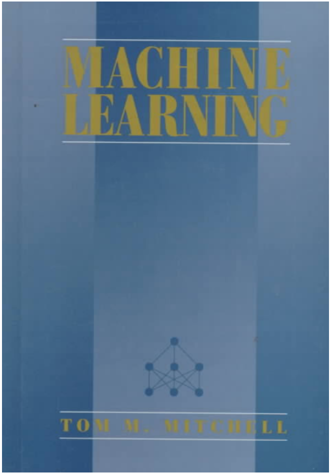
Introductory Course: Machine Learning (WWI15B4)

Concept Learning and the Hypothesis Space

Fabio Ferreira, David Bethge

DHBW Karlsruhe

Recommended Literature



Questions and more questions...

Some questions we will be answering (from a theoretical perspective) in this lecture:

- What does "a model generalizes well" mean?
- Why classification cannot work without making a-priori assumptions?
- What is a bias and what different types do we usually consider?
- Why can we only learn so much what's in the data we provide to a learning algorithm?
- Why are models with lots of parameters usually not a blessing but a curse? (curse of dimensionality)

Motivation

Statements like

- "Dogs bark."
- "Cars drive."
- "I enjoy going to the theater."

require the knowledge of *concepts* (having learned about larger sets like animals, transportation, entertainment, ...)

Motivation

Statements like

- "Dogs bark." $\rightarrow$ **animals**
- "Cars drive." $\rightarrow$ **transportation**
- "I enjoy going to the theater." $\rightarrow$ **entertainment**

require the knowledge of *concepts* (having learned about larger sets like animals, transportation, entertainment, ...)

Motivation

Statements like

- "Dogs bark." → **animals**
- "Cars drive." → **transportation**
- "I enjoy going to the theater." → **entertainment**

require the knowledge of *concepts* (having learned about larger sets like animals, transportation, entertainment, ...)

Concept

Description of a subset of objects/events which are defined over a larger set.

Why are concepts important?

- Learning often involves inducing general functions from specific (training) examples
- building block for:
  - classification, events
  - inferring possible consequences
  - creating more complex knowledge (relations etc.)
- concept learning: "searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples" [Mitchell, 1997] $\Rightarrow$ **many machine learning algorithms perform hypothesis space search** (e.g. ID3, SVM)

Concept Learning

Concept Learning

Automatic inference of a boolean-valued function based on training examples that yields **true** if an object (e.g. dog) is member of its larger (e.g. animals), **false** if not a member.

- Input: training samples (either member or non-member)
- Goal: automatic inference of definition of the underlying concept

Concept Learning

Example:
$foo(lion) \rightarrow true$
$foo(giraffe) \rightarrow true$
$foo(jackal) \rightarrow true$
$foo(elephant) \rightarrow true$
$foo(cougar) \rightarrow false$
$foo(snowleopard) \rightarrow false$

Concept Learning

Example:
$foo(lion) \rightarrow true$
$foo(giraffe) \rightarrow true$
$foo(jackal) \rightarrow true$
$foo(elephant) \rightarrow true$
$foo(cougar) \rightarrow false$
$foo(snowleopard) \rightarrow false$
Concept: **animals inhabiting Africa**

Table of Contents

Concept Learning Task

(The following examples are taken from [Mitchell, 1997])

- goal: learn the concept "days on which my friend Aldo enjoys his favorite water sport."
- i.e.: $c : EnjoySport : X \rightarrow \{true, false\}$
- set of instances X: possible days described by 6 attributes:
  - *Sky* (Sunny, Cloudy, Rainy)
  - *AirTemp* (Warm, Cold)
  - *Humidity* (Normal, High))
  - *Wind* (Strong, Weak))
  - *Water* (Warm, Cool)
  - *Forecoast* (Same, Change)
- e.g. <Sunny, Warm, Normal, Strong, Warm, Same>

Concept Learning Task

- set of hypotheses H specified as a vector of six constraints, which can be evaluated as:
  - attribute value (e.g. "Warm")
  - "?" (any value is acceptable)
  - "$\emptyset$" (no value acceptable)
- e.g. <Sunny, Warm, Normal, Strong, ?, Same>
- most **general** hypothesis: $<?, ?, ?, ?, ?, ? >$
- most **specific** hypothesis: $< \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >$

Concept Learning Task

- set of hypotheses H specified as a vector of six constraints, which can be evaluated as:
  - attribute value (e.g. "Warm")
  - "?" (any value is acceptable)
  - "$\emptyset$" (no value acceptable)
- e.g. <Sunny, Warm, Normal, Strong, ?, Same>
- most **general** hypothesis: $<?, ?, ?, ?, ?, ? >$
- most **specific** hypothesis: $< \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >$
- "x satisfies h" if $h(x) = 1$, $\exists x \in X$
- a hypothesis is **consistent** if it correctly classifies all samples
- **goal**: determine $h \in H$ s.t. $h(x) = c(x), \forall x \in X$

Concept Learning Task

Positive and negative examples of *EnjoySport*

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

from [Mitchell, 1997]

Hypothesis Space

How is the hypotheses space shaped?

- the space of hypotheses is implicitly shaped by selecting the hypothesis representation
- number of instances[1] in X: $3 * 2 * 2 * 2 * 2 * 2 = 96$
- number of hypotheses[2] in H: $5 * 4 * 4 * 4 * 4 * 4 = 5120$

Many algorithms address learning by viewing it as *searching the hypothesis space* in order to find hypotheses that best fit the data.

[1] distinct
[2] syntactically distinct

General-to-specific ordering

Example:

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

More general than or equal to: $(\geq)$

Let h_j and h_k be defined over X. Then $h_j \geq h_k$ iff:

$$h_k(x) = 1 \Rightarrow h_j(x) = 1, \forall x \in X$$

General-to-specific ordering

Example:

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

More general than or equal to: $(\geq)$

Let h_j and h_k be defined over X. Then $h_j \geq h_k$ iff:

$$h_k(x) = 1 \Rightarrow h_j(x) = 1, \forall x \in X$$

More general than: $(>)$

Let h_j and h_k be defined over X. Then $h_j > h_k$ iff:

$$h_j \geq h_k \wedge h_k \not\geq h_j$$

General-to-specific ordering

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

Evaluate the following expressions:

- $h2 > h1$?
- $h3 > h2$?

General-to-specific ordering

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

Evaluate the following expressions:

- $h2 > h1$? **true**, since h1 $\Rightarrow$ h2 AND it is NOT h2 $\Rightarrow$ h1 ($\forall$x)
- $h3 > h2$? **false**, since NOT h2 $\Rightarrow$ h3 ($\exists$ x)

General-to-specific ordering

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

Verify the following expressions:

- $h2 > h1$? **true**, since h1 $\Rightarrow$ h2 AND it is NOT h2 $\Rightarrow$ h1 ($\forall$x)
- $h3 > h2$? **false**, since NOT h2 $\Rightarrow$ h3 ($\exists$ x)
- $h3 \geq h1$?

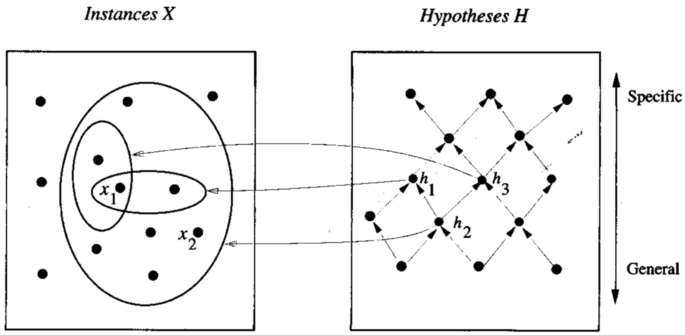
General-to-specific ordering

- h1 = <Sunny, ?, ?, Strong, ?, ?>
- h2 = <Sunny, ?, ?, ?, ?, ?>
- h3 = <Sunny, ?, ?, ?, Cool, ?>

Verify the following expressions:

- $h2 > h1$? **true**, since h1 $\Rightarrow$ h2 AND it is NOT h2 $\Rightarrow$ h1 ($\forall$x)
- $h3 > h2$? **false**, since h2 NOT $\Rightarrow$ h3 ($\exists$ x)
- $h3 \geq h1$? **false**, x=<Sunny, Warm, High, Strong, Warm, Same> $\Rightarrow h1(x) = 1 \not\Rightarrow h3(x) = 1$

Visualization of X and H



$x_1 = \langle Sunny, Warm, High, Strong, Cool, Same \rangle$
$x_2 = \langle Sunny, Warm, High, Light, Warm, Same \rangle$

$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$
$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$
$h_3 = \langle Sunny, ?, ?, ?, Cool, ? \rangle$

from [Mitchell, 1997]

Table of Contents

Learning as Search in the Hypothesis Space

1. Search from general to specific
   - initial point is the most general hypothesis <?,...,?>
   - positive samples are omitted
   - negative samples are used for specialization
2. Search from specific to general
   - initial point is the most specific hypothesis <∅,...,∅>
   - negative samples are omitted
   - positive samples are used for generalization
3. Combination of both: Candidate Elimination algorithm (later)

Learning as Search in the Hypothesis Space

1. Search from general to specific
   - initial point is the most general hypothesis <?,...,?>
   - positive samples are omitted
   - negative samples are used for specialization
2. **Search from specific to general**
   - initial point is the most specific hypothesis <∅,...,∅>
   - negative samples are omitted
   - positive samples are used for generalization
3. Combination of both: Candidate Elimination algorithm (later)

Specific-to-General Algorithm

1. initialize h as most specific hypothesis in H
2. for each positive training sample
   - for each attribute constraint a_i in h $<a_0,...,a_n>$
     - if a_i is satisfied by x: do nothing
     - else replace a_i in h by the next more general constraint that is satisfied by x
3. return hypothesis h

Specific-to-General Algorithm (2)

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- initialization: h=$< \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset >$
- first sample is positive (EnjoySport(x_1) = true)
- However, h(x_1)=false $\rightarrow$ generalize
- h=<Sunny, Warm, Normal, Strong, Warm, Same>

Specific-to-General Algorithm (3)

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

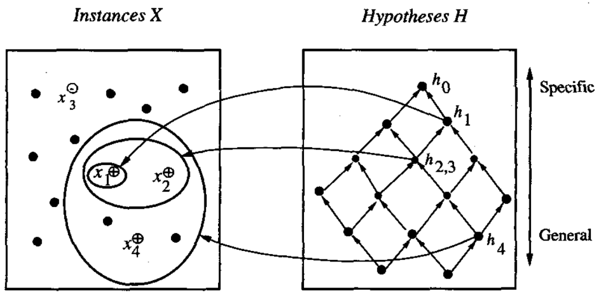
- h=<Sunny, Warm, Normal, Strong, Warm, Same>
- second sample is positive
- However, h(x_2)=false $\rightarrow$ generalize (minimally)
- h=<Sunny, Warm, ?, Strong, Warm, Same>

Specific-to-General Algorithm (4)

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- h=<Sunny, Warm, ?, Strong, Warm, Same>
- third sample is negative (EnjoySport(x_3) = false) $\rightarrow$ **ignored**
- forth sample is positive
- However, h(x_2)=false $\rightarrow$ generalize (minimally)
- **h=<Sunny, Warm, ?, Strong, ?, ?>**
- termination, return h

Specific-to-General Algorithm (5)



x_1 = <Sunny Warm Normal Strong Warm Same>, +

x_2 = <Sunny Warm High Strong Warm Same>, +

x_3 = <Rainy Cold High Strong Warm Change>, -

x_4 = <Sunny Warm High Strong Cool Change>, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

h_1 = <Sunny Warm Normal Strong Warm Same>

h_2 = <Sunny Warm ? Strong Warm Same>

h_3 = <Sunny Warm ? Strong Warm Same>

h_4 = <Sunny Warm ? Strong ? ? >

Specific-to-general Algorithm (5)

Properties of the Specific-to-General Algorithm

- algorithm is **guaranteed to output the most specific hypothesis** within H that is consistent with the positive training samples
- final hypothesis is also **consistent with negative samples** provided
  - the correct target concept is in H
  - the examples are correct

Specific-to-general Algorithm (5)

Properties of the Specific-to-General Algorithm

- algorithm is **guaranteed to output the most specific hypothesis** within H that is consistent with the positive training samples
- final hypothesis is also **consistent with negative samples** provided
  - the correct target concept is in H
  - the examples are correct

BUT:

- is the found *h* the only one consistent with the data?
- why prefer the most specific $h_{specific}$? Unclear whether we should prefer it over the most general $h_{general}$.
- data-inefficient (negative samples ignored) ... and more

Version Space and Candidate Elimination Algorithm

Candidate Elimination algorithm

The Candidate Elimination algorithm finds all describable hypotheses that are consistent with the observed training examples.

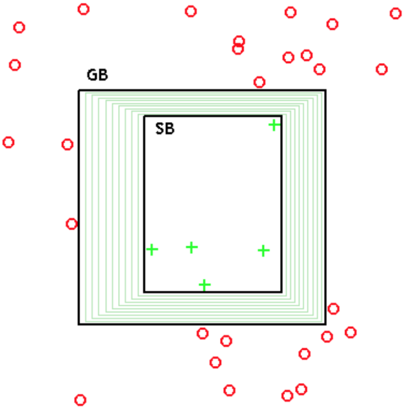
$\rightarrow$ represents the set of *all* hypotheses consistent with the observed data. This set is called *Version Space*

Version Space?

The version space, denoted $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H *consistent* with the training examples in D.

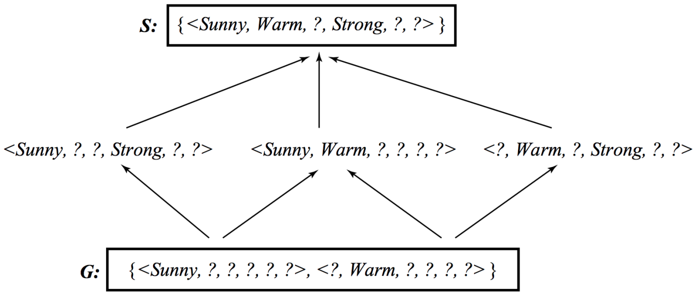
[Russell and Norvig, 2003, Mitchell, 1997]

Example of the Version Space (2)



GB = maximally **general** positive boundary, SB = maximally **specific** positive boundary
[Wikipedia: The Free Encyclopedia, 2006]

Example of the Version Space (2)



The version space for the *EnjoySports* concept learning task with the previously shown samples. The VS includes all six hypotheses. However, it can simply represented by G and S.

Candidate Elimination Algorithm

Preliminary:

- S $= \{s \mid s$ being a hypothesis consistent with the observed samples $\wedge$ there exists no hypothesis which is more **specific** than s that is also consistent with all samples$\}$
- initialize S with $< \emptyset >$
- G $= \{g \mid g$ being a hypothesis consistent with the observed samples $\wedge$ there exists no hypothesis which is more **general** than g that is also consistent with all samples$\}$
- initialize G with $<? >$

Candidate Elimination Algorithm (2)

x is a negative sample:

- remove from S any hypothesis inconsistent with x
- for each hypothesis g in G inconsistent with x:
  - remove g from G
  - add to G all minimal specializations h s.t. h is consistent with x, and some member of S is more specific than h
- remove from G any hypothesis less general than another hypothesis in G

Candidate Elimination Algorithm (2)

x is a negative sample:

- **remove from *S*** any hypothesis inconsistent with x
- for each hypothesis *g* in *G* inconsistent with x:
  - **remove *g* from *G***
  - **add to *G*** all minimal **specializations** *h* of *S* s.t. *h* is consistent with x, and some member of *S* is more **specific** than *h*
- **remove from *G*** any hypothesis less general than another hypothesis in *G*

x is a positive sample:

- **remove from *G*** any hypothesis inconsistent with x
- for each hypothesis *s* in *S* inconsistent with x:
  - **remove *s* from *S***
  - **add to *S*** all minimal **generalizations** *h* of *S* s.t. *h* is consistent with x, and some member of G is more **general** than *h*
- **remove from *S*** any hypothesis less general than other hypothesis in *S*

Candidate Elimination Algorithm Example

i=1

- S0 = {<∅,∅,∅,∅,∅,∅>}
- G0 = {<?,?,?,?,?,?>}

<∅,∅,∅,∅,∅,∅>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

i=2

- x1= <Sunny, Warm, Normal, Strong, Warm, Same>, c(x1)=true
- G consistent with x1
- S too specific → generalize until consistent

<∅,∅,∅,∅,∅,∅>

<Sunny, Warm, Normal, Strong, Warm, Same>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

i=3

- x2= <Sunny, Warm, High, Strong, Warm, Same>, c(x2)=true
- G consistent with x2
- S too specific → generalize until consistent

<∅,∅,∅,∅,∅,∅>

<Sunny, Warm, Normal, Strong, Warm, Same>

<Sunny, Warm, ?, Strong, Warm, Same>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

i=4

- x3= <Rainy, Cold, High, Strong, Warm, Change>, c(x3)=false
- S consistent with x3

<∅,∅,∅,∅,∅,∅>

<Sunny, Warm, Normal, Strong, Warm, Same>

<Sunny, Warm, ?, Strong, Warm, Same>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

i=4

- x3= <Rainy, Cold, High, Strong, Warm, Change>, c(x3)=false
- S consistent with x3 (no hypotheses in S classify x3 positive)
- G too general→ search minimal specializations, sort out inconsistencies, at least one s∈S must be more specific

<∅,∅,∅,∅,∅,∅>

<Sunny, Warm, Normal, Strong, Warm, Same>

<Sunny, Warm, ?, Strong, Warm, Same>

inconsistent

not more general

<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,Normal,?,?,?>,
<?,?,?,Weak,?,?>, <?,?,?,?,Cool,?>, <?,?,?,?,?,Same>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

i=5

- x4= <Sunny, Warm, High, Strong, Cool, Change>, c(x4)=true
- One hypothesis inconsistent in G with x4
- S inconsistent with x4 → generalize until consistent

<∅,∅,∅,∅,∅,∅>

<Sunny, Warm, Normal, Strong, Warm, Same>

<Sunny, Warm, ?, Strong, Warm, Same>

<Sunny, Warm, ?, Strong, ?, ?>

<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>

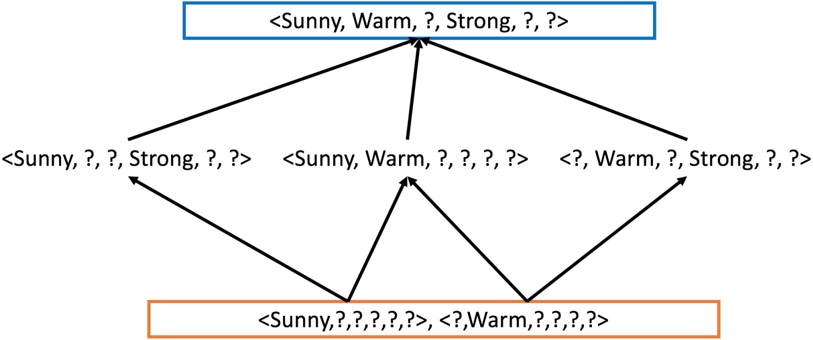
<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,?,?,?,Same>

<?,?,?,?,?,?>

Candidate Elimination Algorithm Example

Resulting Version Space

- all consistent hypotheses



Candidate Elimination Algorithm Summary

Summary:

- the version space learned by the algorithm converges toward the hypothesis correctly describing the target concept given:
  - there are no errors in the training samples
  - there is some hypothesis in H that correctly describes the target concept

Candidate Elimination Algorithm Summary(2)

Advantages:

- instances don't have to be stored (lazy-learning)
- convergence is determinable (S=G)
- data-efficient

Disadvantages:

- consistent samples required
- noisy data problematic
- target concept must be represented by hypothesis space

In semi-supervised learning: optimal query strategy is to request instances that satisfy half the hypotheses in the current VS

Table of Contents

Inductive Learning

What has been shown is associated with *inductive learning*

Induction

Plausible conclusion to the general given an input (specific).

Deduction

Logical reasoning from given knowledge (e.g. rules).

Inductive Learning Hypothesis

Inductive learning makes one **fundamental assumption:**

Induction Learning Hypothesis

Any hypothesis found to approximate a target function well over a sufficiently large set of training examples will also approximate a target function well over unknown samples.

$\Rightarrow$ **The success of learning an inductive learning machine is heavily dependent on the provided data**
$\Rightarrow$ **it can at best guarantee that the output hypothesis fits the target concept over the training data.**

Inductive Learning Hypothesis

- problem:
  target concept might not be contained in the hypothesis space
- solution(?):
  use hypothesis space that includes all possible hypotheses

Inductive Learning Hypothesis

- problem:
  target concept might not be contained in the hypothesis space
- solution(?):
  use hypothesis space that includes all possible hypotheses

Fundamental Property of Inductive Inference:

An inductive learning machine that makes no a-priori assumptions about the identity of the target concept has no rational basis for classifying unseen instances.

See [Mitchell, 1997] for an example of the futility of bias-free learning.

Inductive Learning Hypothesis(2)

Inductive bias of the Candidate Elimination algorithm

The target concept is contained in the hypothesis space and the concept could be represented by a conjunction of attribute values.

Biases

General Bias

Specification after which hypotheses are constructed. For example: classification accuracy, costs for storing hypotheses, human readability etc.

Hypothesis Space Bias

What could this be?

Preference Bias

What could this be?

Biases

General Bias

Specification after which hypotheses are constructed. For example: classification accuracy, costs for storing hypotheses or human readability

Hypothesis Space Bias

An hypothesis belongs to a restricted space of hypotheses, e.g. boolean conjunctions, linear threshold functions or 3-nearest neighbor

Preference Bias

There exists an order within the hypothesis space, e.g. prefer hypotheses with fewer disjunctions or prefer smaller decision trees

Biases (2)

Adjusting for the **Hypothesis Space Bias**:

- good classification might require complex hypothesis
- might result in overfitting

Adjusting for the **Preference Bias**:

- choose an hypothesis that correctly classifies as many samples as possible
- misclassification might be taken into account

Table of Contents

Two central questions

- Recap: inductive learning hypothesis states that any hypothesis that approx. a target function well over train data will also approx it well over unknown examples
- Now: is there a way
  - **to quantify a (classification) model's test error over unseen data?**
  - **that yields the amount of data necessary to learn?**
- Vapnik-Chervonenkis (VC) theory can help us here [Vapnik, 1971]
- PAC (probably approximately correct) not addressed here

Definition

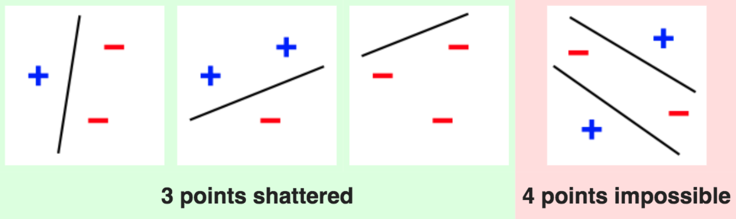
A measure of the **capacity** of the hypothesis space that can be learned by a classification model.

VC-Dimension

The cardinality of the largest set of points that a classification algorithm can *shatter*.

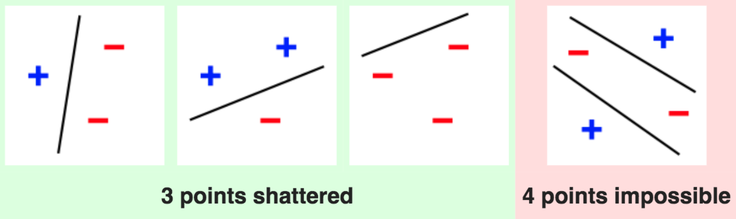
- Assume a binary data separation problem. What does *shatter* mean?
- Intuitively described:
- $\Longrightarrow$ all +/- label combinations of a fixed set of points can be separated with the classifier

Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

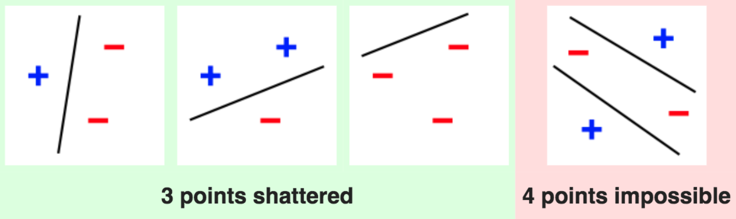
Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

- to show the VC dimension is $\geq$ a number:
  - $\exists$data points $\forall$labelings $\exists$hypothesis

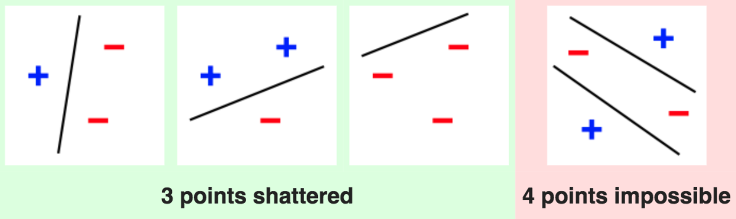
Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

- to show the VC dimension is $\geq$ a number:
  - $\exists$data points $\forall$labelings $\exists$hypothesis
- to show the VC dimension is **not** $\geq$ a number:
  - $\forall$data points $\exists$labelings $\neg(\exists$hypothesis$)$

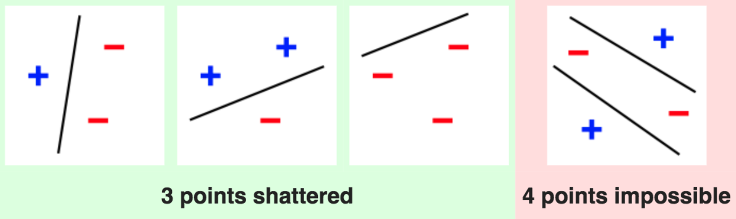
Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

- to show the VC dimension is $\geq$ a number:
  - $\exists$data points $\forall$labelings $\exists$hypothesis
- to show the VC dimension is **not** $\geq$ a number:
  - $\forall$data points $\exists$labelings $\neg(\exists$hypothesis$)$
- showing the lower bound is easier than showing higher bound

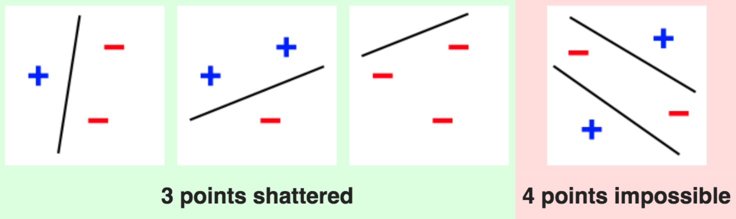
Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

- to show the VC dimension is $\geq$ a number:
  - $\exists$data points $\forall$labelings $\exists$hypothesis
- to show the VC dimension is **not** $\geq$ a number:
  - $\forall$data points $\exists$labelings $\neg(\exists$hypothesis$)$
- showing the lower bound is easier than showing higher bound
- Question: which arrangement of 3 points cannot be shattered in a binary classification problem?

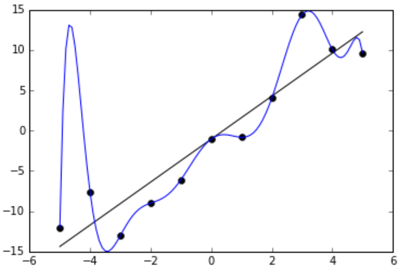
Example: VC dimension 3



from [Wikipedia: The Free Encyclopedia, 2012]

- to show the VC dimension is $\geq$ a number:
  - $\exists$data points $\forall$labelings $\exists$hypothesis
- to show the VC dimension is **not** $\geq$ a number:
  - $\forall$data points $\exists$labelings $\neg(\exists$hypothesis$)$
- showing the lower bound is easier than showing higher bound
- Question: which arrangement of 3 points cannot be shattered in a binary classification problem?
  $\Longrightarrow$ 3 collinear points

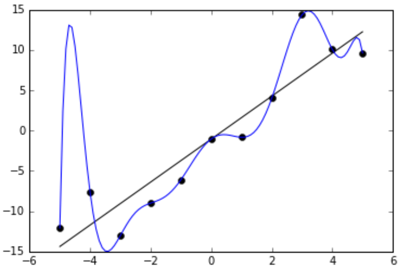
Implications



from [Encyclopedia, 2016]

- VC(H) high capacity
- VC(H) low capacity
- assumption: the higher VC(H) the better the classification model

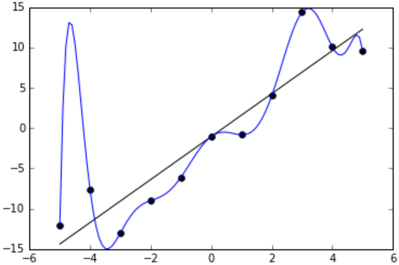
Implications



from [Encyclopedia, 2016]

- VC(H) high capacity
- VC(H) low capacity
- assumption: the higher VC(H) the better the classification model

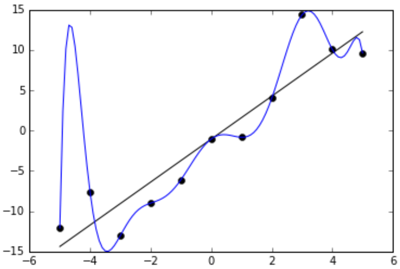
Implications



from [Encyclopedia, 2016]

- VC(H) high capacity
- VC(H) low capacity
- assumption: the higher VC(H) the better the classification model

Implications



from [Encyclopedia, 2016]

- VC(H) high capacity
- VC(H) low capacity
- assumption: the higher VC(H) the better the classification model
- $\Longrightarrow$ wrong

Implications

the VC dimension can predict a probabilistic **upper bound** on the test error:

$$Pr\left(testerror \leq trainingerror + \sqrt{\ldots\frac{VC(H)}{N}\ldots}\right) = 1 - \delta$$

while N being the size of the training set, $VC(H)$ the VC dimension of the family of hypotheses H and $0 \leq \delta \leq 1$. What are the implications?

Implications

$$Pr\left(testerror \leq trainingerror + \sqrt{...\frac{VC(H)}{N}...}\right) = 1 - \delta$$

Without going much into the details, it follows:

- as more data is added the lower the error of the model will become

Implications

$$Pr\left(testerror \leq trainingerror + \sqrt{\ldots \frac{VC(H)}{N} \ldots} \right) = 1 - \delta$$

Without going much into the details, it follows:

- as more data is added the lower the error of the model will become
- the more complex the model is the more data is required to generalize well

Implications

$$Pr\left(testerror \leq trainingerror + \sqrt{\ldots\frac{VC(H)}{N}\ldots}\right) = 1 - \delta$$

Without going much into the details, it follows:

- as more data is added the lower the error of the model will become
- the more complex the model is the more data is required to generalize well
- amount of data is linear to the VC dimension $\implies$ this is one reason why deep neural nets require so much data

Implications

$$Pr\left(testerror \leq trainingerror + \sqrt{\ldots\frac{VC(H)}{N}\ldots}\right) = 1 - \delta$$

Without going much into the details, it follows:

- as more data is added the lower the error of the model will become
- the more complex the model is the more data is required to generalize well
- amount of data is linear to the VC dimension $\implies$ this is one reason why deep neural nets require so much data

main takeaway: if you use a complex algorithm you will require a lot of data to generalize well

Curse of Dimensionality

Problems that face the *curse of dimensionality* in inductive learning typically refer to the phenomenon that, when increasing the model dimensionality[3], the available data becomes sparse
$\Rightarrow$ the demand for additional data often grows exponentially with the dimensionality

[3] e.g. attributes of concepts <x1, x2...>, weights/learned parameters in parameterized models

In Practice: Structural Risk Minimization

- computing the VC dimension is often intractable or impossible
- e.g. when need to prove that it's impossible to find any set that can be shattered

In Practice: Structural Risk Minimization

- computing the VC dimension is often intractable or impossible
- e.g. when need to prove that it's impossible to find any set that can be shattered

In Practice: Structural Risk Minimization

- computing the VC dimension is often intractable or impossible
- e.g. when need to prove that it's impossible to find any set that can be shattered

In Practice: Structural Risk Minimization

- computing the VC dimension is often intractable or impossible
- e.g. when need to prove that it's impossible to find any set that can be shattered

Structural Risk Minimization (SRM)

1. choose a class of functions (e.g. polynomials of degree n)
2. divide the class of functions into subsets: $H_1 \subset H_2 \subset ... \subset H_n$ (e.g. polynomials of increasing degree)
3. perform parameter selection (optimize)
4. select the model with lowest (empirical) error (and implicitly lowest VC dim)

In Practice: Structural Risk Minimization

- computing the VC dimension is often intractable or impossible
- e.g. when need to prove that it's impossible to find any set that can be shattered

Structural Risk Minimization (SRM)

1. choose a class of functions (e.g. polynomials of degree n)
2. divide the class of functions into subsets: $H_1 \subset H_2 \subset \ldots \subset H_n$ (e.g. polynomials of increasing degree)
3. perform parameter selection (optimize)
4. select the model with lowest (empirical) error (and implicitly lowest VC dim)

for example SVM performs SRM

Encyclopedia, W. T. F. (2016).
Overfitting.

Mitchell, T. M. (1997).
Machine Learning.
McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Russell, S. J. and Norvig, P. (2003).
Artificial Intelligence: A Modern Approach.
Pearson Education, 2 edition.

Vapnik, V.N., C. A. Y. (1971).
On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities.
Theory of Probability & Its Applications, 16:264.

Wikipedia: The Free Encyclopedia (2006).
Version Space Learning.

Wikipedia: The Free Encyclopedia (2012).

VC dimension.