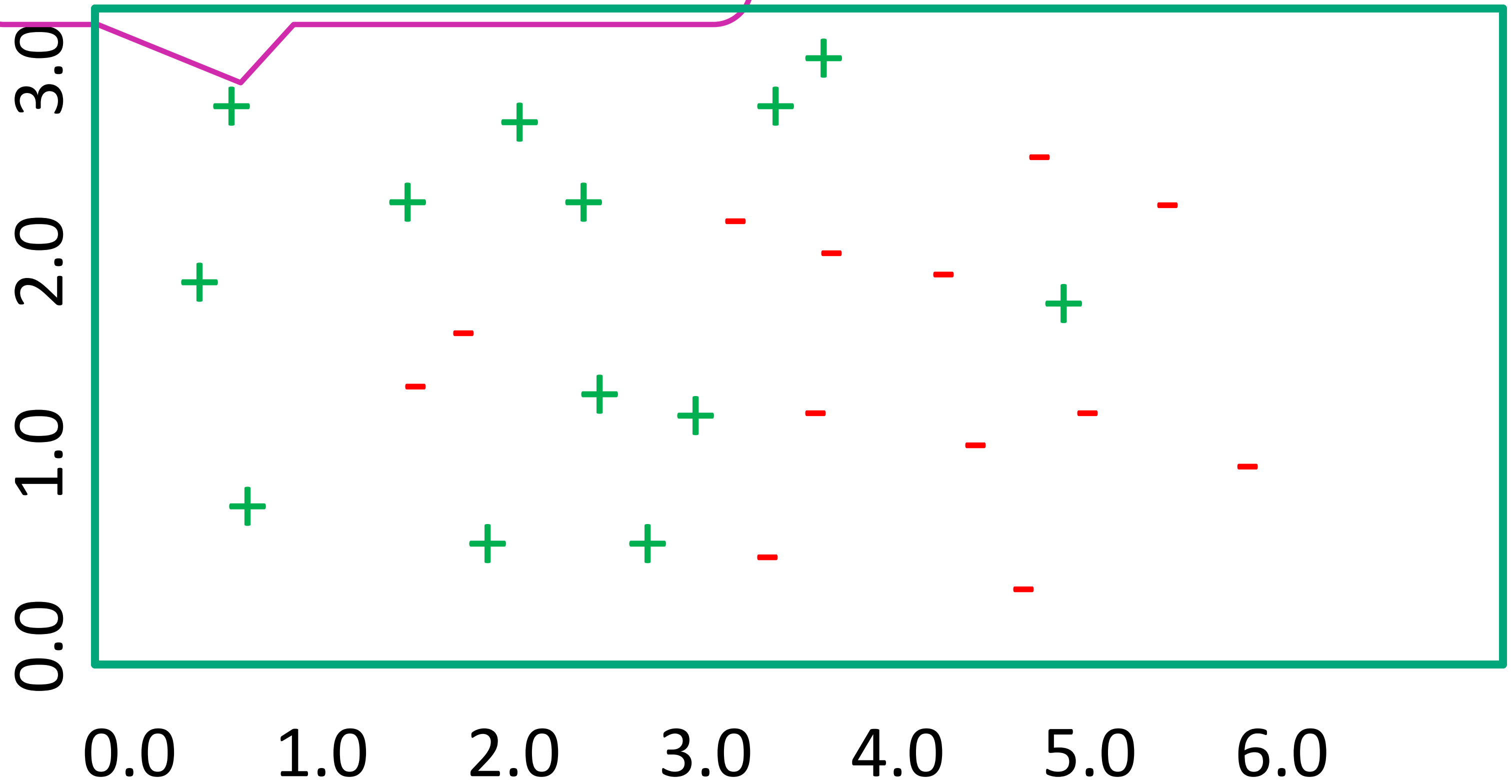
- Plot examples as points in an *N*-dimensional space



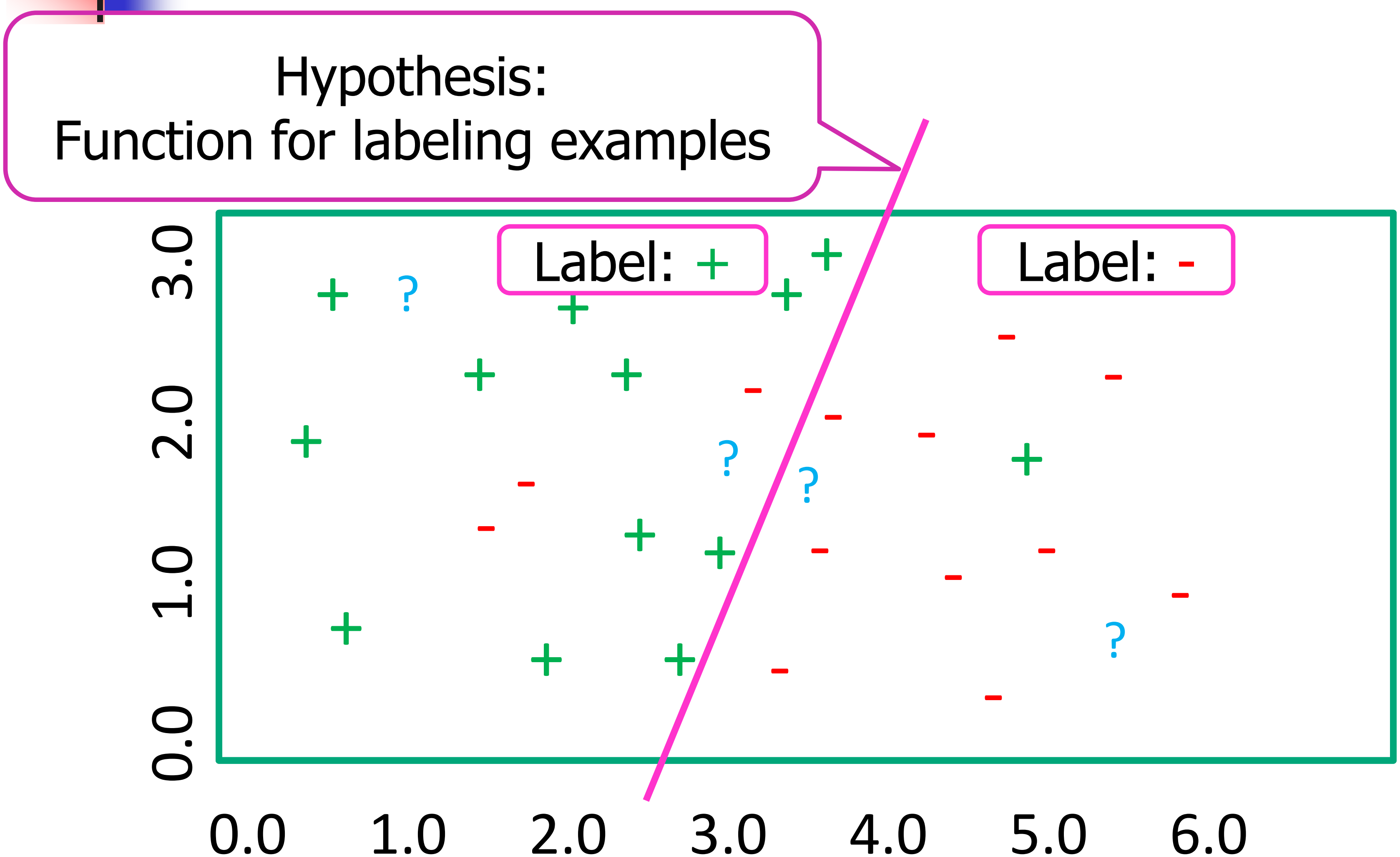
A "concept" is then a (possibly disjoint) *volume* in this space.

Terminology

Example or instance:
<0.5,2.8,+>

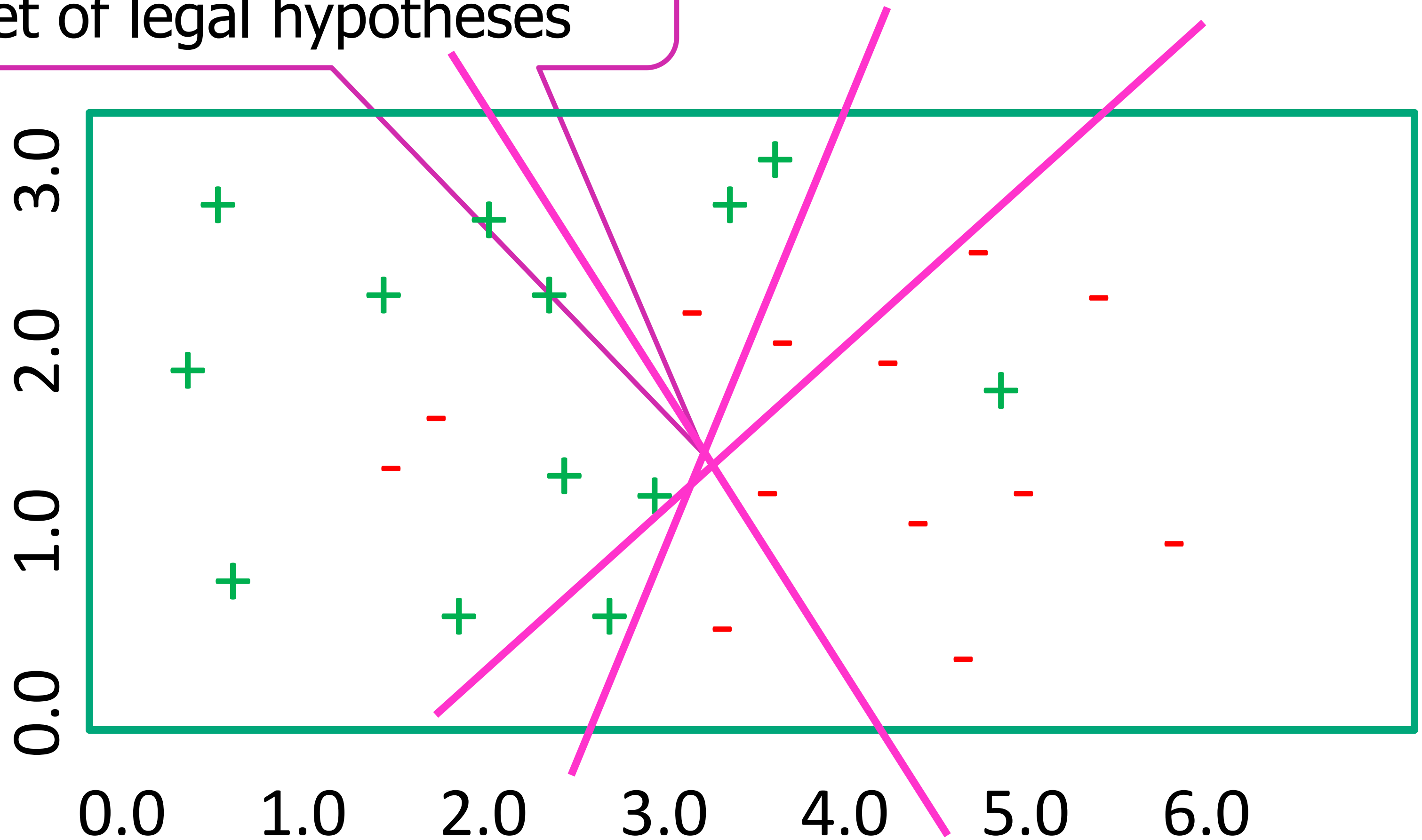


Terminology



Terminology

Hypothesis Space:
Set of legal hypotheses



Terminology Overview

- **Training example:** Data point of the form <x, f(x)>
- **Target function (concept):** the true f
- **Hypothesis (or model):** A proposed function h, believed to be similar to f
- **Concept:** A Boolean function
  - Examples where f(x) = 1 are called positive examples or positive instances
  - Examples where f(x) = 0 are called negative examples or negative instances

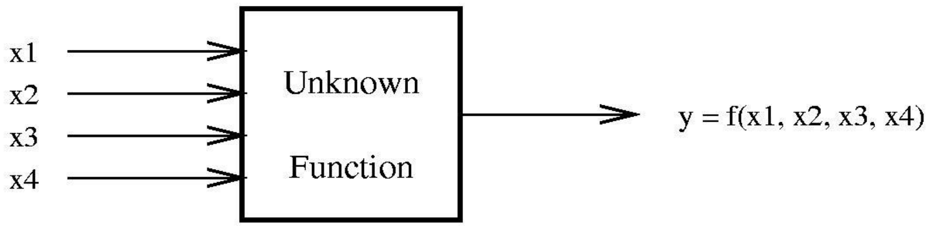
Terminology Overview

- **Classifier:** A discrete-valued function f {1,…,K}
  - Each of 1,…,K are called classes or labels
- **Hypothesis space:** The space of all hypotheses that can be output by the learner
- **Version space:** The set of all hypotheses (in the hypothesis space) that haven't been ruled by the training data

Example

- Consider IMDB as a problem.
- Work in groups for 5 minutes
- Think about
  - What tasks could you perform?
    - E.g., predict genre, predict how much the movie will gross, etc.
  - What features are relevant

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Hypothesis Spaces

- **Complete Ignorance.** There are $2^{16} = 65536$ possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 7 examples, we still have 2^9 possibilities.

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Inductive Bias

- Need to make assumptions
  - Experience alone doesn't allow us to make conclusions about unseen data instances
- Two types of bias:
  - Restriction: Limit the hypothesis space (e.g., look at rules)
  - Preference: Impose ordering on hypothesis space (e.g., more general, consistent with data)

Hypothesis Spaces (2)

- **Simple Rules.** There are only 16 simple conjunctive rules.

Rule	Counterexample
$\Rightarrow y$	1
$x_1 \Rightarrow y$	3
$x_2 \Rightarrow y$	2
$x_3 \Rightarrow y$	1
$x_4 \Rightarrow y$	7
$x_1 \wedge x_2 \Rightarrow y$	3
$x_1 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \Rightarrow y$	3
$x_2 \wedge x_4 \Rightarrow y$	3
$x_3 \wedge x_4 \Rightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3

Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

No simple rule explains the data. The same is true for simple clauses.

Hypothesis Space (3)

- m-**of**-n **rules**. There are 32 possible rules (includes simple conjunctions and clauses).

	Counterexample			
variables	1-of	2-of	3-of	4-of
$\{x_1\}$	3	–	–	–
$\{x_2\}$	2	–	–	–
$\{x_3\}$	1	–	–	–
$\{x_4\}$	7	–	–	–
$\{x_1, x_2\}$	3	3	–	–
$\{x_1, x_3\}$	4	3	–	–
$\{x_1, x_4\}$	6	3	–	–
$\{x_2, x_3\}$	2	3	–	–
$\{x_2, x_4\}$	2	3	–	–
$\{x_3, x_4\}$	4	4	–	–
$\{x_1, x_2, x_3\}$	1	3	3	–
$\{x_1, x_2, x_4\}$	2	3	3	–
$\{x_1, x_3, x_4\}$	1	***	3	–
$\{x_2, x_3, x_4\}$	1	5	3	–
$\{x_1, x_2, x_3, x_4\}$	1	5	3	3

Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

$x_1 \Rightarrow y$

$x_3 \Rightarrow y$

$x_4 \Rightarrow y$

Two Views of Learning

- **Learning is the removal of our remaining uncertainty**. Suppose we *knew* that the unknown function was an m-of-n boolean function, then we could use the training examples to infer which function it is.
- **Learning requires guessing a good, small hypothesis class**. We can start with a very small class and enlarge it until it contains an hypothesis that fits the data.

We could be wrong!

- **Our prior knowledge might be wrong**
- **Our guess of the hypothesis class could be wrong**
  The smaller the hypothesis class, the more likely we are wrong.

Example: $x_4 \wedge Oneof\{x_1, x_3\} \Rightarrow y$ is also consistent with the training data.

Example: $x_4 \wedge \neg x_2 \Rightarrow y$ is also consistent with the training data.

If either of these is the unknown function, then we will make errors when we are given new x values.

Key Issues in Machine Learning

- **What are good hypothesis spaces?**
  Which spaces have been useful in practical applications and why?
- **What algorithms can work with these spaces?**
  Are there general design principles for machine learning algorithms?
- **How can we optimize accuracy on future data points?**
  This is sometimes called the "problem of overfitting".
- **How can we have confidence in the results?**
  How much training data is required to find accurate hypotheses? (the *statistical question*)
- **Are some learning problems computationally intractable?**
  (the *computational question*)
- **How can we formulate application problems as machine learning problems?** (the *engineering question*)

A Framework for Hypothesis Spaces

- **Size**. Does the hypothesis space have a **fixed size** or **variable size**?
  Fixed-size spaces are easier to understand, but variable-size spaces are generally more useful. Variable-size spaces introduce the problem of overfitting.
- **Randomness**. Is each hypothesis **deterministic** or **stochastic**?
  This affects how we evaluate hypotheses. With a deterministic hypothesis, a training example is either *consistent* (correctly predicted) or *inconsistent* (incorrectly predicted). With a stochastic hypothesis, a training example is *more likely* or *less likely*.
- **Parameterization**. Is each hypothesis described by a set of **symbolic** (discrete) choices or is it described by a set of **continuous** parameters? If both are required, we say the hypothesis space has a **mixed** parameterization.
  Discrete parameters must be found by combinatorial search methods; continuous parameters can be found by numerical search methods.

Two Strategies for Machine Learning

- **Develop Languages for Expressing Prior Knowledge:** Rule grammars and stochastic models.
- **Develop Flexible Hypothesis Spaces:** Nested collections of hypotheses. Decision trees, rules, neural networks, cases.

In either case:

- **Develop Algorithms for Finding an Hypothesis that Fits the Data**

A Framework for Learning Algorithms

- **Search Procedure**.

  Direction Computation: solve for the hypothesis directly.

  Local Search: start with an initial hypothesis, make small improvements until a local optimum.

  Constructive Search: start with an empty hypothesis, gradually add structure to it until local optimum.

- **Timing**.

  Eager: Analyze the training data and construct an explicit hypothesis.

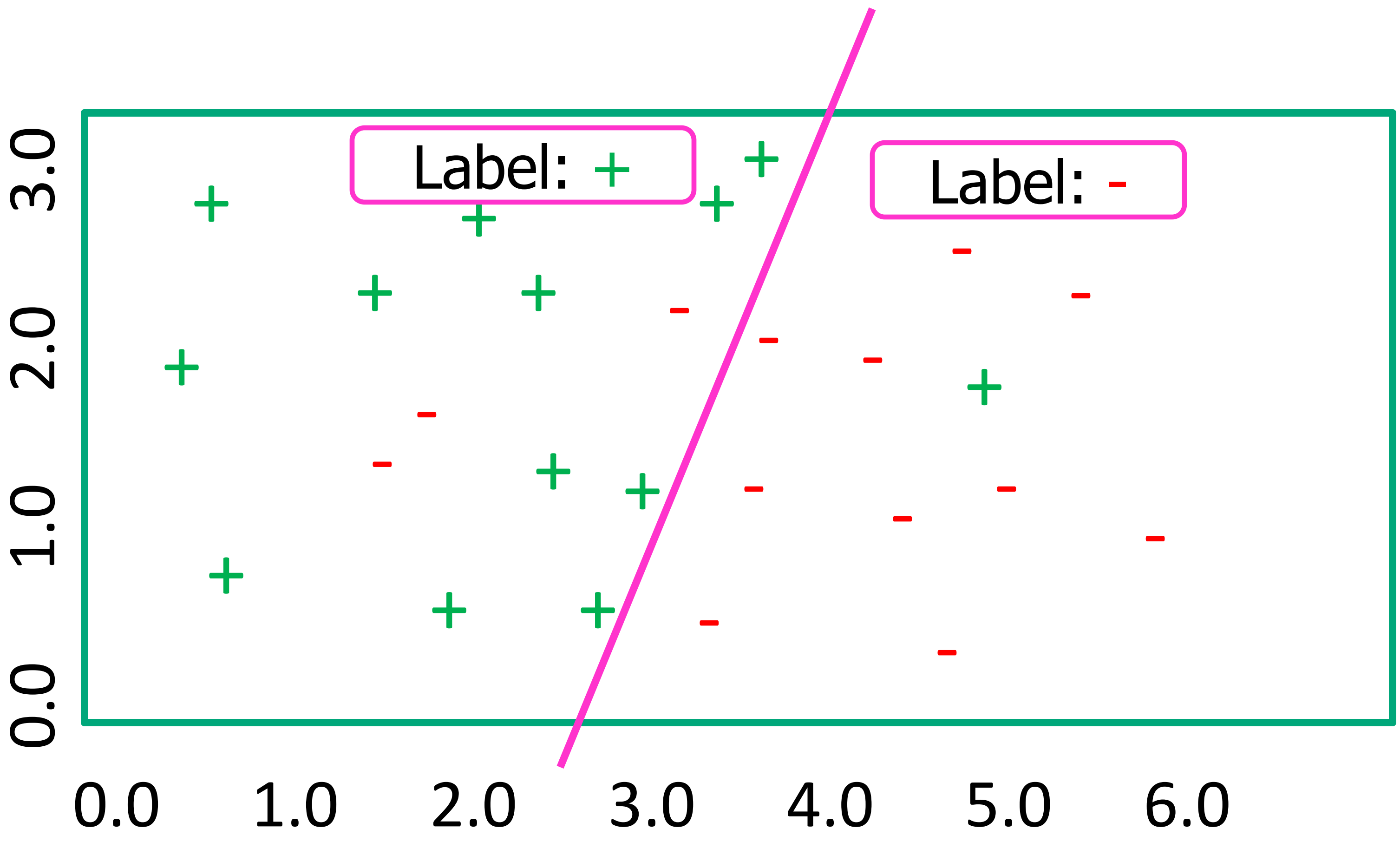
  Lazy: Store the training data and wait until a test data point is presented, then construct an ad hoc hypothesis to classify that one data point.

- **Online vs. Batch**. (for eager algorithms)

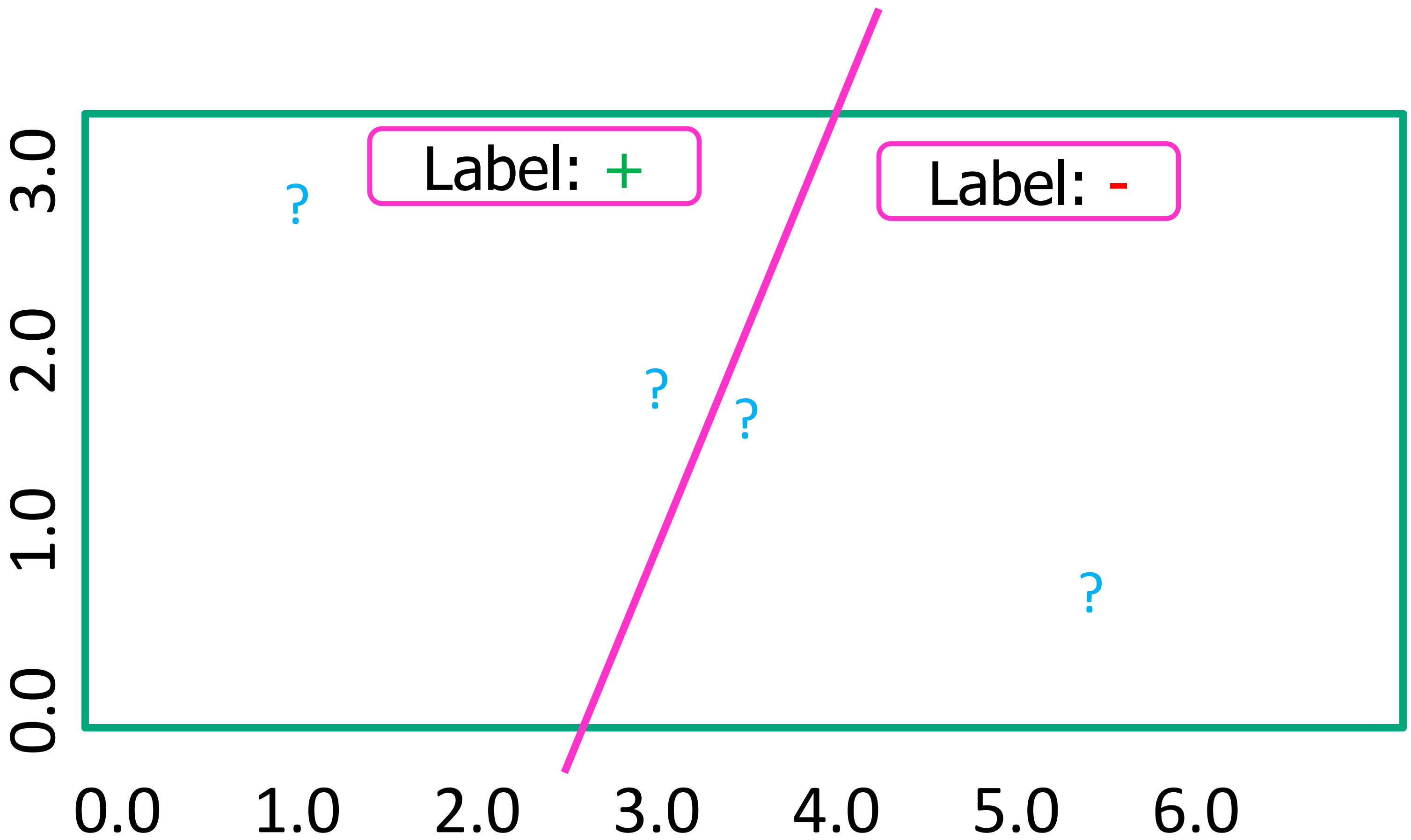
  Online: Analyze each training example as it is presented.

  Batch: Collect training examples, analyze them, output an hypothesis.

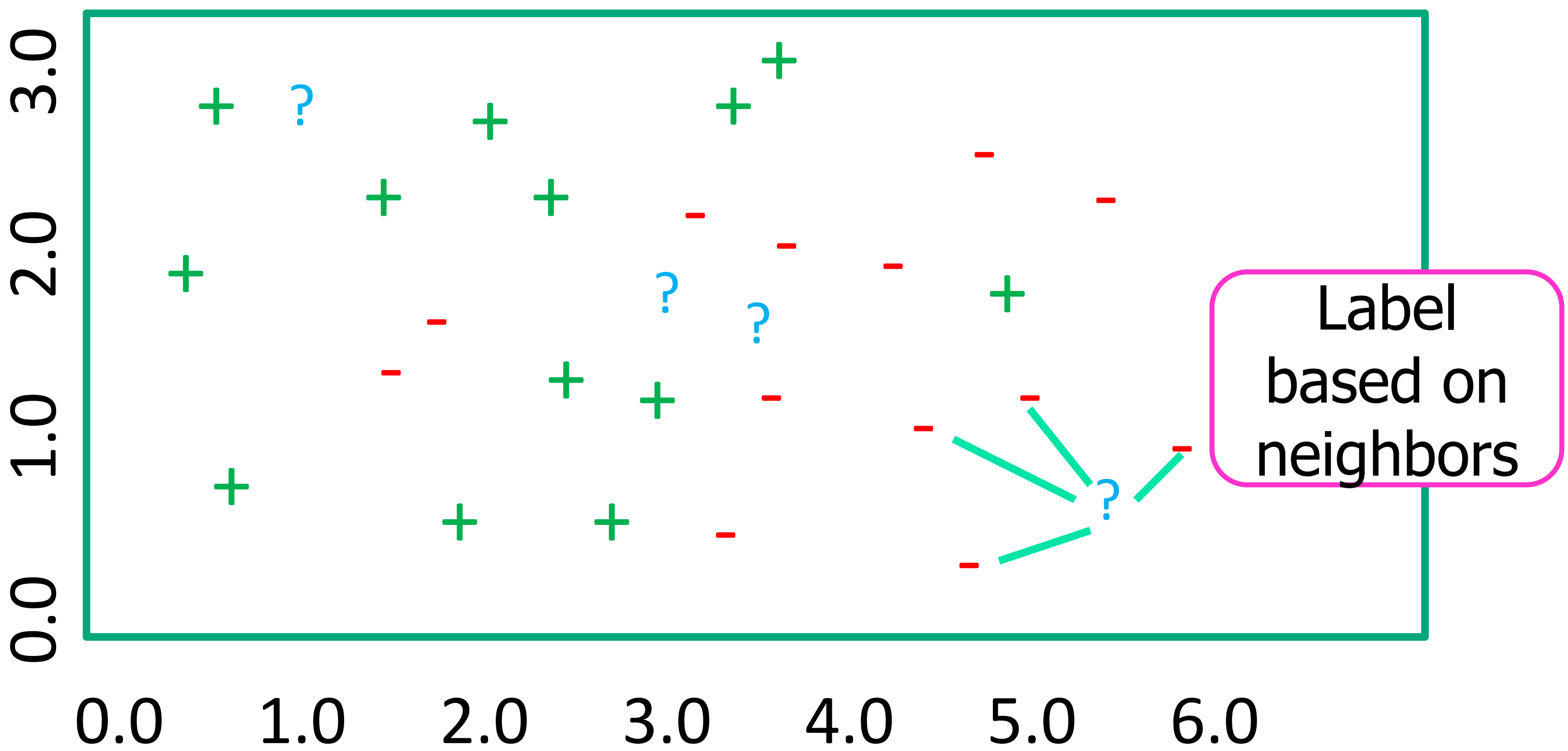
Eager



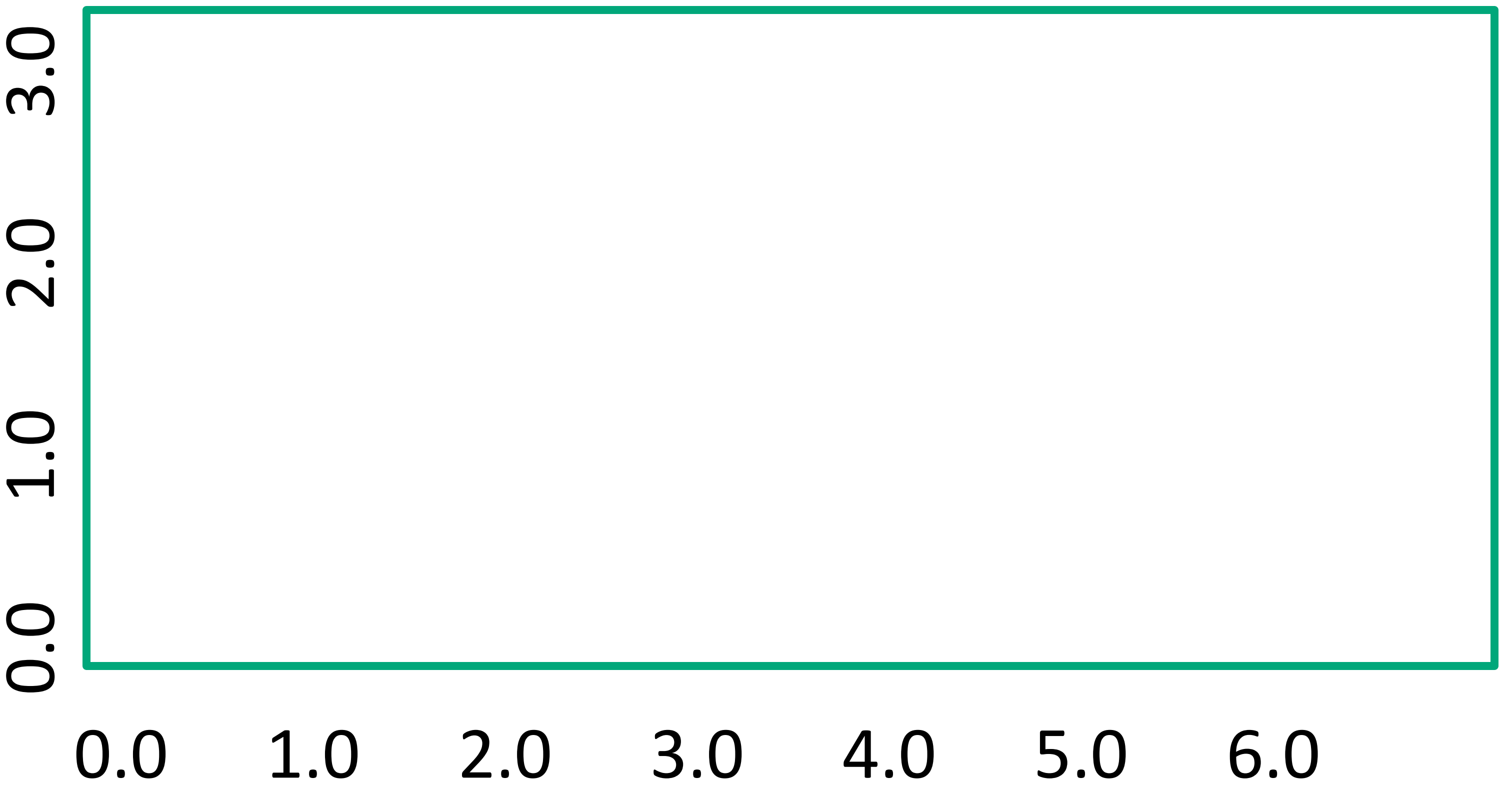
Eager



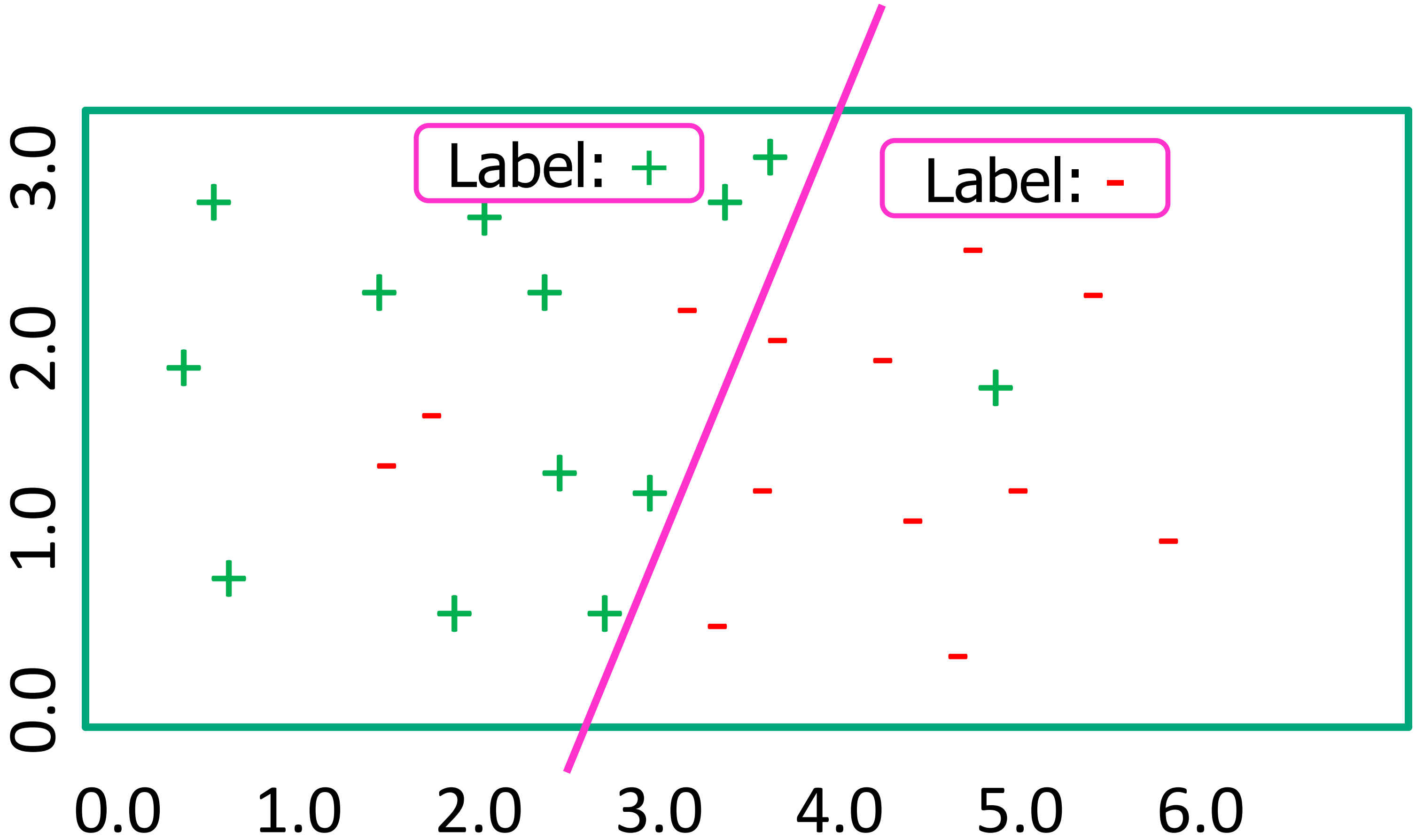
Lazy



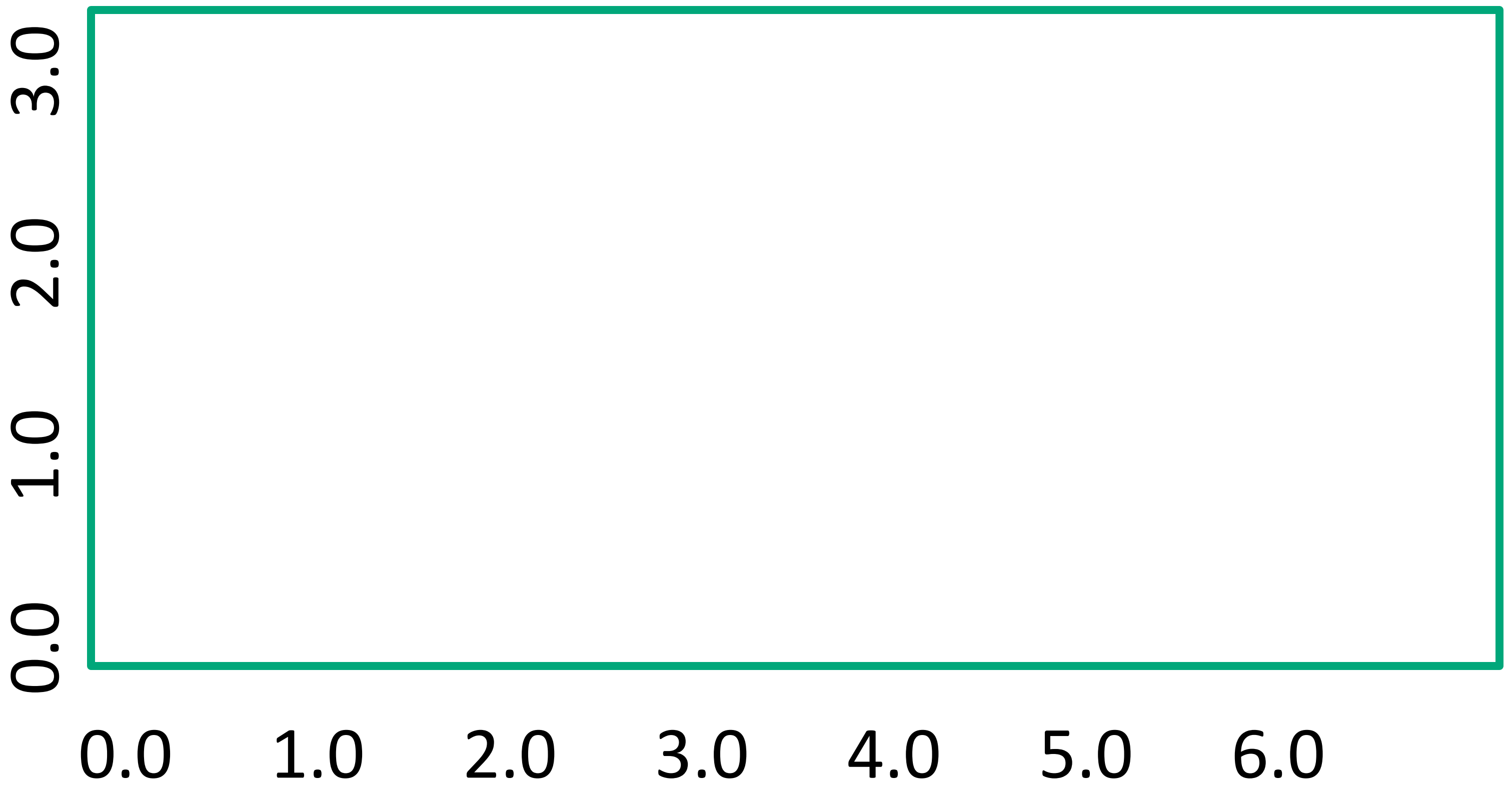
Batch



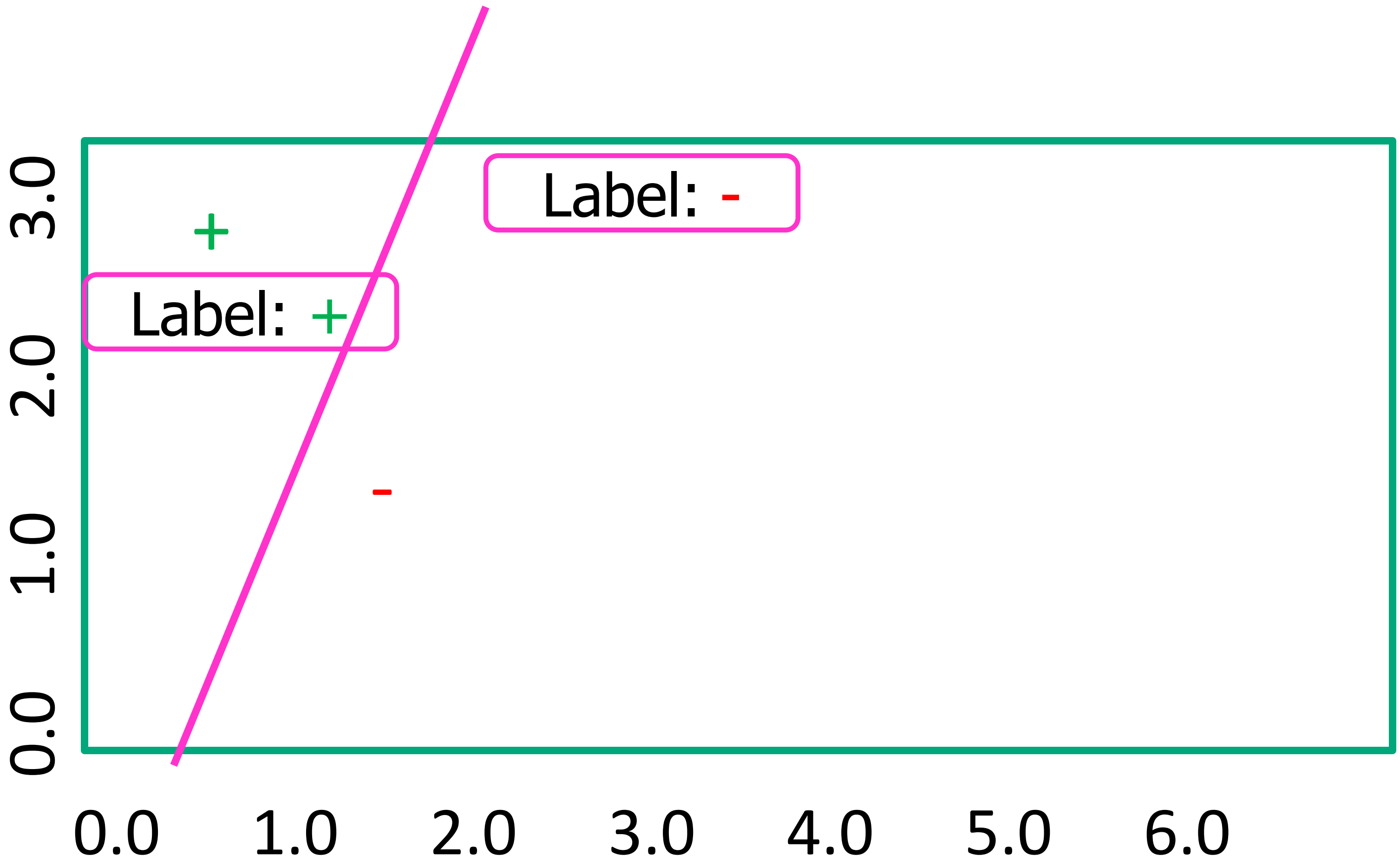
Batch



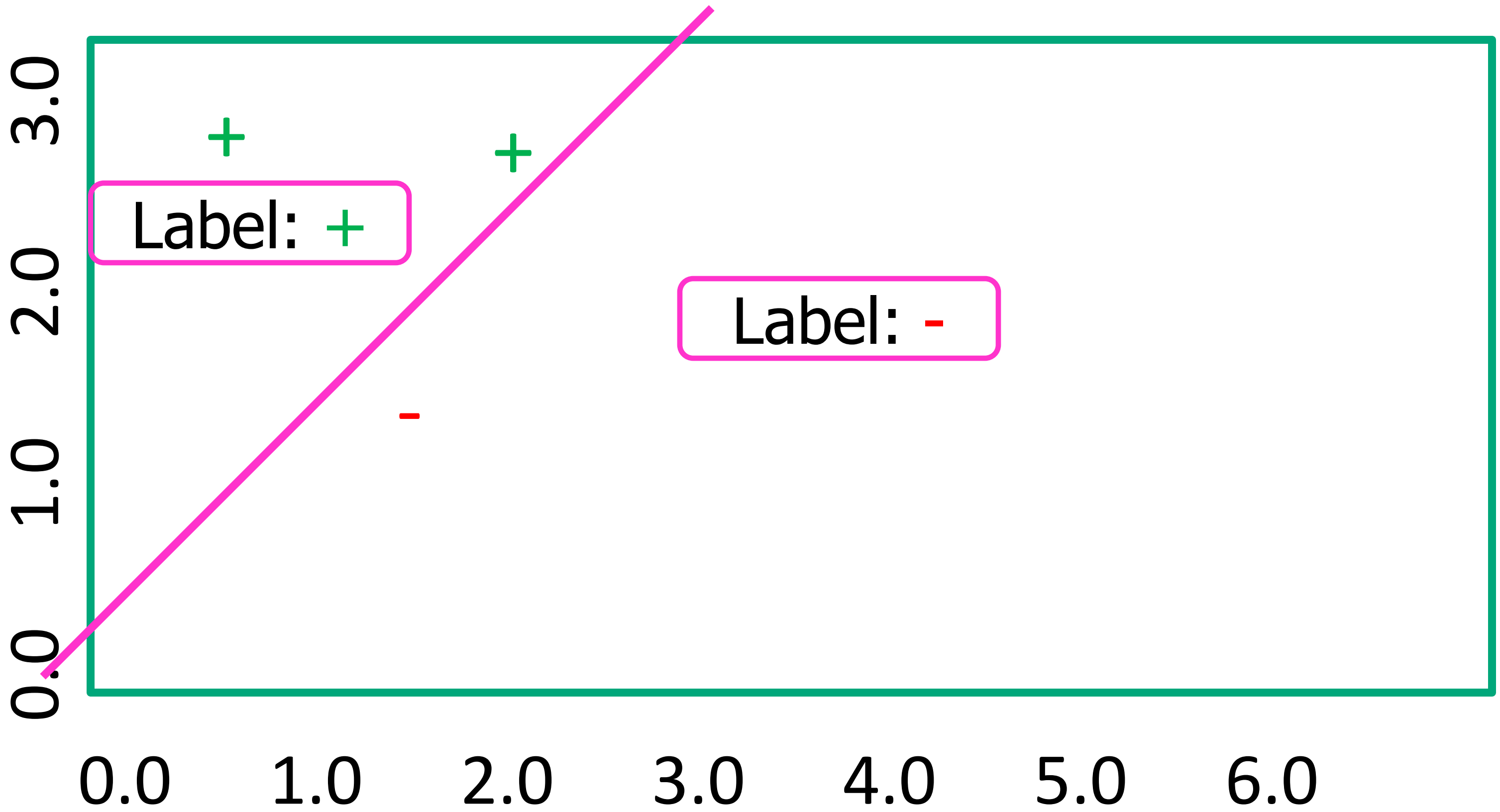
Online



Online



Online



Take a 15 minute break

Instance Based Learning

Simple Idea: Memorization

- Employed by first learning systems
- Memorize training data and look for exact match when presented with a new example
- If a new example does not match what we have seen before, it makes no decision

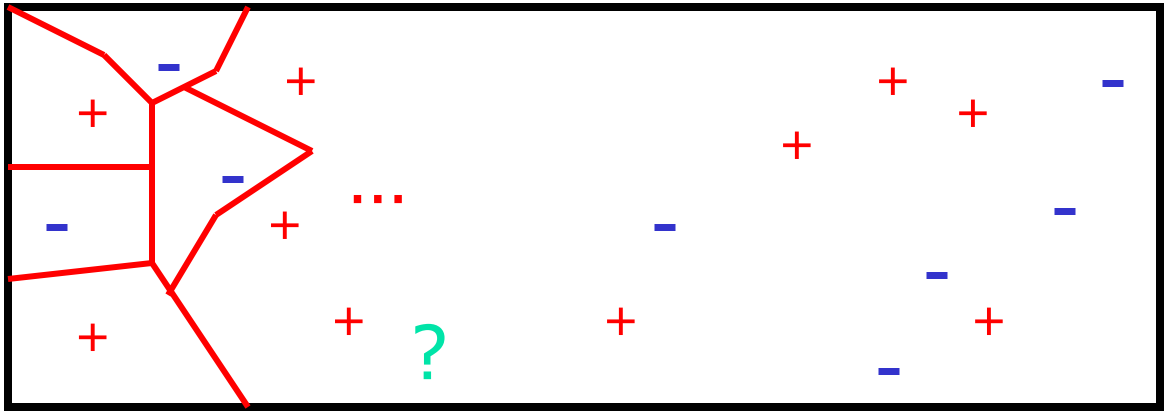
- *Need computer to generalize from experience*

Nearest-Neighbor Algorithms

- Learning ≈ memorize training examples
- Classification: Find most similar example and output its category
- Regression: Find most similar example and output its value

Venn

"Voronoi Diagrams" (pg 233)



Example

Training Set

1. a=0, b=0, c=1 +
2. a=0, b=0, c=0 -
3. a=1, b=1, c=1 -

Test Example

- a=0, b=1, c=0 ?

"Hamming Distance"

- Ex 1 = 2
- Ex 2 = 1 ← So output -
- Ex 3 = 2

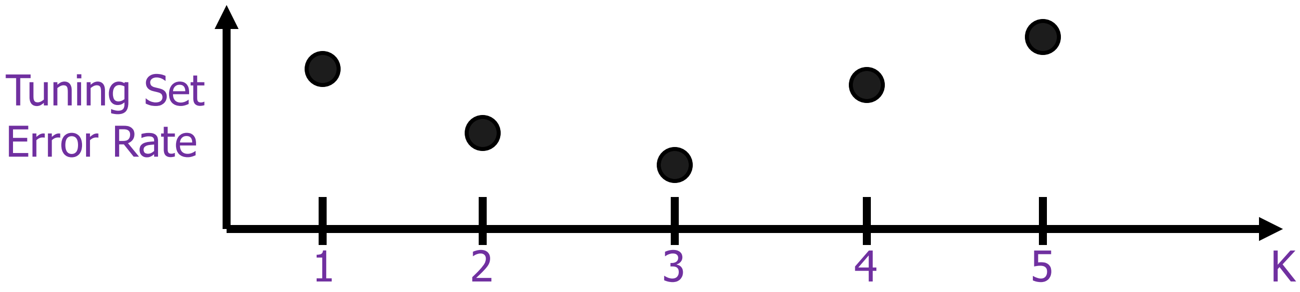
Sample Experimental Results
(see UCI archive for more)

Testbed	Testset Correctness		
	1-NN	D-Trees	Neural Nets
Wisconsin Cancer	98%	95%	96%
Heart Disease	78%	76%	?
Tumor	37%	38%	?
Appendicitis	83%	85%	86%

Simple algorithm works quite well!

K-NN Algorithm

- Learning ≈ memorize training examples
- For example unseen test example e, collect *K* nearest examples to e
  - Combine the classes to label e's
- Question: How do we pick *K*?
  - Highly problem dependent
  - Use tuning set to select its value



Distance Functions:

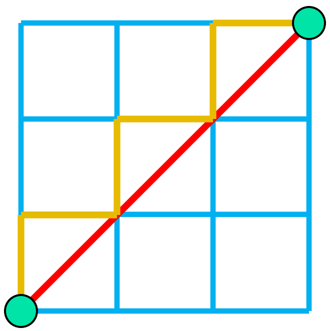
- Hamming: Measures overlap/differences between examples
- Value difference metric: Attribute values are close if they make similar predictions

$$\delta(val_i, val_j) = \sum_{h=1}^{\#\text{classes}} |P(c_h|val_i) - P(c_h|val_j)|^n$$

1. a=0, b=0, c=1 +
2. a=0, b=2, c=0 -
3. a=1, b=3, c=1 -
4. a=1, b=1, c=0 +

Distance functions

- Euclidean



- Manhattan
- L^n norm

$$L^n(\mathbf{x}_1, \mathbf{x}_2) = \sqrt[n]{\sum_{i=1}^{\#\text{dim}} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^n}$$

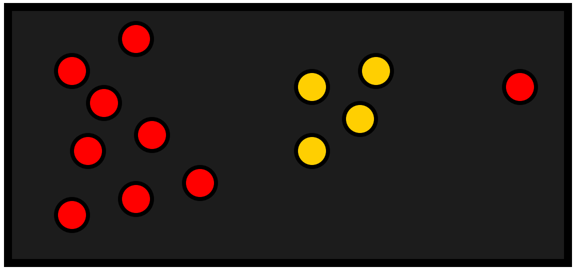
- Note: Often want to normalize these values

In general, distance function is problem specific

Variations on a Theme

(From Aha, Kibler and Albert in ML Journal)

- IB1 – keep all examples

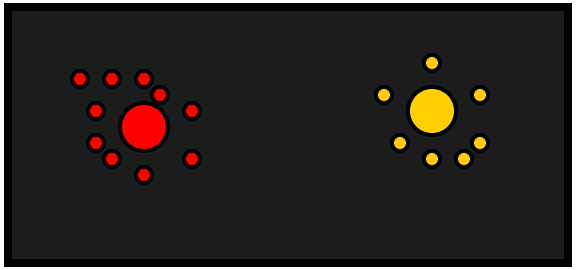


- IB2 – keep next instance if incorrectly classified by using previous instances
  - Uses less storage (good)
  - Order dependent (bad)
  - Sensitive to noisy data (bad)

Variations on a Theme (cont.)

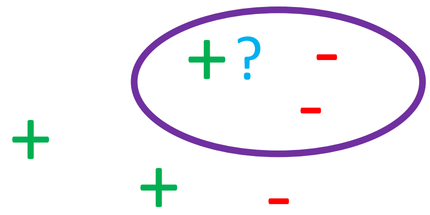
- IB3 – extend IB2 to more intelligently decide which examples to keep (see article)
  - Better handling of noisy data

- Another Idea - cluster groups, keep example from each (median/centroid)
  - Less storage, faster lookup



Distance Weighted *K*-NN

- Consider the following example for 3-NN



- The unseen example is much closer to the positive example, but labeled as a negative

- Idea: Weight nearer examples more heavily

Distance Weighted *K*-NN

- Classification function is:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

Where

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

- Notice that now we should use all training examples instead of just k

Advantages of K-NN

- Training is very fast
- Learn complex target function easily
- No loss of information from training data
- Easy to implement
- Good baseline for empirical evaluation
- Possible to do incremental learning
- Plausible model for human memory

Disadvantages of K-NN

- Slow at query time
- Memory intensive
- Easily fooled by irrelevant attributes
- Picking the distance function can be tricky
- No insight into the domain as there is no explicit model
- Doesn’t exploit, notice structure in examples

Reducing the Computation Cost

- Use clever data structures
  - E.g., k-D trees (for low dimensional spaces)
- Efficient similarity computation
  - Use a cheap, approximate metric to weed out examples
  - Use expensive metric on remaining examples
- Use a subset of the features

Reducing the Computation Cost

- Form prototypes
- Use a subset of the training examples
  - Remove those that don’t effect the frontier
  - Edited k-NN

Edited k-Nearest Neighbor

EDITED_k-NN(S)
    S: Set of instances
For each instance $\mathbf{x}$ in S
    If $\mathbf{x}$ is correctly classified by $S - \{\mathbf{x}\}$
        Remove $\mathbf{x}$ from S
Return S

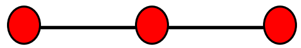
EDITED_k-NN(S)
    S: Set of instances
$T = \emptyset$
For each instance $\mathbf{x}$ in S
    If $\mathbf{x}$ is **not** correctly classified by T
        Add $\mathbf{x}$ to T
Return T

Curse of Dimensionality

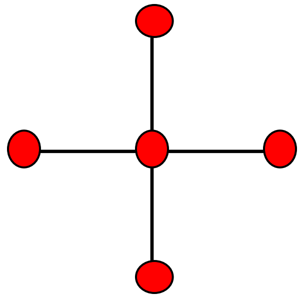
- Imagine instances are described by 20 attributes, but only two are relevant to the concept
- Curse of dimensionality
  - With lots of features, can end up with spurious correlations
  - Nearest neighbors are easily mislead with high-dim X
  - Easy problems in low-dim are hard in high-dim
  - Low-dim intuition doesn't apply in high-dim

Example: Points on Hypergrid

- In 1-D space: 2 NN are equidistant



- In 2-D space: 4 NN are equidistant



Feature Selection

Filtering-Based Feature Selection

all features

↓

FS algorithm

↓

subset of features

↓

ML algorithm

↓

model

Wrapper-Based Feature Selection

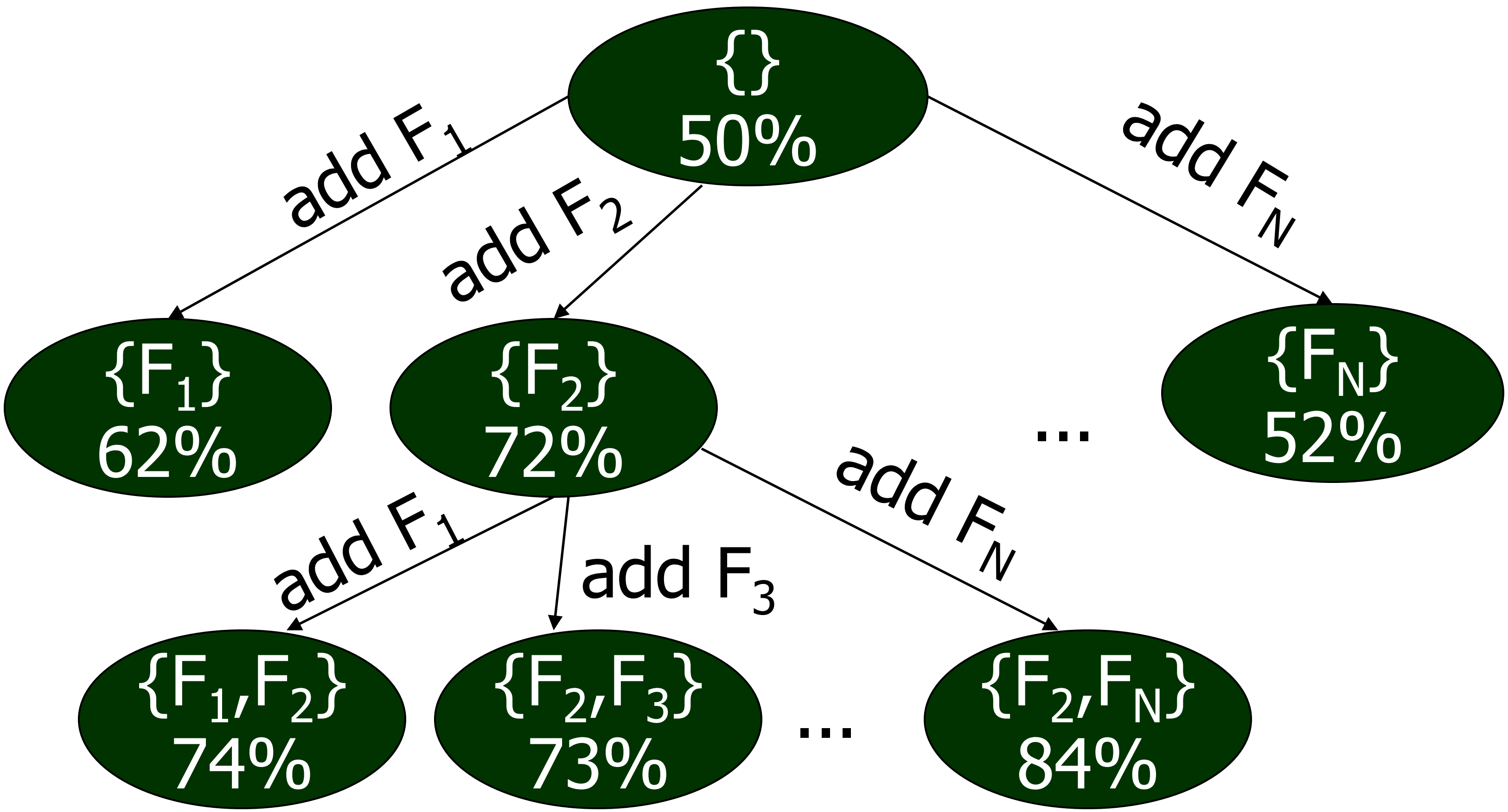
all features → FS algorithm calls ML algorithm <u>many</u> times, uses it to help select features | ML algorithm → model

Feature Selection as Search Problem

- State = set of features
  - Start state = *empty* (forward selection)
    or *full* (backward selection)
  - Goal test = highest scoring state
- Operators
  - add/subtract features
- Scoring function
  - accuracy on training (or tuning) set of ML algorithm using this state's feature set

Forward Feature Selection

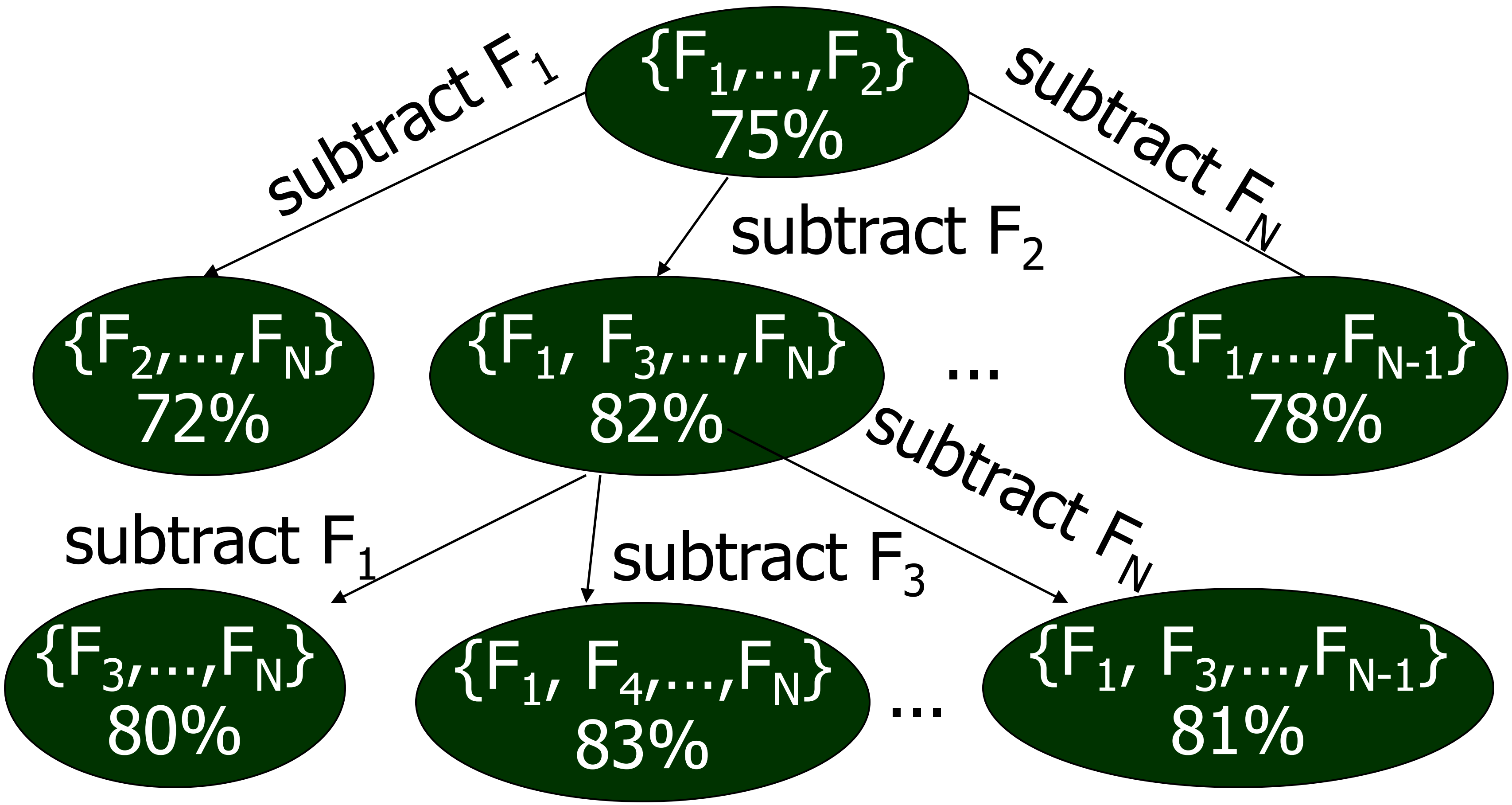
Greedy search (aka "Hill Climbing")



```
FORWARD_SELECTION(FS)
    FS: Set of features used to describe examples
Let SS = ∅
Let BestEval = 0
Repeat
    Let BestF = None
    For each feature F in FS and not in SS
        Let SS' = SS ∪ {F}
        If Eval(SS') > BestEval
            Then Let BestF = F
                 Let BestEval = Eval(SS')
    If BestF ≠ None
        Then Let SS = SS ∪ {BestF}
Until BestF = None or SS = FS
Return SS
```

Backward Feature Selection

Greedy search (aka "Hill Climbing")



```
BACKWARD_ELIMINATION(FS)
    FS: Set of features used to describe examples
Let SS = FS
Let BestEval = Eval(SS)
Repeat
    Let WorstF = None.
    For each feature F in SS
        Let SS' = SS − {F}
        If Eval(SS') ≥ BestEval
            Then Let WorstF = F
                    Let BestEval = Eval(SS')
    If WorstF ≠ None
        Then Let SS = SS − {WorstF}
Until WorstF = None or SS = ∅
Return SS
```

Forward vs. Backward Feature Selection

Forward

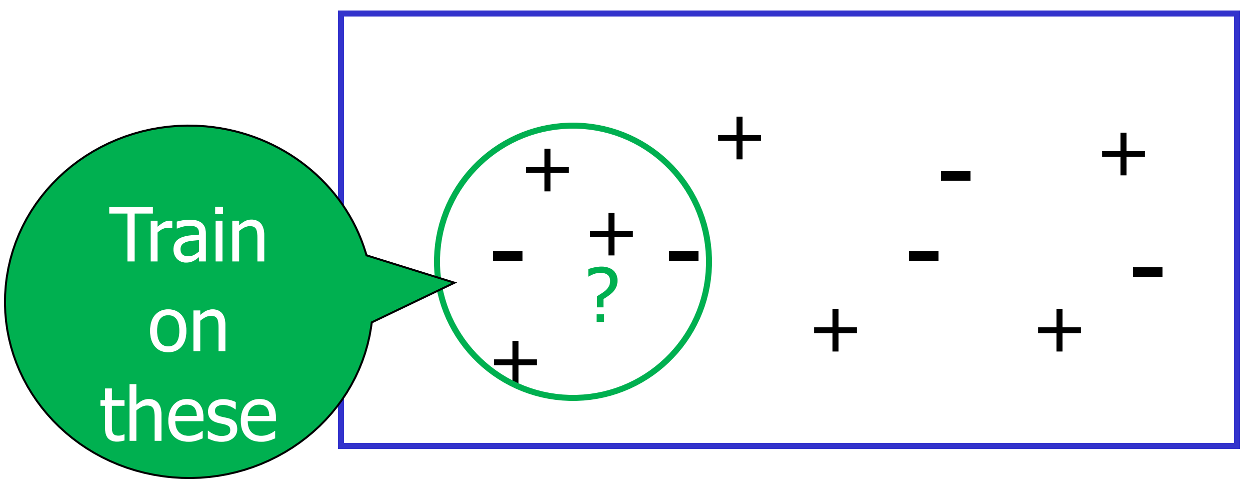
- Faster in early steps because fewer features to test
- Fast for choosing a small subset of the features
- Misses features whose usefulness requires other features (feature synergy)

Backward

- Fast for choosing all but a small subset of the features
- Preserves features whose usefulness requires other features
  - Example: area important, features = length, width

Local Learning

- Collect *k* nearest neighbors
- Give them to some supervised ML algo
- Apply learned model to test example



Locally Weighted Regression

- Form an explicit approximation for each query point seen
- Fit learn linear, quadratic, etc., function to the k nearest neighbors
- Provides a piecewise approximation to f

Several choices of error to minimize:

- Squared error over k nearest neighbors

$$E_1(x_q) \equiv \sum_{x \in\ kNN(x_q)} (f(x) - \hat{f}(x))^2$$

- Distance-weighted squared error over all neighbors

$$E_2(x_q) \equiv \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

- ...

Homework 1: Programming Component

- Implement collaborative filtering algorithm
- Apply to (subset of) Netflix Prize data
  - 1821 movies, 28,978 users, 3.25 million ratings (* - *****)
- Try to improve predictions
- Optional: Add your ratings & get recommendations
- Paper: Breese, Heckerman & Cadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering" (UAI-98)

Collaborative Filtering

- Problem: Predict whether someone will like a Web page, movie, book, CD, etc.
- Previous approaches: Look at content
- Collaborative filtering
  - Look at what similar users liked
  - Intuition is that similar users will have similar likes and dislikes

Collaborative Filtering

- Represent each user by vector of ratings
- Two types:
  - Yes/No
  - Explicit ratings (e.g., $0 - *****$)
- Predict rating:

$$\hat{R}_{ik} = \overline{R}_i + \alpha \sum_{X_j \in \mathbf{N}_i} W_{ij}(R_{jk} - \overline{R}_j)$$

- Similarity (Pearson coefficient):

$$W_{ij} = \frac{\sum_k (R_{ik} - \overline{R}_i)(R_{jk} - \overline{R}_j)}{\left[\sum_k (R_{ik} - \overline{R}_i)^2 \sum_k (R_{jk} - \overline{R}_j)^2\right]^{0.5}}$$

Fine Points

- Primitive version:

$$\hat{R}_{ik} = \alpha \sum_{X_j \in \mathbf{N}_i} W_{ij} R_{jk}$$

- $\alpha = (\sum |W_{ij}|)^{-1}$
- $\mathbf{N}_i$ can be whole database, or only k nearest neighbors
- R_{jk} = Rating of user j on item k
- $\overline{R}_j$ = Average of all of user j's ratings
- Summation in Pearson coefficient is over all items rated by *both* users
- In principle, any prediction method can be used for collaborative filtering

Example

	R1	R2	R3	R4	R5	R6
Alice	2	-	4	4	-	2
Bob	1	5	4	-	-	2
Chris	4	3	-	-	-	5
Diana	3	-	2	4	-	5

Compare Alice and Bob

Example

	R1	R2	R3	R4	R5	R6
Alice	2	-	3	2	-	1
Bob	1	5	4	-	-	2
Chris	4	3	-	-	-	5
Diana	3	-	2	4	-	5

$\overline{\text{Alice}} = 2$

$\overline{\text{Bob}} = 3$

$W = [0 + (1)(1) + (-1)(-1)] / \ldots = 2 / 12^{0.5}$

$\text{Alice}_{R2} = 2 + 1/w * [w *(5-3)] = 4$

Summary

- Brief introduction to data mining
- Overview of inductive learning
  - Problem definition
  - Key terminology
- Instance-based learning: k-NN
- Homework 1: Collaborative filtering

Next Class

- Decision Trees
  - Read Mitchell chapter 3
- Empirical methodology
  - Provost, Fawcett and Kohavi, “The Case Against Accuracy Estimation”
  - Davis and Goadrich, “The Relationship Between Precision-Recall and ROC Curves”
- Homework 1 overview

Questions?