Computational Learning Theory: The Theory of Generalization

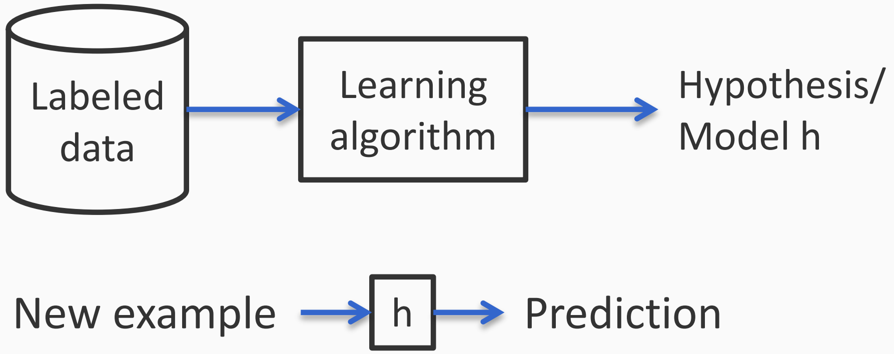
Machine Learning
Spring 2018

THE UNIVERSITY OF UTAH

The slides are mainly from Vivek Srikumar

Checkpoint: The bigger picture

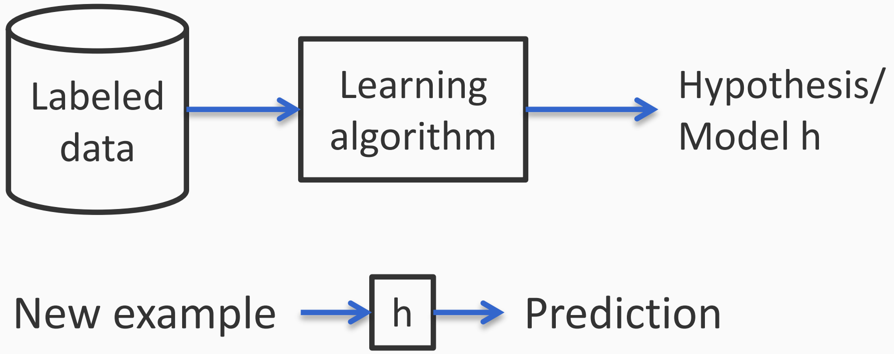
- Supervised learning: instances, labels, and hypotheses



Checkpoint: The bigger picture

- Supervised learning: instances, labels, and hypotheses

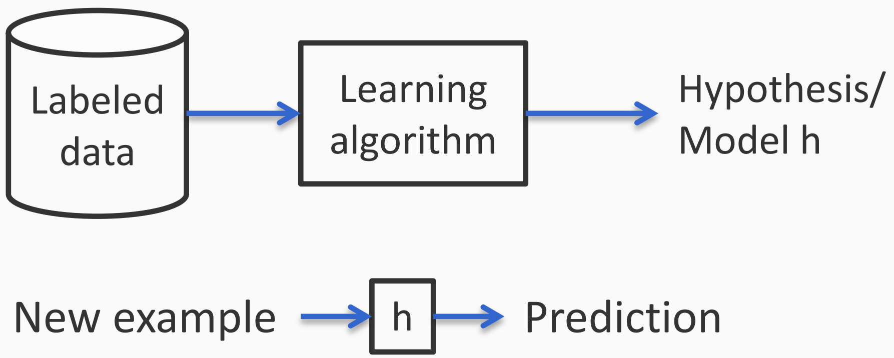
- Specific learners
  - Decision trees
  - Perceptron
  - LMS



Checkpoint: The bigger picture

- Supervised learning: instances, labels, and hypotheses

- Specific learners
  - Decision trees
  - Perceptron
  - LMS

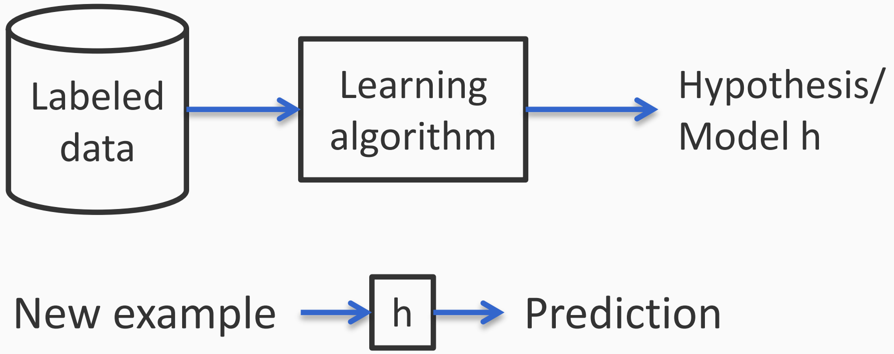


- General ML ideas
  - Features as high dimensional vectors
  - Overfitting
  - Mistake-bound

Checkpoint: The bigger picture

- Supervised learning: instances, labels, and hypotheses

- Specific learners
  - Decision trees
  - Perceptron
  - LMS



- General ML ideas
  - Features as high dimensional vectors
  - Overfitting
  - Mistake-bound

Questions?

Next Topic: Computational Learning Theory

- The Theory of Generalization
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

This lecture: Computational Learning Theory

- The Theory of Generalization
  - When can we trust the learning algorithm?
  - What functions can be learned?
  - Batch learning
- Probably Approximately Correct (PAC) learning
- Positive and negative learnability results
- Agnostic Learning
- Shattering and the VC dimension

Computational Learning Theory

Are there general "laws of nature" related to learnability?

We want theory that can relate

- Probability of successful Learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples are presented

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels (f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels (f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels (f(x))
  - <($\boxed{1}$,1,1,1,1,1,…,1,1), $\boxed{1}$>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <($\boxed{1}$,1,1,1,1,0,...0,1,1), $\boxed{1}$>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <($\boxed{1}$,1,1,1,1,0,...0,0,1), $\boxed{1}$>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <($\boxed{1}$,1,1,1,1,1,…,0,1), $\boxed{1}$>
  - <(0,1,0,1,0,0,...0,1,1), 0>

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Whenever the output is 1, x_1 is present

$$f = x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Learning Conjunctions

Some random source (nature) provides training examples

- Teacher (Nature) provides the labels (f(x))
  - <(1,1,1,1,1,1,…,1,1), 1>
  - <(1,1,1,0,0,0,…,0,0), 0>
  - <(1,1,1,1,1,0,...0,1,1), 1>
  - <(1,0,1,1,1,0,...0,1,1), 0>
  - <(1,1,1,1,1,0,...0,0,1), 1>
  - <(1,0,1,0,0,0,...0,1,1), 0>
  - <(1,1,1,1,1,1,…,0,1), 1>
  - <(0,1,0,1,0,0,...0,1,1), 0>

For a reasonable learning algorithm (by *elimination*), the final hypothesis will be

$$h = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_{100}$$

Whenever the output is 1, x_1 is present

With the given data, we only learned an *approximation* to the true concept.
Is it good enough?

Two Frameworks for How good is our learning algorithm?

- *Mistake Driven* Learning algorithms
  - Update your hypothesis only when you make mistakes
  - Define *good* in terms of how many mistakes you make before you stop
  - ***Online learning***

- Analyze the probabilistic intuition
  - Never saw a feature in positive examples, maybe we'll never see it
  - And if we do, it will be with small probability, so the concepts we learn may be *pretty good*
    - *Pretty good:* In terms of performance on future data
  - ***PAC framework***

The mistake-bound approach

- The mistake bound model is a theoretical approach
  - We can determine the number of mistakes the learning algorithm can make before converging

- But no answer to "*How many examples do you need before converging to a good hypothesis?*"

- Because the mistake-bound model makes no assumptions about the order or distribution of training examples
  - Both a strength and a weakness of the mistake bound model

PAC learning

- A model for *batch learning*
  - Train on a fixed training set
  - Then deploy it in the wild

- How well will your learned hypothesis do on *future* instances?

The setup

- Instance Space: X, the set of examples

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: $f \in C$ is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: f ∈ C is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …
- Hypothesis Space: H, the set of possible hypotheses
  - This is the set that the learning algorithm explores

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: f ∈ C is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …
- Hypothesis Space: H, the set of possible hypotheses
  - This is the set that the learning algorithm explores
- Training instances: S×{-1,1}: positive and negative examples of the target concept. (S is a finite subset of X)

$$< x_1, f(x_1) >, < x_2, f(x_2) >, \ldots < x_n, f(x_n) >$$

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: f ∈ C is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …
- Hypothesis Space: H, the set of possible hypotheses
  - This is the set that the learning algorithm explores
- Training instances: S×{-1,1}: positive and negative examples of the target concept. (S is a finite subset of X)

$$< x_1, f(x_1) >, < x_2, f(x_2) >, \ldots < x_n, f(x_n) >$$

- What we want: A hypothesis h ∈ H such that h(x) = f(x)
  - A hypothesis h ∈ H such that h(x) = f(x) for all x ∈ S ?
  - A hypothesis h ∈ H such that h(x) = f(x) for all x ∈ X ?

The setup

- Instance Space: X, the set of examples
- Concept Space: C, the set of possible target functions: $f \in C$ is the hidden target function
  - Eg: all n-conjunctions; all n-dimensional linear functions, …
- Hypothesis Space: H, the set of possible hypotheses
  - This is the set that the learning algorithm explores
- Training instances: $S \times \{-1,1\}$: positive and negative examples of the target concept. (S is a finite subset of X)
  - *Training instances are generated by a fixed unknown probability distribution D over X*
- What we want: A hypothesis $h \in H$ such that $h(x) = f(x)$
  - Evaluate h on subsequent examples $x \in X$ drawn according to D

PAC Learning – Intuition

- The assumption of fixed distribution is important for two reasons:
  1. Gives us hope that what we learn on the training data will be meaningful on future examples
  2. Also gives a well-defined notion of the error of a hypothesis according to the target function

PAC Learning – Intuition

- The assumption of fixed distribution is important for two reasons:
  1. Gives us hope that what we learn on the training data will be meaningful on future examples
  2. Also gives a well-defined notion of the error of a hypothesis according to the target function

- "*The future will be like the past*": We have seen many examples (drawn according to the distribution D)
  – Since in all the positive examples x_1 was active (1), it is very likely that it will be active in future positive examples
  – If not, in any case, x_1 is active only in a small percentage of the examples so our error will be small

Generalization Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

$$err_D(h) = Pr_{x \sim D}[h(x) \neq f(x)]$$

Generalization Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

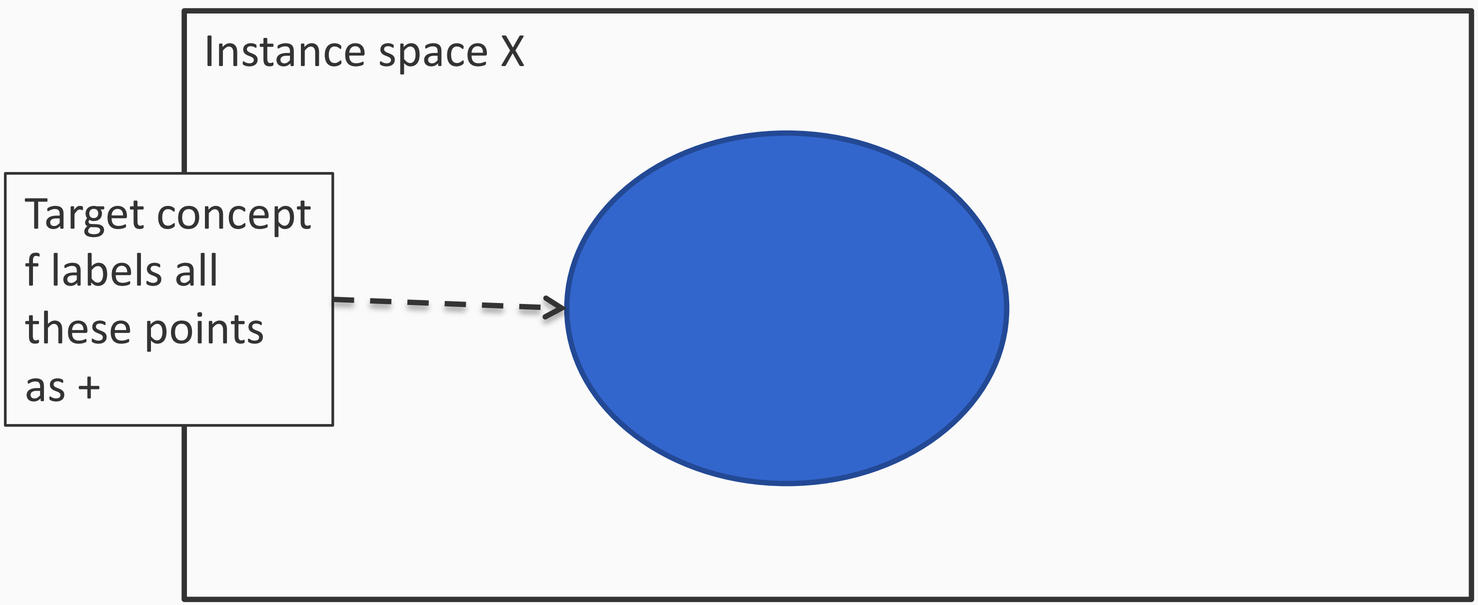
$$err_D(h) = Pr_{x \sim D}[h(x) \neq f(x)]$$

Generalization Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

$$\mathrm{err}_D(h) = \Pr_{x \sim D}[h(x) \neq f(x)]$$

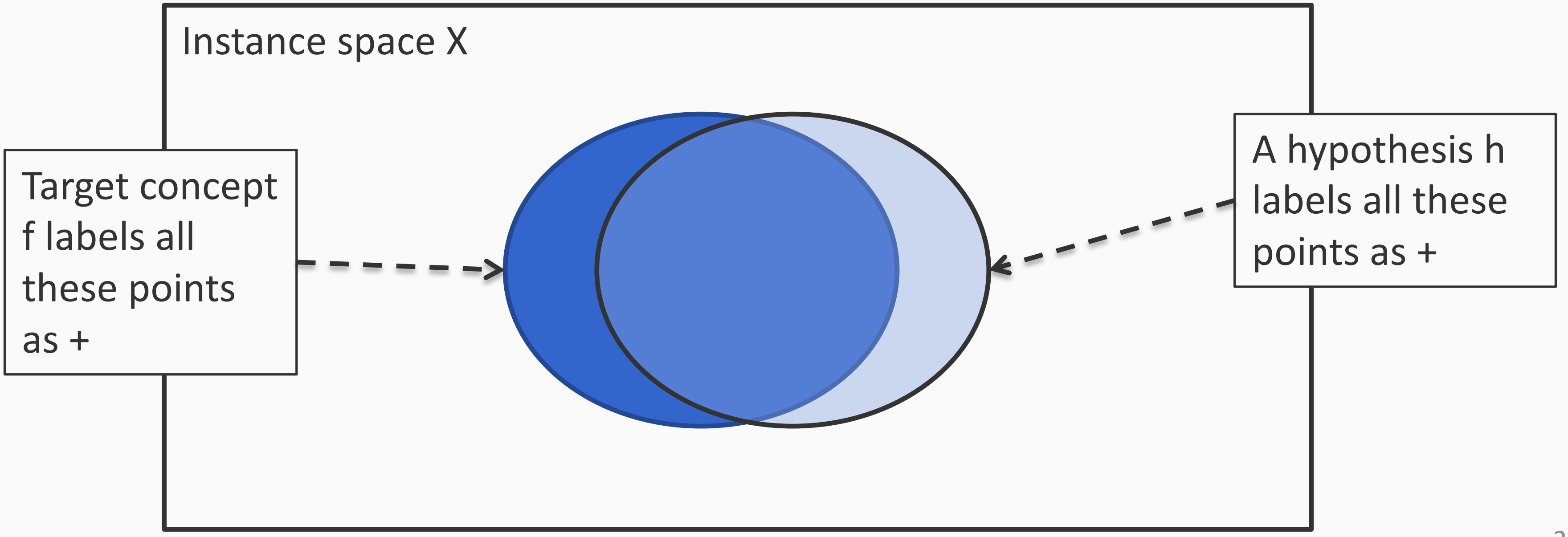


Generalization Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

$$err_D(h) = Pr_{x \sim D}[h(x) \neq f(x)]$$

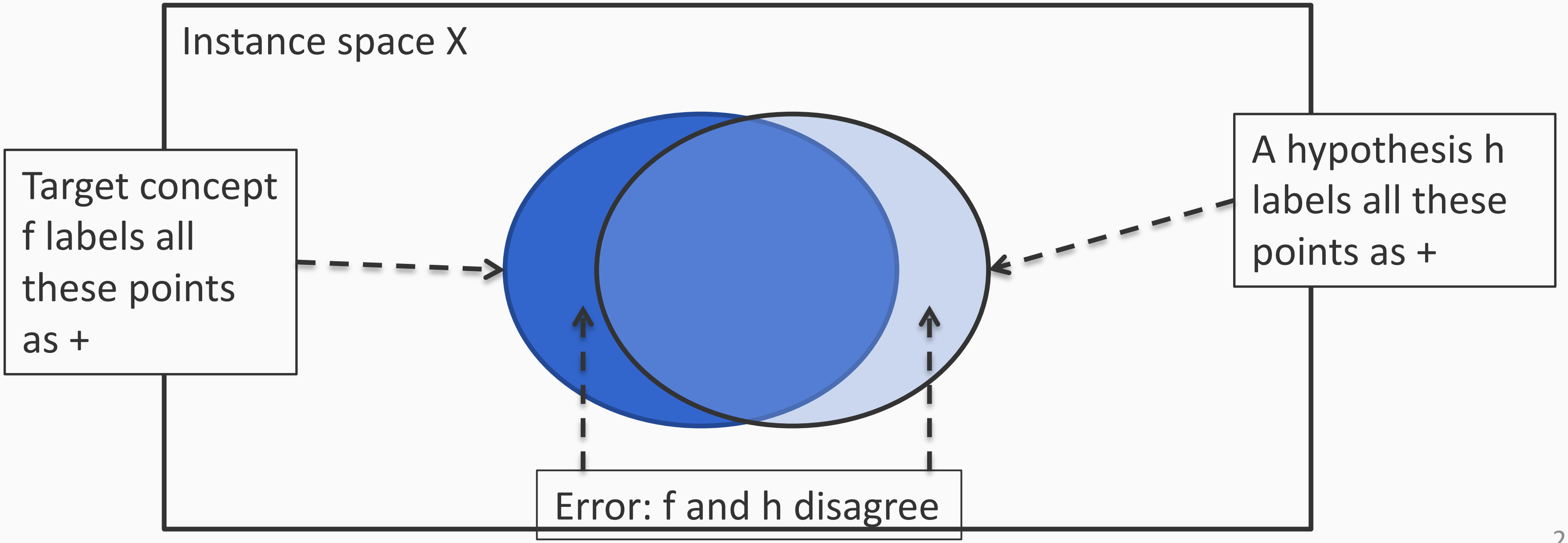


Generalization Error of a hypothesis

Definition

Given a distribution D over examples, the *error* of a hypothesis h with respect to a target concept f is

$$err_D(h) = Pr_{x \sim D}[h(x) \neq f(x)]$$



Empirical error

Contrast true error against the *empirical error*

For a target concept f, the empirical error of a hypothesis h is defined for a training set S as the fraction of examples x in S for which the functions f and h disagree. That is, $h(x) \neq f(x)$

Denoted by $err_S(h)$

Overfitting: When the empirical error on the training set $err_S(h)$ is substantially lower than $err_D(h)$

Mistake driven learning vs. Batch learning

Mistake driven learning

Batch learning

Mistake driven learning vs. Batch learning

Mistake driven learning

- No assumptions about the distribution of examples

Batch learning

- Examples are drawn from a fixed (probably unknown) probability distribution D over the instance space

Mistake driven learning vs. Batch learning

Mistake driven learning

- No assumptions about the distribution of examples
- Learning is a sequence of trials
  - Learner sees a single example, makes a prediction
  - If mistake, update hypothesis

Batch learning

- Examples are drawn from a fixed (probably unknown) probability distribution D over the instance space
- Learning uses a training set S, drawn i.i.d from the distribution D

Mistake driven learning vs. Batch learning

Mistake driven learning

- No assumptions about the distribution of examples
- Learning is a sequence of trials
  - Learner sees a single example, makes a prediction
  - If mistake, update hypothesis
- Goal: To bound the total number of mistakes over time

Batch learning

- Examples are drawn from a fixed (probably unknown) probability distribution D over the instance space
- Learning uses a training set S, drawn i.i.d from the distribution D
- Goal: To find a hypothesis that has low chance of making a mistake on a new example from D