Introduction to the Theory of Machine Learning

The Language of Learning Theory

Rachit Nimavat

IIIT Surat

June 27, 2025

Outline

Machine learning can be used to...

Recognize speech, images, stock market patterns

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

Design course,

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

Design course, generate course slides, produce video of an instructor teaching those slides, generate assignments,

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

Design course, generate course slides, produce video of an instructor teaching those slides, generate assignments, solve assignments,

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

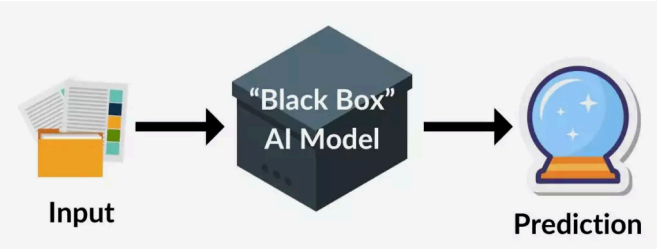
Design course, generate course slides, produce video of an instructor teaching those slides, generate assignments, solve assignments, grade assignments

Machine learning can be used to...

Recognize speech, images, stock market patterns

Classify documents, predict protein sequences

Design course, generate course slides, produce video of an instructor teaching those slides, generate assignments, solve assignments, grade assignments



Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

What kind of **guarantees** we can achieve with that data

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

What kind of **guarantees** we can achieve with that data

Optimize Learning!

How **efficient** our learning process can be

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

What kind of **guarantees** we can achieve with that data

Optimize Learning!

How **efficient** our learning process can be

Relating data size, processing time, space, accuracy

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

What kind of **guarantees** we can achieve with that data

Optimize Learning!

How **efficient** our learning process can be

Relating data size, processing time, space, accuracy

Improve Learning!

How does **biases** in data affect learning

Goals of ML Theory

Understand Learning!

What kinds of **tasks** can we *hope* to learn

What kind of **data** is needed to learn that task

What kind of **guarantees** we can achieve with that data

Optimize Learning!

How **efficient** our learning process can be

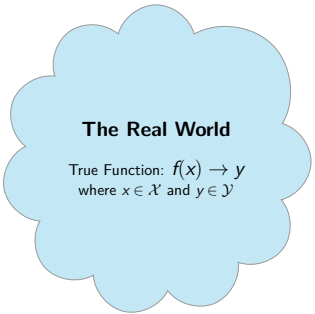
Relating data size, processing time, space, accuracy

Improve Learning!

How does **biases** in data affect learning

Preserve **Privacy**? **Unlabeled** data? **Interactive** environments? ...

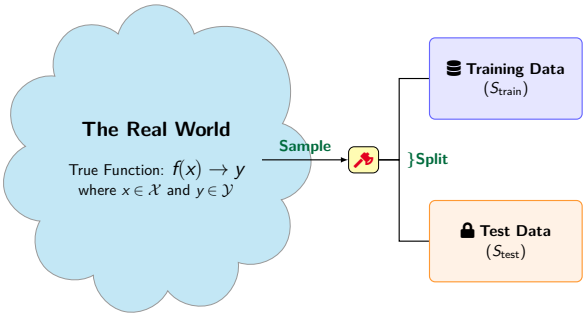
Understanding Learning



Example:

- $\mathcal{X}$ is the set of all images
- $\mathcal{Y} = \{\text{CAT}, \text{DOG}\}$ is the set of all labels
- $x \in \mathcal{X}$ is a *single* image
- $y = f(x)$ is its true label

Understanding Learning

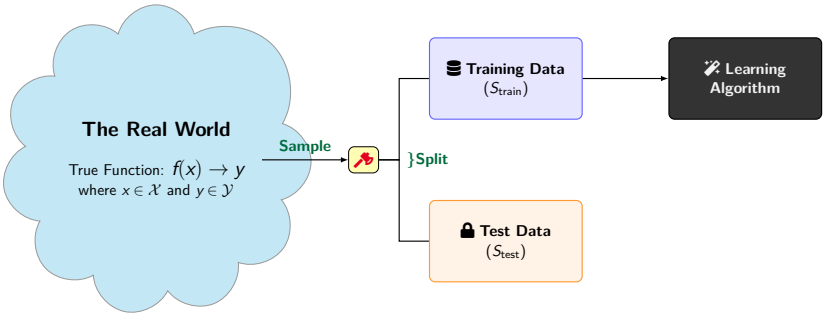


Example:

- $\mathcal{X}$ is the set of all images
- $\mathcal{Y} = \{\text{CAT}, \text{DOG}\}$ is the set of all labels
- $x \in \mathcal{X}$ is a *single* image
- $y = f(x)$ is its true label

Split sampled data into training and test sets

Understanding Learning

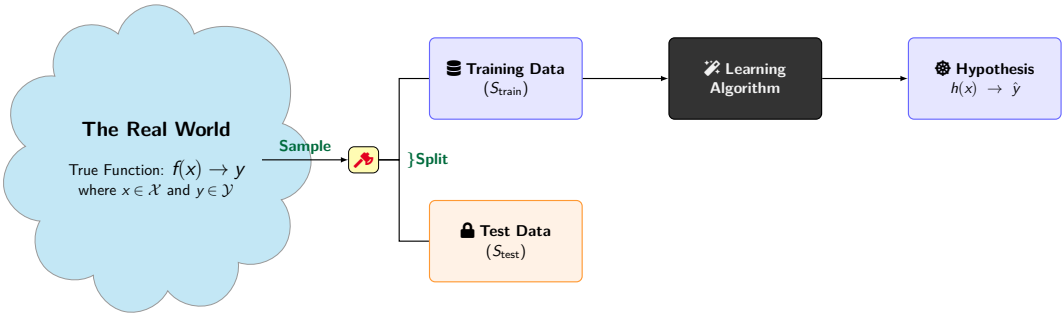


Example:

- $\mathcal{X}$ is the set of all images
- $\mathcal{Y} = \{\text{CAT}, \text{DOG}\}$ is the set of all labels
- $x \in \mathcal{X}$ is a *single* image
- $y = f(x)$ is its true label

The learning phase

Understanding Learning



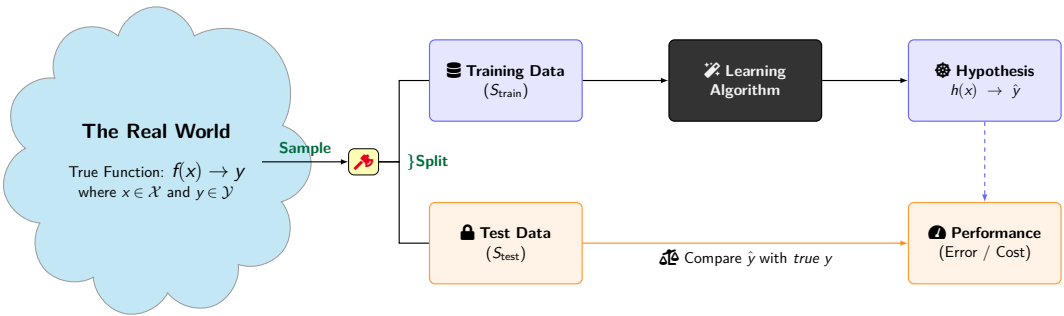
Example:

- $\mathcal{X}$ is the set of all images
- $\mathcal{Y} = \{\text{CAT}, \text{DOG}\}$ is the set of all labels
- $x \in \mathcal{X}$ is a *single* image
- $y = f(x)$ is its true label

Output of model our *guess* h for *true* f

$\hat{y} = h(x)$ is the models output label

Understanding Learning



Example:

- $\mathcal{X}$ is the set of all images
- $\mathcal{Y} = \{\text{CAT}, \text{DOG}\}$ is the set of all labels
- $x \in \mathcal{X}$ is a *single* image
- $y = f(x)$ is its true label

Evaluating performance

$\hat{y} = h(x)$ is the models output label

Cost: $\mathbf{1}[y \neq \hat{y}]$

Understanding Learning

Physicists	**Mathematicians**
Start with: Observations / Data	Start with: Axioms / Laws
	

Understanding Learning

Physicists	**Mathematicians**
Start with: Observations / Data	Start with: Axioms / Laws
	
Goal: Find laws that correctly explain the observations/data	Goal: Find theorems/observations that follow from the axioms

Understanding Learning

Physicists	**Mathematicians**
Start with: Observations / Data	Start with: Axioms / Laws
	
Goal: Find laws that correctly explain the observations/data	Goal: Find theorems/observations that follow from the axioms

Where Machine Learning Fits?

Find rules that make *good predictions* about unseen data.

Finding Good Rules that *Generalize* Well

The Core Question of Generalization

- Model trained on **training data**

Finding Good Rules that *Generalize* Well

The Core Question of Generalization

- Model trained on **training data**
- Performs well on **test data**

Finding Good Rules that *Generalize* Well

The Core Question of Generalization

- Model trained on **training data**
- Performs well on **test data**
- Confidence about performance on **new, unseen data**?

Finding Good Rules that *Generalize* Well

The Core Question of Generalization

- Model trained on **training data**
- Performs well on **test data**
- Confidence about performance on **new, unseen data**?

The Fundamental Assumption of ML

All data points are drawn independently from a fixed, but unknown, probability distribution $\mathcal{D}$.

The Language of Learning Theory : Data

$\mathcal{X}$: The set of all data points

$\mathcal{Y}$: The set of all labels

$\mathcal{D}$: The underlying, but unknown joint probability distribution over $\mathcal{X} \times \mathcal{Y}$.

The Language of Learning Theory : Data

$\mathcal{X}$: The set of all data points

$\mathcal{Y}$: The set of all labels

$\mathcal{D}$: The underlying, but unknown joint probability distribution over $\mathcal{X} \times \mathcal{Y}$.

Single Labeled Example

A *data point* x with (true) *label* y is drawn from $\mathcal{D}$

$$(x, y) \sim \mathcal{D}$$

The Language of Learning Theory : Data

$\mathcal{X}$: The set of all data points

$\mathcal{Y}$: The set of all labels

$\mathcal{D}$: The underlying, but unknown joint probability distribution over $\mathcal{X} \times \mathcal{Y}$.

Single Labeled Example

A *data point* x with (true) *label* y is drawn from $\mathcal{D}$

$$(x, y) \sim \mathcal{D}$$

Training Set

m labeled examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ are drawn independently and identically from $\mathcal{D}$

$$S \sim \mathcal{D}^m$$

The Language of Learning Theory : Data

$\mathcal{X}$: The set of all data points

$\mathcal{Y}$: The set of all labels

$\mathcal{D}$: The underlying, but unknown joint probability distribution over $\mathcal{X} \times \mathcal{Y}$.

Single Labeled Example

A *data point* x with (true) *label* y is drawn from $\mathcal{D}$

$$(x, y) \sim \mathcal{D}$$

Training Set

m labeled examples $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ are drawn independently and identically from $\mathcal{D}$

$$S \sim \mathcal{D}^m$$

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

"The training set S drew m **i.i.d** examples from a single, unknown, underlying distribution $\mathcal{D}$."

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Learning Algorithm

$\mathcal{A}$ does some computation over S to produce a **hypothesis** (prediction rule) h.

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Learning Algorithm

$\mathcal{A}$ does some computation over S to produce a **hypothesis** (prediction rule) h.

Under the hood, $\mathcal{A}$ searches within a **Hypothesis Class** (pre-defined 'universe' of possible hypotheses) $\mathcal{H}$.

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Learning Algorithm

$\mathcal{A}$ does some computation over S to produce a **hypothesis** (prediction rule) h.

Under the hood, $\mathcal{A}$ searches within a **Hypothesis Class** (pre-defined 'universe' of possible hypotheses) $\mathcal{H}$.

$\mathcal{H}$ is the "library" of models
our algorithm can choose from:
linear separators,
width-100 depth-10 neural networks,
100-line python programs...

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Learning Algorithm

$\mathcal{A}$ does some computation over S to produce a **hypothesis** (prediction rule) h.

Under the hood, $\mathcal{A}$ searches within a **Hypothesis Class** (pre-defined 'universe' of possible hypotheses) $\mathcal{H}$.

Some ways in which $\mathcal{A}$ may work:

SVM: Out of all linear separators, pick one with maximum margin

Gradient Descent: Random initialization of weights and back-propagate gradients until we hit a local minima

LLM: Ask ChatGPT to generate a 100 line python program

The Language of Learning Theory : Learning Algorithm

Training Set

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

The Learning Algorithm

$\mathcal{A}$ does some computation over S to produce a **hypothesis** (prediction rule) h.

Under the hood, $\mathcal{A}$ searches within a **Hypothesis Class** (pre-defined 'universe' of possible hypotheses) $\mathcal{H}$.

Some ways in which $\mathcal{A}$ may
SVM: Out of all linear se
Gradient Descent: Ranc adients
until we hit a local minim
LLM: Ask ChatGPT to generate a 100 line python program

A typical approach:
Fix hypothesis class $\mathcal{H}$
collect training data S and
choose a $h \in \mathcal{H}$ that minimizes error on S.
Empirical Risk Minimization (ERM)

The Language of Learning Theory : Measuring Error

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \text{ outputs } h \in \mathcal{H}$$

The Language of Learning Theory : Measuring Error

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \text{ outputs } h \in \mathcal{H}$$

Empirical Error: what we **can** calculate

$$\text{err}_S(h) = \frac{1}{m} \sum_{i_1}^{m} \mathbf{1}[h(x_i) \neq y_i]$$

The Language of Learning Theory : Measuring Error

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \text{ outputs } h \in \mathcal{H}$$

Empirical Error: what we **can** calculate

$$\text{err}_S(h) = \frac{1}{m} \sum_{i_1}^{m} \mathbf{1}[h(x_i) \neq y_i]$$

and what $\mathcal{A}$ typically minimizes:

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\text{err}_S(h))$$

The Language of Learning Theory : Measuring Error

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \text{ outputs } h \in \mathcal{H}$$

Empirical Error: what we **can** calculate

$$\text{err}_S(h) = \frac{1}{m} \sum_{i_1}^{m} \mathbf{1}[h(x_i) \neq y_i]$$

and what $\mathcal{A}$ typically minimizes:

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} \left(\text{err}_S(h)\right)$$

True Error: what we **want** to minimize

$$\text{err}_{\mathcal{D}}(h) = \mathbf{Pr}_{x \sim \mathcal{D}}\left[h(x) \neq f(x)\right]$$

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\mathrm{err}_S(h))$$

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\mathrm{err}_S(h))$$

What is the true error $\mathrm{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} \left(\mathrm{err}_S(h)\right)$$

What is the true error $\mathrm{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}}\left[h(x) \neq f(x)\right] < \epsilon$

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\mathrm{err}_S(h))$$

What is the true error $\mathrm{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

ϵ is a small positive number representing accuracy.
$\epsilon = 0.1$ **means:**
hypothesis h is *correct* on 90% of the data points*.

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\mathrm{err}_S(h))$$

What is the true error $\mathrm{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}}[h(x) \neq f(x)] < \epsilon$

Approximately Correct!

ϵ is a small positive number representing accuracy.
$\epsilon = 0.1$ **means:**
hypothesis h is *correct* on 90% of the data points*.

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\mathrm{err}_S(h))$$

What is the true error $\mathrm{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

Approximately Correct!

Really??

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\text{err}_S(h))$$

What is the true error $\text{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

Approximately Correct!

Really??

S is drawn **randomly** from the distribution $\mathcal{D}$.

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\text{err}_S(h))$$

What is the true error $\text{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

Approximately Correct!

Really??

Can only guarantee low error with high probability over the draw of training data S.

$$\mathbf{Pr}_{S \sim \mathcal{D}^m} [\text{err}_{\mathcal{D}}(h) < \epsilon] > 1 - \delta$$

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\text{err}_S(h))$$

What is the true error $\text{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

Approximately Correct!

Really??

Can only guarantee low error with high probability over the draw of training data S.

$$\mathbf{Pr}_{S \sim \mathcal{D}^m} [\text{err}_{\mathcal{D}}(h) < \epsilon] > 1 - \delta$$

δ is a small positive number representing probability of failure.
$\delta = 0.2$ **and** $\epsilon = 0.1$ **means:**
80% of the time, $\mathcal{A}$ outputs a hypothesis that is correct on 90% of the data points.

'Probably Approximately Correct' Learning

The learning algorithm $\mathcal{A}$ reports a hypothesis h when given training data S.

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \sim \mathcal{D}^m$$

$$\mathcal{A}(S) \equiv \arg\min_{h \in \mathcal{H}} (\text{err}_S(h))$$

What is the true error $\text{err}_{\mathcal{D}}(h)$ of the reported hypothesis h?

Goal: perform well on fresh data: $\mathbf{Pr}_{x \sim \mathcal{D}} [h(x) \neq f(x)] < \epsilon$

Approximately Correct!

Really??

Can only guarantee low error with high probability over the draw of training data S.

$$\mathbf{Pr}_{S \sim \mathcal{D}^m} [\text{err}_{\mathcal{D}}(h) < \epsilon] > 1 - \delta$$

"Probably"

δ is a small positive number representing probability of failure.

$\delta = 0.2$ **and** $\epsilon = 0.1$ **means:**

80% of the time, $\mathcal{A}$ outputs a hypothesis that is correct on 90% of the data points.

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that,

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$,

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$,

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data,

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Are we done?

Leslie Valiant, 1984

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1. **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1 **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?

How to find $\mathcal{A}$?
How many samples (m) needed for (ϵ, δ)-PAC guarantee?
Efficiency of $\mathcal{A}$?

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1. **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?
2. **any target $f \in \mathcal{H}$...**: What about target function f?

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1. **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?

2. **any target $f \in \mathcal{H}$...**: What about target function *f*?
   - What if $f \notin \mathcal{H}$?
   - What if *no true target function exists*?
   - Noise/Errors in labels or features?
   - Errros in training data?

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1. **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?
2. **any target $f \in \mathcal{H}$...**: What about target function f?
3. **any distribution $\mathcal{D}$...**: Does such $\mathcal{D}$ truly exist?

The PAC Definition (Informal)

Definition 3.1

A hypothesis class $\mathcal{H}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$, that, for any desired accuracy $\epsilon > 0$, any desired confidence $\delta < 1$, any *target* $f \in \mathcal{H}$, any distribution $\mathcal{D}$ over data, with probability at least $1 - \delta$, reports a hypothesis $h \in \mathcal{H}$ with $\text{err}_{\mathcal{D}}(h) < \epsilon$.

Considerations

1. **if there exists an algorithm $\mathcal{A}$, that, ...** : What is this algorithm $\mathcal{A}$?
2. **any target $f \in \mathcal{H}$...**: What about target function f?
3. **any distribution $\mathcal{D}$...**: Does such $\mathcal{D}$ truly exist?
   - Future data may diverge
   - Possibly no *fixed* $\mathcal{D}$ even for training data

Break Time!

Questions?

Break Time!

Questions?

Let's take a 10-minute break.
We'll resume at 11am.

References

An excellent introductory course on Theory of ML: TTIC 31250 by Avrim Blum

THE textbook of ML: Understanding Machine Learning: From Theory to Algorithms by Shai Shalev-Shwartz and Shai Ben-David