

Algorithmic Information Theory

Consider the following Strings:

```
01010101010101010101010101010101010101010101010101010101010101010101010101010101010101
```

```
1100100001100001110111101110110011111010010000100101011110010110
```

How much information is in each of the strings?

Intuitively, very little in the first one because we can describe/compress the string as $32 * 01$.

It seems that we can only describe the second string by its own representation, that is, the only way to describe it is by repeating its sequence of zeros and ones. Therefore, the second string seems to contain more information.

We define the quantity of information contained in an object to be the size of that object's smallest representation or description.

So,

```
|32 * 01| < |1100100001100001110111101110110011111010010000100101011110010110|
```

Minimal Description Lengths

Observation: We can use algorithms to describe the structure of strings.

We say that $\langle M, w \rangle$ is a description of some string x if running the machine M with w as input results in string x , formally

$$x = \langle M, w \rangle$$

Note: If we assume that we encode all our descriptions and strings as binary sequences and if w is a binary sequence, then

$$\langle M, w \rangle = \langle M \rangle w$$

Example: A trivial description. Consider the machine I ,

$I =$ "On input w , where w is a binary string:

1. *accept.*"

That is, $x = \langle I, x \rangle = \langle I \rangle x$.

Minimal Description Lengths

The next theorem states that repeating a string does not significantly increase the information content of the overall string.

Theorem:

$$\forall x \exists q [K(xx) \leq K(x) + q].$$

Proof: Let $d(x) = \langle N, w \rangle$, that is $\langle N, w \rangle$ is a minimal lengths description of x . Now, consider the machine M ,

$M =$ "On input $\langle T, s \rangle$, where T is a TM and s is a string:

1. Run T on s until it halts and produces a string z .
2. Output string zz ."

Then $xx = \langle M \rangle \langle N, w \rangle = \langle M \rangle d(x)$. It follows,

$$K(xx) \leq |xx| = |\langle M \rangle d(x)| = |d(x)| + |M| = K(x) + q,$$

with $|M| = q$. \square



Incompressible Strings

Definition: Let x be a string. We say that x is *compressible* if

$$K(x) < |x|.$$

and we say that x is *incompressible* if

$$K(x) = |x|.$$

Incompressible Strings

Before moving on we need to prove the following lemma:

Lemma: For all $n \geq 1$,

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

Proof: By induction on n . The base case for $n = 1$ is easily shown to hold: $2^0 = 2^1 - 1 = 1$. For the inductive step we assume that the above holds for any n and we show that that implies that the above equation also holds for $n + 1$,

$$\begin{aligned} \sum_{i=0}^{(n+1)-1} 2^i &= \sum_{i=0}^n 2^i \\ &= \sum_{i=0}^{n-1} 2^i + 2^n \\ &= 2^n - 1 + 2^n \\ &= 2 \times 2^n - 1 \\ &= 2^{n+1} - 1 \end{aligned}$$

□



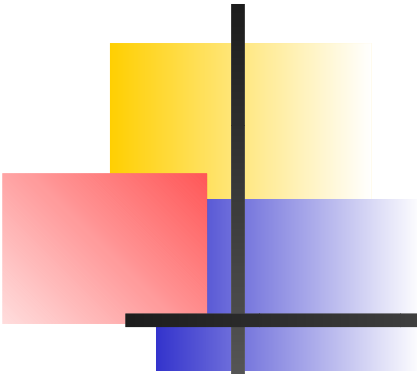
Incompressible Strings

Theorem: Incompressible strings of every length exist.

Proof: We only consider binary strings here because any alphabet can be encoded in binary. Let n be any length, it follows that there exist 2^n binary strings of length n . However, there only exist

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

descriptions of lengths less than n . Given that each description can at most describe a single string it follows that there exists at least one string that is incompressible. \square



$K(x)$ is not Computable

Theorem: $K(x)$ is not computable.

Proof: Proof by contradiction. Assume that $K(x)$ is computable; also assume the binary alphabet. We can then construct a machine M that given a length n will construct all possible binary strings and apply $K(x)$ to each string. It will halt and return the first incompressible string s it finds.

Observe that the machine M will always halt with an incompressible string s on its output tape (see the previous theorem). Observe also that for $n \gg |\langle M, n \rangle|$ the description $\langle M, n \rangle$ could serve as a description for s with,

$$|\langle M, n \rangle| < |s| = n$$

This is a contradiction, since s is incompressible by construction.

Therefore, our assumption that $K(x)$ is computable must be wrong. \square



Alg. Info. Theory

Even though $K(x)$ is not computable, approximations to this function exist and are very useful. Consider

- data compression
- machine learning



Data Compression

Data compression is accomplished with a pair of functions,

- E for encoding,
- D for decoding.

Then, applying the encoding function to some string s gives,

$$E(s) \mapsto d'(s).$$

Where $d'(s)$ is an *approximation to the minimal description* and is called the *compressed archive*.

If we let,

$$K(x) \approx |d'(x)|,$$

then we can use the encoding function to compute relative complexity measures between strings.

To obtain the original string from a compressed archive we apply the decoding function,

$$D(d'(s)) \mapsto s.$$



Machine Learning

An interesting application of minimal description lengths is in machine learning. In some sense this says that learning can be viewed as data compression. ^a

^aThis discussion is based on a tutorial by Peter Grünwald, <http://www.cwi.nl/~pdg/ftp/mdlintro.pdf>



Machine Learning

A more quantitative definition of machine learning:

Given

- A data universe X .
- A sample set S where $S \subset X$.
- Some target function (labeling process) $f : X \rightarrow \{true, false\}$.
- A labeled training set D , where $D = \{(x, y) \mid x \in S \text{ and } y = f(x)\}$.

Compute a function $\hat{f} : X \rightarrow \{true, false\}$ using D such that,

$$\hat{f}(x) \cong f(x),$$

for all $x \in X$.

This definition of machine learning is referred to as *supervised learning* due to the fact that the algorithm needs a labeled dataset D .

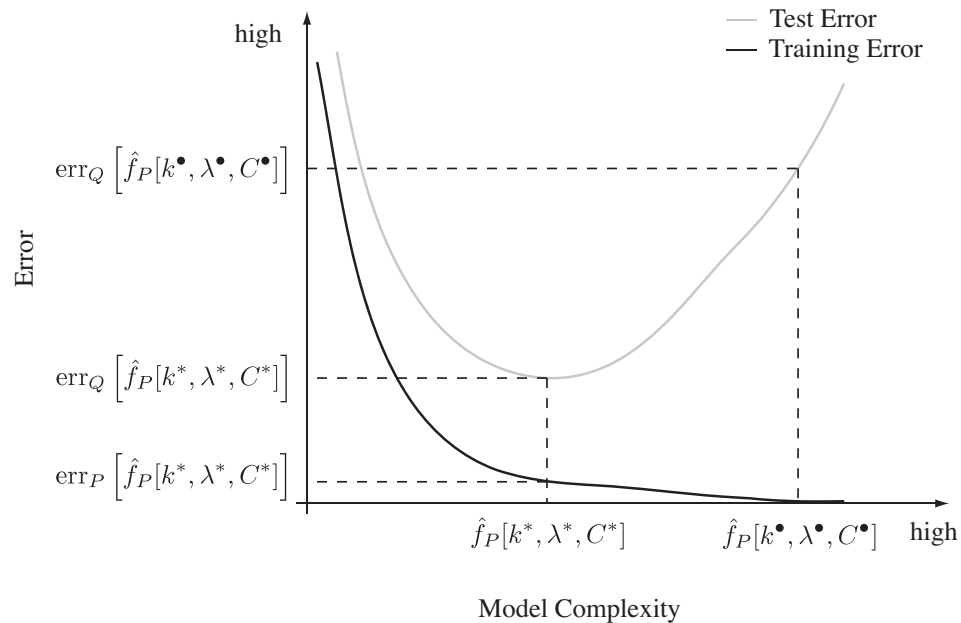
Observation: We can view the function \hat{f} as a *model* or *approximation* of the original function f . The model is computed only based on the observations in the training dataset D .

Machine Learning

One way to compute \hat{f} is to detect regularities in the training set D and describe these regularities as patterns.

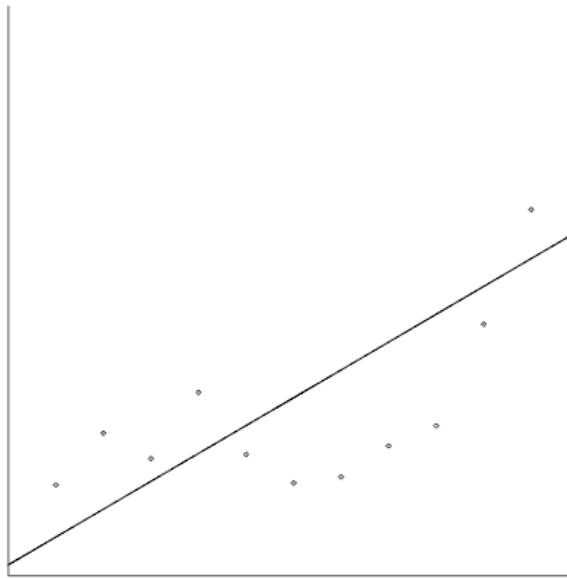
Now, it can be shown that there is a trade-off between learning all the regularities in the training set and model error. If we learn the training set too detailed then we obtain an overfitted model that does not generalize well.

Another way of saying these complex models do not generalize well.



Machine Learning

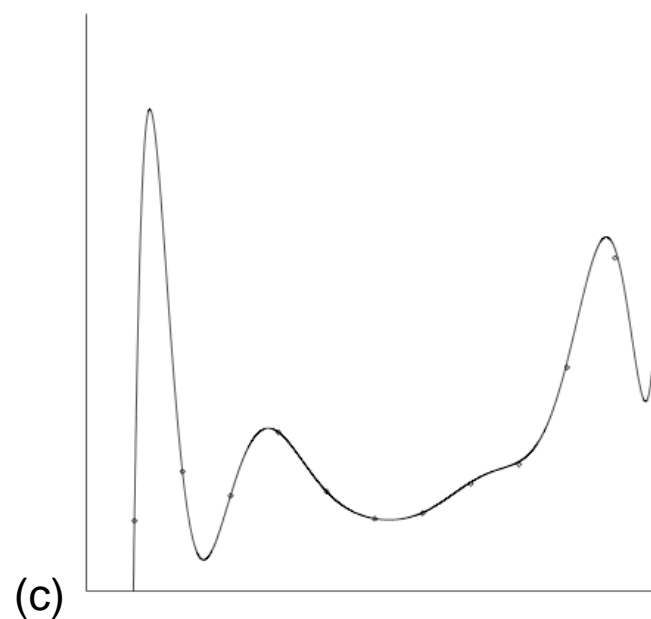
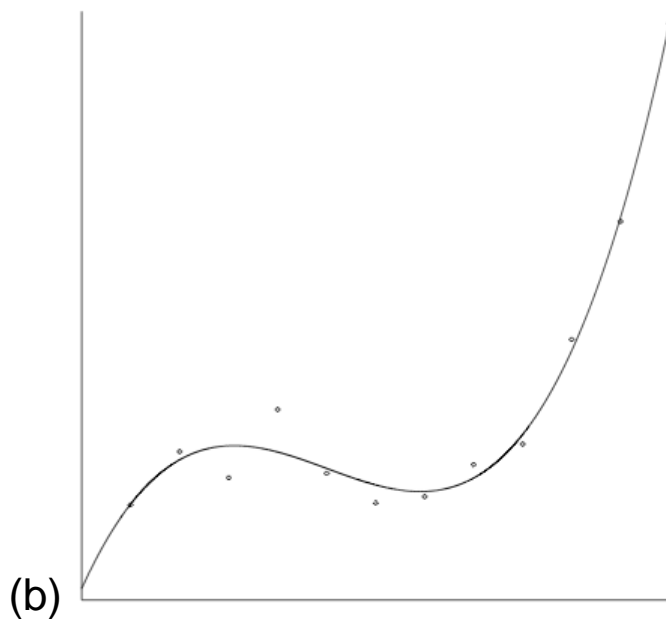
Consider the following, we have a simple two dimensional data set, X , and we want describe the relation between the two variables. A linear model is the simplest and obvious choice, but it will most likely commit many errors.



(a)

Machine Learning

We could try higher order polynomials to see if they fit the data better.



Here (b) commits some errors and (c) commits no errors.

We would consider model (c) more *complex* than (b).



Machine Learning

Observation: Assuming that X is only a sample from a larger population, there is a tradeoff between a model's ability of fitting the *training set* X and being able to generalize to points that are not in the training set. This phenomenon is called *overfitting*.

To avoid overfitting we want to tradeoff model complexity and error rate on the training set X . This process is usually referred to as *model selection*.

⇒ One way to address this tradeoff is by using maximal description lengths.



Machine Learning

Consider the following. We can represent the data using models as follows,

$$|d(X)| = |d(m)| + |d(\varepsilon)|$$

where m is a model and ε is an error term that describes on which elements of X the model made a mistake.

Let m_0, m_1, m_2 be the linear, medium powered, and high powered models we saw in Figures (a), (b), and (c), respectively, then we have,

$$|d(m_0)| < |d(m_1)| < |d(m_2)|.$$

That is, it is easier to describe linear models than it is to describe high powered polynomial models.

We also saw that m_0 committed the most errors and m_2 committed none, so,

$$|d(\varepsilon_2)| < |d(\varepsilon_1)| < |d(\varepsilon_0)|$$

So, using m_0 to encode the data, we have a very simple model, i.e., a small value for $|d(m_0)|$ but the model commits many errors, so $|d(\varepsilon_0)|$ is going to be large. On the other hand, if we use m_2 to encode the data, then we have a complex model, that is, a large value for $|d(m_2)|$ but a very small value for the error term $|d(\varepsilon_2)|$.



Machine Learning

This lets us state this as an optimization problem where we prefer a model that gives the minimal description length for X ,

$$\min |d(X)| = \min_m (|d(m)| + |d(\varepsilon)|).$$

It has been observed that models selected by minimal description lengths do not overfit and have good generalization behavior. Or we can paraphrase this as

A good approximation of the data is a reasonable model together with its exceptions.



Machine Learning

Example: Given a data set X , which of the following models is the most appropriate model for representing X in the minimal length description:

model	$ d(\text{model}) $	$ d(\varepsilon) $
m_0	5	25
m_1	10	10
m_2	20	5

Here we chose model m_1 , since $|d(X)| = |d(m_1)| + |d(\varepsilon_1)|$ is the smallest representation (minimal length description) of the dataset X . This represents an appropriate tradeoff between overfitting of the data and model complexity.