Machine Learning and Algorithmic Model Theory

Martin Grohe
RWTH Aachen
Part I: Machine learning

- Some background on learning theory
Part I: Machine learning

- Some background on learning theory

Part II: Finite and Algorithmic Model Theory

- A declarative model-theoretic framework
- Learning FO-definable hypotheses on bounded degree structures
- Learning MSO-definable hypotheses on strings
Part I: Machine learning

- Some background on learning theory

Part II: Finite and Algorithmic Model Theory

- A declarative model-theoretic framework
- Learning FO-definable hypotheses on bounded degree structures
- Learning MSO-definable hypotheses on strings

Part III: Open Problems and Research Directions
Part I: Machine Learning
Task
Learn to predict whether an unknown fruit, say, a papaya, is tasty or not (from observations we can make from the outside).

As a training set, we get a sample of papayas that we can try.
Example

Task
Learn to predict whether an unknown fruit, say, a papaya, is tasty or not (from observations we can make from the outside).

As a training set, we get a sample of papayas that we can try.

Remarks

▶ This is an example of a Boolean classification problem: we want to learn a Boolean-valued target function. In general, learning finite-valued functions is known as classification, and learning real valued functions is known as regression.
Task
Learn to predict whether an unknown fruit, say, a papaya, is tasty or not (from observations we can make from the outside).

As a training set, we get a sample of papayases that we can try.

Remarks

- This is an example of a Boolean classification problem: we want to learn a Boolean-valued target function. In general, learning finite-valued functions is known as classification, and learning real valued functions is known as regression.

- We are in a supervised learning setting: we learn from training examples labelled by the correct value.
Example

Task
Learn to predict whether an unknown fruit, say, a papaya, is tasty or not (from observations we can make from the outside).

As a training set, we get a sample of papayas that we can try.

Remarks

- This is an example of a **Boolean classification problem**: we want to learn a Boolean-valued target function. In general, learning finite-valued functions is known as classification, and learning real valued functions is known as regression.

- We are in a supervised learning setting: we learn from training examples labelled by the correct value.

- We are in a passive learning setting: the training examples are given to us, we cannot actively ask for the values for specific new instances.
Feature selection
Maybe based on experience with other fruit, we decide to base our decision on two features of the papayas:

- **colour**, ranging from green through yellow and red to brown
- **softness**, ranging from hard to mushy
Feature selection

Maybe based on experience with other fruit, we decide to base our decision on two features of the papayas:

- **colour**, ranging from green through yellow and red to brown
- **softness**, ranging from hard to mushy

### Training Data

<table>
<thead>
<tr>
<th>colour</th>
<th>softness</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+ = tasty  
- = not tasty
Model selection

We may choose a simple *parametric model*: all papayas whose colour is in a certain range \([c_{\text{min}}, c_{\text{max}}]\) and whose softness is in a range \([s_{\text{min}}, s_{\text{max}}]\).
**Model selection**
We may choose a simple parametric model: all papayas whose colour is in a certain range \([c_{\text{min}}, c_{\text{max}}]\) and whose softness is in a range \([s_{\text{min}}, s_{\text{max}}]\).

**Parameter learning**
(aka parameter estimation)
We try to learn the parameters \(c_{\text{min}}, c_{\text{max}}, s_{\text{min}}, s_{\text{max}}\) in such a way that the resulting hypothesis explains the data best.

In our example, the space of all possible hypotheses is the set of all axis-parallel rectangles in the “colour-softness plane.”
Model selection
We may choose a simple \textit{parametric model}: all papayas whose colour is in a certain range \([c_{\text{min}}, c_{\text{max}}]\) and whose softness is in a range \([s_{\text{min}}, s_{\text{max}}]\).

Parameter learning
(aka parameter estimation)
We try to learn the parameters \(c_{\text{min}}, c_{\text{max}}, s_{\text{min}}, s_{\text{max}}\) in such a way that the resulting hypothesis explains the data best.

In our example, the \textit{space of all possible hypotheses} is the set of all axis-parallel rectangles in the “color-softness plane”.

Model learning

We can also try to learn the model directly and the parameters for this model (if there are any).
Model learning

We can also try to learn the model directly and the parameters for this model (if there are any).

Example (cont’d)
Model learning

We can also try to learn the model directly and the parameters for this model (if there are any).

However, we have to be careful to avoid overfitting.
Model learning

We can also try to learn the model directly and the parameters for this model (if there are any).

However, we have to be careful to avoid overfitting.

Remarks

- Our model (hypothesis space) of rectangles in the plane is extremely simple. Even the feature selection and the mapping of features to real numbers are questionable.
Model learning
We can also try to learn the model directly and the parameters for this model (if there are any).
However, we have to be careful to avoid overfitting.

Remarks

- Our model (hypothesis space) of rectangles in the plane is extremely simple. Even the feature selection and the mapping of features to real numbers are questionable.

- As our features probably won’t determine the tastiness of a papaya alone, a more realistic type of model would just output a probability that the corresponding papayas are tasty for each point in the plane.
Goal
Given the training examples, we want to “learn” a hypothesis that generalises well, that is, is a good approximation of the unknown target function.
Goal
Given the training examples, we want to “learn” a hypothesis that generalises well, that is, is a good approximation of the unknown target function.

Caveat
The training data have to be “typical” data points representative for the whole instance space. We need to make assumptions guaranteeing this, otherwise we can’t expect to be succeed.
**Goal**

Given the training examples, we want to "learn" a hypothesis that generalises well, that is, is a good approximation of the unknown target function.

**Caveat**

The training data have to be "typical" data points representative for the whole instance space. We need to make assumptions guaranteeing this, otherwise we can’t expect to be succeed.

**I.i.d. Assumption**

We assume that all data are drawn from an unknown probability distribution $\mathcal{D}$, the data generating distribution. We assume that the training instances in the training set as well as future instances are independent and identically distributed according to this distribution $\mathcal{D}$. 
Goal
Given the training examples, we want to “learn” a hypothesis that generalises well, that is, is a good approximation of the unknown target function.

Caveat
The training data have to be “typical” data points representative for the whole instance space. We need to make assumptions guaranteeing this, otherwise we can’t expect to be succeed.

I.i.d. Assumption
We assume that all data are drawn from an unknown probability distribution $\mathcal{D}$, the data generating distribution.
We assume that the training instances in the training set as well as future instances are independent and identically distributed according to this distribution $\mathcal{D}$.

Generalisation error
We want to minimise the expected error of our hypothesis on instances drawn from $\mathcal{D}$.
We consider Boolean classification problems.

- We have an instance space $\mathbb{U}$, and our goal is to learn an unknown target function

$$f^* : \mathbb{U} \rightarrow \{0, 1\}.$$
We consider Boolean classification problems.

- We have an instance space $\mathbb{U}$, and our goal is to learn an unknown target function

$$f^* : \mathbb{U} \rightarrow \{0, 1\}.$$ 

- We assume furthermore that there is an unknown data generating probability distribution $\mathcal{D}$ on $\mathbb{U}$. 
We consider Boolean classification problems.

- We have an instance space $\mathcal{U}$, and our goal is to learn an unknown target function

$$f^* : \mathcal{U} \rightarrow \{0, 1\}.$$ 

- We assume furthermore that there is an unknown data generating probability distribution $\mathcal{D}$ on $\mathcal{U}$.

- The generalisation error of a hypothesis $h : \mathcal{U} \rightarrow \{0, 1\}$ is

$$\text{err}_\mathcal{D}(h) := \Pr_{u \sim \mathcal{D}} (h(u) \neq f^*(u)).$$
Our algorithm receives as input a Training sequence

\[ T = \left( (u_1, \lambda_1), \ldots, (u_m, \lambda_m) \right) \in (U \times \{0, 1\})^m, \]

where \( \lambda_i = f^*(u_i) \), and produces a hypothesis \( h_T : U \rightarrow \{0, 1\} \) from some hypothesis class \( \mathcal{H} \).
Our algorithm receives as input a Training sequence

\[ T = ((u_1, \lambda_1), \ldots, (u_m, \lambda_m)) \in (\mathbb{U} \times \{0, 1\})^m, \]

where \( \lambda_i = f^*(u_i) \), and produces a hypothesis \( h_T : \mathbb{U} \to \{0, 1\} \) from some hypothesis class \( \mathcal{H} \).

The training error of a hypothesis \( h : \mathbb{U} \to \{0, 1\} \) (w.r.t. to the training sequence \( T \)) is

\[ \text{err}_T(h) := \frac{1}{m} | \{ h(u_i) \neq \lambda_i \mid i \in [m] \} |. \]

If \( \text{err}_T(h) = 0 \) then \( h \) is consistent with \( T \).
We write \( T \sim D \) to denote that the instances \( u_1, \ldots, u_m \) of a training sequence

\[
T = \left( (u_1, f^*(u_1)), \ldots, (u_m, f^*(u_m)) \right) \in (\mathbb{U} \times \{0, 1\})^m
\]

are drawn independently from \( D \).
We write $T \sim \mathcal{D}$ to denote that the instances $u_1, \ldots, u_m$ of a training sequence

$$T = \left((u_1, f^*(u_1)), \ldots, (u_m, f^*(u_m))\right) \in (\mathbb{U} \times \{0, 1\})^m$$

are drawn independently from $\mathcal{D}$.

**Probably Approximately Correct Learning**

A learning algorithm that on input $T$ produces a hypothesis $h_T$ is a **PAC learning algorithm** if for all $\epsilon, \delta > 0$ there is an $m = m(\epsilon, \delta)$ such that for every probability distribution $\mathcal{D}$ on $\mathbb{U}$

$$\Pr_{T \sim \mathcal{D}, |T| = m} \left( \text{err}_\mathcal{D}(h_T) \leq \epsilon \right) > 1 - \delta.$$
Empirical Risk Minimisation

An ERM algorithm with a hypothesis class $\mathcal{H}$ returns, on input $T$, a hypothesis $h_T$ such that

$$\text{err}_T(h_T) = \min_{h \in \mathcal{H}} \text{err}_T(h).$$
Empirical Risk Minimisation

An ERM algorithm with a hypothesis class $\mathcal{H}$ returns, on input $T$, a hypothesis $h_T$ such that

$$\text{err}_T(h_T) = \min_{h \in \mathcal{H}} \text{err}_T(h).$$

Remarks

- If the target function $f^*$ is in $\mathcal{H}$, then an ERM algorithm always returns a consistent hypothesis.
Empirical Risk Minimisation

An ERM algorithm with a hypothesis class $\mathcal{H}$ returns, on input $T$, a hypothesis $h_T$ such that

$$\text{err}_T(h_T) = \min_{h \in \mathcal{H}} \text{err}_T(h).$$

Remarks

- If the target function $f^*$ is in $\mathcal{H}$, then an ERM algorithm always returns a consistent hypothesis.
  The assumption $f^* \in \mathcal{H}$ is sometimes called realisability assumption.
Empirical Risk Minimisation

An ERM algorithm with a hypothesis class $\mathcal{H}$ returns, on input $T$, a hypothesis $h_T$ such that

$$ \text{err}_T(h_T) = \min_{h \in \mathcal{H}} \text{err}_T(h). $$

Remarks

- If the target function $f^*$ is in $\mathcal{H}$, then an ERM algorithm always returns a consistent hypothesis. The assumption $f^* \in \mathcal{H}$ is sometimes called realisability assumption.
- ERM runs the risk of overfitting, in particular for hypothesis classes $\mathcal{H}$ of high capacity (i.e., “rich” classes).
Theorem (Uniform Convergence)

If the hypothesis class $\mathcal{H}$ is finite and we have a training sequence of length at least $\log |\mathcal{H}|$ (roughly), then for every hypothesis in $\mathcal{H}$ the training error is close to the generalisation error.
Uniform Convergence and Agnostic PAC Learning (informally)

Theorem (Uniform Convergence)

If the hypothesis class $\mathcal{H}$ is finite and we have a training sequence of length at least $\log |\mathcal{H}|$ (roughly), then for every hypothesis in $\mathcal{H}$ the training error is close to the generalisation error.

Theorem (Agnostic PAC Learning)

If the hypothesis class $\mathcal{H}$ is finite, then an ERM algorithm that sees at least $\log |\mathcal{H}|$ examples finds a hypothesis that generalises almost as good as the best hypothesis in $\mathcal{H}$. 
Uniform Convergence and Agnostic PAC Learning (informally)

Theorem (Uniform Convergence)
If the hypothesis class $\mathcal{H}$ is finite and we have a training sequence of length at least $\log |\mathcal{H}|$ (roughly), then for every hypothesis in $\mathcal{H}$ the training error is close to the generalisation error.

Theorem (Agnostic PAC Learning)
If the hypothesis class $\mathcal{H}$ is finite, then an ERM algorithm that sees at least $\log |\mathcal{H}|$ examples finds a hypothesis that generalises almost as good as the best hypothesis in $\mathcal{H}$.

In particular, if the realisability assumption holds then the hypothesis found by the ERM algorithm generalises well.
Theorem (Uniform Convergence)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\epsilon, \delta > 0$ and

$$m \geq \log \left( \frac{2|\mathcal{H}|/\delta}{2\epsilon^2} \right).$$

Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}, |T| = m} \left( \forall h \in \mathcal{H} : |\text{err}_T(h) - \text{err}_\mathcal{D}(h)| \leq \epsilon \right) > 1 - \delta.$$
Theorem (Uniform Convergence)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\epsilon, \delta > 0$ and

$$m \geq \frac{\log \left( \frac{2|\mathcal{H}|}{\delta} \right)}{2\epsilon^2}.$$ Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}, |T|=m} \left( \forall h \in \mathcal{H} : |\text{err}_T(h) - \text{err}_\mathcal{D}(h)| \leq \epsilon \right) > 1 - \delta.$$

Theorem (Agnostic PAC Learning Sample Size Bound)

Consider an ERM algorithm with a finite hypothesis class $\mathcal{H}$ that given $T$ produces a hypothesis $h_T$.

$$m \geq \frac{\log \left( \frac{2|\mathcal{H}|}{\delta} \right)}{2\epsilon^2}.$$
Uniform Convergence and Agnostic PAC Learning

Theorem (Uniform Convergence)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\epsilon, \delta > 0$ and

$$m \geq \frac{\log \left( \frac{2|\mathcal{H}|}{\delta} \right)}{2\epsilon^2}.$$  

Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}, \ |T| = m} \left( \forall h \in \mathcal{H} : |\text{err}_T(h) - \text{err}_\mathcal{D}(h)| \leq \epsilon \right) > 1 - \delta.$$

Theorem (Agnostic PAC Learning Sample Size Bound)

Consider an ERM algorithm with a finite hypothesis class $\mathcal{H}$ that given $T$ produces a hypothesis $h_T$.

Let $\epsilon, \delta > 0$ and $m \geq \log \left( \frac{2|\mathcal{H}|}{\delta} \right)/2\epsilon^2$. 

Uniform Convergence and Agnostic PAC Learning

Theorem (Uniform Convergence)
Let $\mathcal{H}$ be a finite hypothesis class. Let $\epsilon, \delta > 0$ and
\[ m \geq \log \left( \frac{2|\mathcal{H}|}{\delta} \right) / 2\epsilon^2. \]
Then for any data generating distribution $\mathcal{D}$,
\[ \Pr_{T \sim \mathcal{D}} \left( \forall h \in \mathcal{H} : |\text{err}_T(h) - \text{err}_\mathcal{D}(h)| \leq \epsilon \right) > 1 - \delta. \]

Theorem (Agnostic PAC Learning Sample Size Bound)
Consider an ERM algorithm with a finite hypothesis class $\mathcal{H}$ that given $T$ produces a hypothesis $h_T$.
Let $\epsilon, \delta > 0$ and
\[ m \geq \log \left( \frac{2|\mathcal{H}|}{\delta} \right) / 2\epsilon^2. \]
Let $\mathcal{D}$ be a data generating distribution and
\[ h^* = \arg \min_{h \in \mathcal{H}} \text{err}_\mathcal{D}(h). \]
Uniform Convergence and Agnostic PAC Learning

Theorem (Uniform Convergence)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\epsilon, \delta > 0$ and 
$m \geq \frac{\log \left( \frac{2|\mathcal{H}|}{\delta} \right)}{2\epsilon^2}$. Then for any data generating distribution $\mathcal{D}$,

$$\Pr \left( \forall h \in \mathcal{H} : \left| \text{err}_T(h) - \text{err}_\mathcal{D}(h) \right| \leq \epsilon \right) > 1 - \delta.$$ 

Theorem (Agnostic PAC Learning Sample Size Bound)

Consider an ERM algorithm with a finite hypothesis class $\mathcal{H}$ that given $T$ produces a hypothesis $h_T$. Let $\epsilon, \delta > 0$ and $m \geq \frac{\log \left( \frac{2|\mathcal{H}|}{\delta} \right)}{2\epsilon^2}$.

Let $\mathcal{D}$ be a data generating distribution and $h^* = \arg \min_{h \in \mathcal{H}} \text{err}_\mathcal{D}(h)$. Then

$$\Pr \left( \left| \text{err}_\mathcal{D}(h_T) - \text{err}_\mathcal{D}(h^*) \right| \leq 2\epsilon \right) > 1 - 2\delta.$$
Example (Discretisation Trick)

Suppose that our hypothesis space consists of hyperplanes in 10-dimensional Euclidean space that our learning algorithm specifies by giving a normal vector in 64 bit floating point numbers.
Example (Discretisation Trick)

Suppose that our hypothesis space consists of hyperplanes in 10-dimensional Euclidean space that our learning algorithm specifies by giving a normal vector in 64 bit floating point numbers. Thus the size of the hypothesis space of the algorithm is $2^{640}$. 

We let $\epsilon = 0.1$ and $\delta = 0.05$. Then if our algorithm receives a training sequence of size (at least) $2 \log (\frac{4|H|}{\delta}) \epsilon^2 \approx 129.264$, produces a hypothesis (a hyperplane) consistent with the training data, then with probability at least 95% the generalization error is less than 10%. 


Example (Discretisation Trick)

Suppose that our hypothesis space consists of hyperplanes in 10-dimensional Euclidean space that our learning algorithm specifies by giving a normal vector in 64 bit floating point numbers. Thus the size of the hypothesis space of the algorithm is $2^{640}$.

We let $\epsilon = 0.1$ and $\delta = 0.05$. Then if

1. our algorithm receives a training sequence of size (at least)

$$
\frac{2 \log(4|\mathcal{H}|/\delta)}{\epsilon^2} \approx 129.264,
$$

2. produces a hypothesis (a hyperplane) consistent with the training data,

then with probability at least 95% the generalization error is less than 10%.
VC Dimension

The VC dimension of a class $\mathcal{H} \subseteq 2^\mathbb{U}$ is the maximum cardinality of a set $X \subseteq \mathbb{U}$ such that

$$\{ h|_X \mid h \in \mathcal{H} \} = 2^X$$

(we say that $X$ is shattered by $\mathcal{H}$), or $\infty$ if this maximum does not exist.
VC Dimension

The **VC dimension** of a class $\mathcal{H} \subseteq 2^\mathbb{U}$ is the maximum cardinality of a set $X \subseteq \mathbb{U}$ such that

$$\{ h|_X \mid h \in \mathcal{H} \} = 2^X$$

(we say that $X$ is **shattered** by $\mathcal{H}$), or $\infty$ if this maximum does not exist.

**Theorem (Uniform Convergence for Bounded VC Dimension)**

Let $\mathcal{H}$ be a hypothesis class of finite VC-dimension $d$.

Let $\epsilon, \delta > 0$ and $m \geq c_d \log \left( \frac{1}{\delta} \right) \frac{\epsilon}{2}$ (for a suitable constant $c$).

Then for any data generating distribution $D$,

$$\Pr_{T \sim D} |T| = m \left( \forall h \in \mathcal{H}: \left| \text{err}_T(h) - \text{err}_D(h) \right| \leq \epsilon \right) > 1 - \delta.$$
VC Dimension

The **VC dimension** of a class $\mathcal{H} \subseteq 2^\mathcal{U}$ is the maximum cardinality of a set $X \subseteq \mathcal{U}$ such that

$$\{ h|_X \mid h \in \mathcal{H} \} = 2^X$$

(we say that $X$ is **shattered** by $\mathcal{H}$), or $\infty$ if this maximum does not exist.

**Theorem (Uniform Convergence for Bounded VC Dimension)**

Let $\mathcal{H}$ be a hypothesis class of finite VC-dimension $d$.

Let $\epsilon, \delta > 0$ and $m \geq c \frac{d + \log(1/\delta)}{\epsilon^2}$ (for a suitable constant $c$).
VC Dimension

The VC dimension of a class $\mathcal{H} \subseteq 2^U$ is the maximum cardinality of a set $X \subseteq U$ such that

$$\{ h | X \mid h \in \mathcal{H} \} = 2^X$$

(we say that $X$ is shattered by $\mathcal{H}$), or $\infty$ if this maximum does not exist.

Theorem (Uniform Convergence for Bounded VC Dimension)

Let $\mathcal{H}$ be a hypothesis class of finite VC-dimension $d$.

Let $\epsilon, \delta > 0$ and $m \geq c \frac{d + \log(1/\delta)}{\epsilon^2}$ (for a suitable constant $c$).

Then for any data generating distribution $D$,

$$\Pr_{T \sim D, |T| = m} \left( \forall h \in \mathcal{H} : |\text{err}_T(h) - \text{err}_D(h)| \leq \epsilon \right) > 1 - \delta.$$
Regularisation

If the hypothesis class $\mathcal{H}$ is infinite and of unbounded VC dimension, ERM may not be a wise choice, because it tends to overfit.
If the hypothesis class $\mathcal{H}$ is infinite and of unbounded VC dimension, ERM may not be a wise choice, because it tends to overfit.

To counteract overfitting, we can add a regularisation term to the function we minimise:

$$h_T = \arg \min_{h \in \mathcal{H}} \left( \text{err}_T(h) + \rho(\text{cost}(h)) \right)$$

where:

- $\text{cost}$ is a function that associates a “complexity cost” to every hypothesis. Examples: bit precision of a real vector, degree of a polynomial, quantifier rank of a formula
- $\rho$ is an arbitrary monotone function, often just a linear function.
Regularisation

If the hypothesis class $\mathcal{H}$ is infinite and of unbounded VC dimension, ERM may not be a wise choice, because it tends to overfit.

To counteract overfitting, we can add a regularisation term to the function we minimise:

$$h_\mathcal{T} = \arg \min_{h \in \mathcal{H}} \left( \text{err}_{\mathcal{T}}(h) + \rho(\text{cost}(h)) \right)$$

where:

- cost is a function that associates a “complexity cost” to every hypothesis.
If the hypothesis class $\mathcal{H}$ is infinite and of unbounded VC dimension, ERM may not be a wise choice, because it tends to overfit.

To counteract overfitting, we can add a regularisation term to the function we minimise:

$$h_T = \arg \min_{h \in \mathcal{H}} \left( \text{err}_T(h) + \rho(\text{cost}(h)) \right)$$

where:

- cost is a function that associates a “complexity cost” to every hypothesis.

**Examples:** bit precision of a real vector, degree of a polynomial, quantifier rank of a formula.
Regularisation

If the hypothesis class $\mathcal{H}$ is infinite and of unbounded VC dimension, ERM may not be a wise choice, because it tends to overfit.

To counteract overfitting, we can add a regularisation term to the function we minimise:

$$h_T = \arg\min_{h \in \mathcal{H}} \left( \text{err}_T(h) + \rho(\text{cost}(h)) \right)$$

where:

- **cost** is a function that associates a “complexity cost” to every hypothesis.
  
  **Examples**: bit precision of a real vector, degree of a polynomial, quantifier rank of a formula

- **$\rho$** is an arbitrary monotone function, often just a linear function.
The Moral

- Algorithmically, machine learning amounts to solving (often very complicated) minimisation problems (often on large datasets).
Algorithmically, machine learning amounts to solving (often very complicated) minimisation problems (often on large datasets).

The art is to come up with good hypotheses and the right regularisation terms.
Algorithmically, machine learning amounts to solving (often very complicated) minimisation problems (often on large datasets).

The art is to come up with good hypotheses and the right regularisation terms.

If we keep an eye on the capacity of the hypothesis class and the number of training examples, we may even be able to bound the generalisation error.
Declarative Approach to ML

Today’s ML practice

▷ **Algorithmic focus**: goal is to approximate an unknown function as well as possible (rather than understanding the function)
Declarative Approach to ML

Today’s ML practice

- **Algorithmic focus:** goal is to approximate an unknown function as well as possible (rather than understanding the function)
- Models are determined by the algorithm and (often) have little meaning beyond that.
Declarative Approach to ML

Today’s ML practice

- **Algorithmic focus**: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- Models are determined by the algorithm and (often) have little meaning beyond that.
- It is difficult for a non-expert user which algorithm (with which “hyperparameter” settings) to use
Declarative Approach to ML

Today’s ML practice

- **Algorithmic focus**: goal is to approximate an unknown function as well as possible (rather than understanding the function)
- Models are determined by the algorithm and (often) have little meaning beyond that.
- It is difficult for a non-expert user which algorithm (with which “hyperparameter” settings) to use
- It is difficult to understand and explain what the models do.
Declarative Approach to ML

Today’s ML practice

▶ Algorithmic focus: goal is to approximate an unknown function as well as possible (rather than understanding the function)
▶ Models are determined by the algorithm and (often) have little meaning beyond that.
▶ It is difficult for a non-expert user which algorithm (with which “hyperparameter” settings) to use
▶ It is difficult to understand and explain what the models do.

Declarative approach
Try to separate model from solver as far as possible.
Terminology regarding models, hypotheses, etc. is confusing and sometimes inconsistent.

 Examples
  - mixture of two Gaussian distributions, parameters are the means and variances,
  - class of all halfspaces in $d$-dimensional Euclidean space, parameters are the coefficients of a normal vector,
  - neural network of a certain topology, parameters are the weights.
What is “Model” Supposed to Mean?

Terminology regarding models, hypotheses, etc. is confusing and sometimes inconsistent.

A parametric model is chosen by the human expert, maybe implicitly by the choice of the learning algorithm.
What is “Model” Supposed to Mean?

Terminology regarding models, hypotheses, etc. is confusing and sometimes inconsistent.

A **parametric model** is chosen by the human expert, maybe implicitly by the choice of the learning algorithm.

It encapsulates background knowledge and assumptions on the nature of the target function and data generating distribution.
What is “Model” Supposed to Mean?

Terminology regarding models, hypotheses, etc. is confusing and sometimes inconsistent.

A parametric model is chosen by the human expert, maybe implicitly by the choice of the learning algorithm.

It encapsulates background knowledge and assumptions on the nature of the target function and data generating distribution.

Examples

▶ mixture of two Gaussian distributions, parameters are the means and variances,
▶ class of all halfspaces in $d$-dimensional Euclidean space, parameters are the coefficients of a normal vector,
▶ neural network of a certain topology, parameters are the weights.
What is “Model” Supposed to Mean?

Terminology regarding models, hypotheses, etc. is confusing and sometimes inconsistent.

A **parametric model** is chosen by the human expert, maybe implicitly by the choice of the learning algorithm.

It encapsulates background knowledge and assumptions on the nature of the target function and data generating distribution.

**Examples**

- mixture of two Gaussian distributions, parameters are the means and variances,
- class of all halfspaces in $d$-dimensional Euclidean space, parameters are the coefficients of a normal vector,
- neural network of a certain topology, parameters are the weights.

The learning algorithm “learns” (“estimates”) the parameters. It generates a **hypothesis**, the function defined by model and parameters.
More Remarks on the Terminology

- What we call “hypothesis” is often called “model” (this is the typical usage of the term “model” in science and engineering).
  Obviously, this use of the term model is no the same as our use in “model theory”.
What we call “hypothesis” is often called “model” (this is the typical usage of the term “model” in science and engineering). Obviously, this use of the term model is no the same as our use in “model theory”.

In learning theory, sometimes a distinction is made between a concept space, a wider class of functions that is supposed to contain the target function, and the hypothesis space, consisting of the function the learning algorithm may produce.
More Remarks on the Terminology

- What we call “hypothesis” is often called “model” (this is the typical usage of the term “model” in science and engineering). Obviously, this use of the term model is no the same as our use in “model theory”.

- In learning theory, sometimes a distinction is made between a concept space, a wider class of functions that is supposed to contain the target function, and the hypothesis space, consisting of the function the learning algorithm may produce.

- There is also non-parametric learning, where the hypothesis can be an arbitrary function constrained only by the data. An example is the nearest neighbour algorithm.
We only scratched the surface. Details on the material covered in this section, references, and a lot more can be found in:
We only scratched the surface. Details on the material covered in this section, references, and a lot more can be found in:

Part II:
Finite and Algorithmic Model Theory
Background structure and instance space
Finite or infinite structure $B$ with universe $U(B)$. 

Parameter model
Formula $\phi(\overline{x}; \overline{y})$ of some logic $L$. 

$\overline{x} = (x_1, \ldots, x_k)$ instance variables. 

$\overline{y} = (y_1, \ldots, y_\ell)$ (for some $\ell$) parameter variables.

Hypotheses
For each parameter tuple $\overline{v} \in U(B)^\ell$ a function $J_{\phi}(\overline{x}; \overline{v}) \in K_B : U \to \{0, 1\}$ by 

$J_{\phi}(\overline{x}; \overline{v})\left(\overline{u}\right) := 
\begin{cases} 
1 & \text{if } B \models \phi(\overline{u}; \overline{v}), \\
0 & \text{otherwise} 
\end{cases}$.
Background structure and instance space

Finite or infinite structure $B$ with universe $U(B)$.  
Instance space is $\mathbb{U} = U(B)^k$ for some $k$. 

A Model-Theoretic Framework
A Model-Theoretic Framework

Background structure and instance space

Finite or infinite structure $B$ with universe $U(B)$. Instance space is $\mathbb{U} = U(B)^k$ for some $k$.

Parametric model

Formula $\varphi(\bar{x}; \bar{y})$ of some logic $L$.

$\bar{x} = (x_1, \ldots, x_k)$ instance variables.

$\bar{y} = (y_1, \ldots, y_\ell)$ (for some $\ell$) parameter variables.
A Model-Theoretic Framework

Background structure and instance space
Finite or infinite structure $B$ with universe $U(B)$.
Instance space is $U = U(B)^k$ for some $k$.

Parametric model
Formula $\varphi(\bar{x} ; \bar{y})$ of some logic $L$.
$\bar{x} = (x_1, \ldots, x_k)$ instance variables.
$\bar{y} = (y_1, \ldots, y_\ell)$ (for some $\ell$) parameter variables.

Hypotheses
For each parameter tuple $\bar{v} \in U(B)^\ell$ a function
$\llbracket \varphi(\bar{x} ; \bar{v}) \rrbracket^B : U \to \{0, 1\}$ by

$$\llbracket \varphi(\bar{x} ; \bar{v}) \rrbracket^B(\bar{u}) := \begin{cases} 1 & \text{if } B \models \varphi(\bar{u} ; \bar{v}), \\ 0 & \text{otherwise.} \end{cases}$$
Remember the papaya example. As our background structure, we may take $B := (\mathbb{R}, +, -, \cdot, \leq)$. The instance space $\mathbb{R}^2$ (that is, we have dimension $k = 2$).
Remember the papaya example.  
As our background structure, we may take $B := (\mathbb{R}, +, −, \cdot, \leq)$.  
The instance space $\mathbb{R}^2$ (that is, we have dimension $k = 2$).

- To express the “rectangle model”, we use the FO-formula

$$
\varphi(x_1, x_2; y_1, \ldots, y_4) := y_1 \leq x_1 \leq y_2 \land y_3 \leq x_2 \leq y_4.
$$
Remember the papaya example.

As our background structure, we may take $B := (\mathbb{R}, +, −, \cdot, \leq)$.

The instance space $\mathbb{R}^2$ (that is, we have dimension $k = 2$).

- To express the “rectangle model”, we use the FO-formula

$$\phi(x_1, x_2, y_1, \ldots, y_4) := y_1 \leq x_1 \leq y_2 \land y_3 \leq x_2 \leq y_4.$$  

- The following FO-formula expresses a “circle model”

$$\phi(x_1, x_2, y_1, y_2, y_3) := (x_1 - y_1)^2 + (x_2 - y_2)^2 \leq y_3^2.$$
Other logical frameworks

Inductive logic programming (Muggleton ~ 1990)

Classic framework in algorithmic learning theory, learning problem is encoded in a logic program.
Other logical frameworks

Inductive logic programming (Muggleton ∼ 1990)
Classic framework in algorithmic learning theory, learning problem is encoded in a logic program.

Learning Reductions (Crouch, Immerman, Moss, Kaiser, Jordan ∼ 2010)
Learn syntactical interpretations between finite structures.
Other logical frameworks

Inductive logic programming (Muggleton ∼ 1990)
Classic framework in algorithmic learning theory, learning problem is encoded in a logic program.

Learning Reductions (Crouch, Immerman, Moss, Kaiser, Jordan ∼ 2010)
Learn syntactical interpretations between finite structures.

Also learning in database theory and verification contexts.
Hypothesis space

\[ \mathcal{H}_{B,\varphi} = \left\{ \varphi(\bar{x}; \bar{v})^B \mid \bar{v} \in U(B)^\ell \right\}. \]
VC Dimension

Hypothesis space

\[ \mathcal{H}_{B,\varphi} = \left\{ [\varphi(\bar{x} ; \bar{v})]^B \mid \bar{v} \in U(B)^\ell \right\}. \]

Theorem (G., Turan 2004)

1. Suppose \( \varphi(\bar{x} ; \bar{y}) \) is an MSO-formula and \( B \) a class of structures of bounded tree width or of bounded clique width. Then there is a constant \( c \) such that

\[ \text{VC-dim} \left( \mathcal{H}_{B,\varphi} \right) \leq c \quad \text{for all} \ B \in \mathcal{B}. \]
VC Dimension

Hypothesis space

\[ \mathcal{H}_{B,\varphi} = \left\{ \llbracket \varphi(\bar{x} ; \bar{v}) \rrbracket^B \mid \bar{v} \in U(B)^\ell \right\} . \]

Theorem (G., Turan 2004)

1. Suppose \( \varphi(\bar{x} ; \bar{y}) \) is an MSO-formula and \( B \) a class of structures of bounded tree width or of bounded clique width. Then there is a constant \( c \) such that

\[ \text{VC-dim} \left( \mathcal{H}_{B,\varphi} \right) \leq c \quad \text{for all } B \in \mathcal{B}. \]

2. Suppose \( \varphi(\bar{x} ; \bar{y}) \) is an FO-formula and \( B \) a class of structures of bounded local tree width or of bounded local clique width. Then there is a constant \( c \) such that

\[ \text{VC-dim} \left( \mathcal{H}_{B,\varphi} \right) \leq c \quad \text{for all } B \in \mathcal{B}. \]
Learning Algorithms

Input
Background structure $B$, training sequence $T \in (U(B)^k \times \{0, 1\})^m$
Input
Background structure $B$, training sequence $T \in (U(B)^k \times \{0, 1\})^m$

Parameter Learning
For a fixed formula $\varphi(\bar{x}; \bar{y})$, compute parameter tuple $\bar{v}$ such that $[\varphi(\bar{x}; \bar{v})]^B$ has small generalisation error.
Learning Algorithms

Input
Background structure $B$, training sequence $T \in (U(B)^k \times \{0, 1\})^m$

Parameter Learning
For a fixed formula $\varphi(\bar{x} ; \bar{y})$, compute parameter tuple $\bar{v}$ such that $\lbrack \varphi(\bar{x} ; \bar{v}) \rbrack^B$ has small generalisation error.

Model Learning
Compute formula $\varphi(\bar{x} ; \bar{y})$ and parameter tuple $\bar{v}$ such that $\lbrack \varphi(\bar{x} ; \bar{v}) \rbrack^B$ has small generalisation error.
If the background structure is not too large, we can simply regard it as part of the input.
Local Access to the Background Structure

- If the background structure is not too large, we can simply regard it as part of the input.
- If $B$ is very large or infinite, we assume some form of oracle access to $B$. 


If the background structure is not too large, we can simply regard it as part of the input.

If \( B \) is very large or infinite, we assume some form of oracle access to \( B \).

In this talk, we restrict our attention to a uniform cost model: it takes one step to process a single element of the structure.
Local Access to the Background Structure

- If the background structure is not too large, we can simply regard it as part of the input.
- If $B$ is very large or infinite, we assume some form of oracle access to $B$.
- In this talk, we restrict our attention to a uniform cost model: it takes one step to process a single element of the structure.
- One important case is that $B$ is the field of real numbers, in which case we may use the BSS model.
Local Access to the Background Structure

- If the background structure is not too large, we can simply regard it as part of the input.
- If $B$ is very large or infinite, we assume some form of oracle access to $B$.
- In this talk, we restrict our attention to a uniform cost model: it takes one step to process a single element of the structure.
- One important case is that $B$ is the field of real numbers, in which case we may use the BSS model.
- In the following, we are mainly interested in the case that $B$ is a very large finite structure, and we propose a local access model: we can access the neighbours of elements we hold in memory (initially those in the training sequence).
Parameter Learning

We can use ERM, that is, minimise the training error.
Parameter Learning

We can use ERM, that is, minimise the training error. If \( n := B \) is finite, the hypothesis space has size \( |B|^\ell \), thus \( m = \Omega(\ell \log n) \) examples guarantee probable approximate correctness.
Parameter Learning

We can use ERM, that is, minimise the training error. If $n := B$ is finite, the hypothesis space has size $|B|^\ell$, thus $m = \Omega(\ell \log n)$ examples guarantee probable approximate correctness.

We may also use bounds on the VC-dimension if they are available.
Parameter Learning
We can use ERM, that is, minimise the training error.
If \( n := B \) is finite, the hypothesis space has size \(|B|^\ell\), thus \( m = \Omega(\ell \log n) \) examples guarantee probable approximate correctness.

We may also use bounds on the VC-dimension if they are available.

Model Learning
The proper approach would be regularisation for a cost function depending on the quantifier rank \( q \) and number \( \ell \) of parameters.
Empirical Risk Minimisation

Parameter Learning
We can use ERM, that is, minimise the training error.
If $n := B$ is finite, the hypothesis space has size $|B|^\ell$, thus $m = \Omega(\ell \log n)$ examples guarantee probable approximate correctness.
We may also use bounds on the VC-dimension if they are available.

Model Learning
The proper approach would be regularisation for a cost function depending on the quantifier rank $q$ and number $\ell$ of parameters.
We avoid this by viewing $q$ and $\ell$ as fixed (see this as a “data complexity” approach).
Goal
Design learning algorithms with a running time polynomial in the size $m = |T|$ of the training sequence, independently of the size of $B$ (or maybe with a polylogarithmic or at least sublinear dependence on $B$).
Sublinear Parameter Learning is Impossible

Example (G., Ritzert 2017)
Background structure $B$ with single unary relation $P$. 
Example (G., Ritzert 2017)

Background structure $B$ with single unary relation $P$.

Consider the formula

$$\varphi(x; y) = P(y)$$
Example (G., Ritzert 2017)

Background structure $B$ with single unary relation $P$.

Consider the formula

$$\varphi(x; y) = P(y)$$

Then

$$\llbracket \varphi(x; v) \rrbracket^B(u) = \begin{cases} 
1 & \text{for all } u \text{ if } v \in P(B), \\
0 & \text{for all } u \text{ otherwise}.
\end{cases}$$
Sublinear Parameter Learning is Impossible

Example (G., Ritzert 2017)
Background structure $B$ with single unary relation $P$.

Consider the formula

$$\varphi(x ; y) = P(y)$$

Then

$$[\varphi(x ; v)]^B(u) = \begin{cases} 
1 & \text{for all } u \text{ if } v \in P(B), \\
0 & \text{for all } u \text{ otherwise.}
\end{cases}$$

If we see just one positive example, we know we should choose the '1' hypothesis, but it may take time linear in $B$ to find an element of $P(B)$. 
Theorem (G., Ritzert 2017)

There is a model learner for FO running in time

\[(d + m)^{O(1)}\]

where

- \(m = |T|\) is the length of the training sequence
- \(d\) is the maximum degree of the background structure \(B\)
- the constant hidden in the \(O(1)\) depends on \(k, q, \ell\).
We model strings as structures with a linear order, so they do not have bounded degree. **Local access** in strings means that we can only access neighbours of positions we already see.
We model strings as structures with a linear order, so they do not have bounded degree. Local access in strings means that we can only access neighbours of positions we already see.

**Theorem (G., Löding, Ritzert 2017)**

There is no sublinear model learner for FO (even the $\Sigma^1$-fragment) over strings.
The local access model for strings is very weak.
We also study a stronger model where we allow the learning algorithm a pre-processing of the background structure, which it may use to build an index.
The local access model for strings is very weak.
We also study a stronger model where we allow the learning algorithm a pre-processing of the background structure, which it may use to build an index.

**Theorem (G., Löding, Ritzert 2017)**

*Model learning and parameter learning for MSO over strings is possible with linear time pre-processing and learning time $m^{O(1)}$.***
The local access model for strings is very weak. We also study a stronger model where we allow the learning algorithm a pre-processing of the background structure, which it may use to build an index.

**Theorem (G., Löding, Ritzert 2017)**

*Model learning and parameter learning for MSO over strings is possible with linear time pre-processing and learning time $m^{O(1)}$.*

**Proof Idea**
Use a Simon factorisation tree for a suitable monoid as index structure.
Part III: Open Problems and Research Directions
Sublinear Learning

- Are there sublinear model learning algorithms (possibly after pre-processing)
  - for FO on planar graphs and extensions up to nowhere dense and bounded local clique width,
  - for MSO on trees, structures of bounded tree or clique width?
- What are suitable access models?
- What about logics with counting operators?
- What about finite variable logics?
A Few Technical Problems

Sublinear Learning

- Are there sublinear model learning algorithms (possibly after pre-processing)
  - for FO on planar graphs and extensions up to nowhere dense and bounded local clique width,
  - for MSO on trees, structures of bounded tree or clique width?
- What are suitable access models?
- What about logics with counting operators?
- What about finite variable logics?

Fixed-parameter tractable learning algorithms

In the same setting we may look for algorithms running in time $f(k, \ell, q)n^c$, where $c$ does not depend on $k, \ell, q$. 
Design learning algorithms for our model-theoretic framework that actually work in practice (even if they have no theoretical guarantees).
Directions

- Design logics expressing relevant ML models that at the same time allow feasible learning algorithms.
Directions

- Design logics expressing relevant ML models that at the same time allow feasible learning algorithms.
- Study notions of approximate satisfiability, sampling techniques for query answering, approximate query answering.
Directions

- Design logics expressing relevant ML models that at the same time allow feasible learning algorithms.
- Study notions of approximate satisfiability, sampling techniques for query answering, approximate query answering.
- Extend the framework from Boolean classifications to general classification and regression problems.
Directions

- Design logics expressing relevant ML models that at the same time allow feasible learning algorithms.
- Study notions of approximate satisfiability, sampling techniques for query answering, approximate query answering.
- Extend the framework from Boolean classifications to general classification and regression problems.
- Use our framework to query and understand complicated ML models such as Neural Nets.
References

