Kolmogorov Complexity, revisited
On Minimum Description Length, Inductive Inference and Machine Learning

Jesus Rodriguez Perez

Universiteit van Amsterdam

December 16, 2014
Outline

The problem of the 'priors'

Minimum Description Length

Kolmogorov Complexity

Solomonoff’s Inference and Machine Learning

Conclusions
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d = 1, 3, 5, 7$.

What number will the computer display in the next iteration? and in the $n$-th iteration?

An hypothesis for the data $d$ is $h_1$: $d_n = 2^n - 1$

Another hypothesis is $h_2$: $d_n = 2^n - 1 + (n-1)(n-2)(n-3)(n-4)$

Which one of $h_1$ and $h_2$ "seems" more probable given the data?.

Solution: Pick the hypothesis with highest posterior probability

But how to take a decision with no information other than $\sum h_i h_i = 1$?
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. 

An hypothesis for the data $d$ is $h_1$: $d_n := 2^n - 1$.

Another hypothesis is $h_2$: $d_n := 2^n - 1 + (n-1)(n-2)(n-3)(n-4)$.

Which one of $h_1$ and $h_2$ "seems" more probable given the data?

Solution: Pick the hypothesis with highest posterior probability.
The problem of the ’priors’

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. What number will the computer display in the next iteration?
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. What number will the computer display in the next iteration? and in the $n$-th iteration?
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. What number will the computer display in the next iteration? and in the $n$-th iteration?

- An hypothesis for the data $d$ is $h_1$: $d_n := 2n - 1$
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. What number will the computer display in the next iteration? and in the $n$-th iteration?

- An hypothesis for the data $d$ is $h_1$: $d_n := 2n - 1$
- Another hypothesis is $h_2$: $d_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4)$
Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence $d := 1, 3, 5, 7$. What number will the computer display in the next iteration? and in the $n$-th iteration?

An hypothesis for the data $d$ is $h_1$: $d_n := 2n - 1$

Another hypothesis is $h_2$: $d_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4)$

Which one of $h_1$ and $h_2$ "seems" more probable given the data?
The problem of the 'priors'

Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence \( d := 1, 3, 5, 7 \). What number will the computer display in the next iteration and in the \( n \)-th iteration?

- An hypothesis for the data \( d \) is \( h_1: \ d_n := 2n - 1 \)
- Another hypothesis is \( h_2: \ d_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \)
- Which one of \( h_1 \) and \( h_2 \) "seems" more probable given the data?
- Solution: Pick the hypothesis with highest posterior probability
Consider a computer program looping and printing a number at each iteration. Imagine you pause at a given time the program execution and displayed is the sequence \( d := 1, 3, 5, 7 \). What number will the computer display in the next iteration? and in the \( n \)-th iteration?

- An hypothesis for the data \( d \) is \( h_1: d_n := 2n - 1 \)
- Another hypothesis is \( h_2: d_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \)
- Which one of \( h_1 \) and \( h_2 \) ”seems” more probable given the data?
- Solution: Pick the hypothesis with highest posterior probability
- But how to take a decision with no information other than \( \sum_{h_i} = 1 \)?
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.

- "If more than one theory is consistent with the data, keep them all" - Epicurus of Samos (ca. 342 - 270 BC)

- "Among competing hypotheses, the simplest one should be selected" - William of Ockham (ca. 1287 – 1347)

- By Occam's Razor, the "simplest" hypothesis is most probable. But how to define "simple"?
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.
  
  "If more than one theory is consistent with the data, keep them all". - Epicurus of Samos (ca. 342 - 270 BC)
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.

  - "If more than one theory is consistent with the data, keep them all". - Epicurus of Samos (ca. 342 - 270 BC)

  - "Among competing hypotheses, the simplest one should be selected". - William of Ockham (ca. 1287 – 1347)
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.
  - "If more than one theory is consistent with the data, keep them all".- Epicurus of Samos (ca. 342 - 270 BC)
  - "Among competing hypotheses, the simplest one should be selected".- William of Ockham (ca. 1287 – 1347)
  - By Occam’s Razor, the ”simplest” hypothesis is most probable.
The problem of the 'priors'

- Multiple heuristics proposed over the centuries.

  - "If more than one theory is consistent with the data, keep them all".- Epicurus of Samos (ca. 342 - 270 BC)

  - "Among competing hypotheses, the simplest one should be selected".- William of Ockham (ca. 1287 – 1347)

- By Occam’s Razor, the "simplest" hypothesis is most probable. But how to define "simple"?
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C = i_1 i_2 ... i_n$.

We may call $C$ an "hypothesis" for $d$.

Some $C$ might be shorter than the data, so compress the data.

Now consider the following principle: "the best hypothesis for a given set of data is the one that leads to the best compression of the data". Named Minimum description length principle (due to Jorma Rissanen).

But this is an instance of Occam's Razor, in which we define "simplest" as "shortest".
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C=i_1 \ i_2 \ ...i_n$. 

Minimum Description Length
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C=i_1 \ i_2 \ ... \ i_n$.

We may call $C$ an "hypothesis" for $d$. 

Minimum Description Length
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C= i_1 \; i_2 \; ... \; i_n$.

We may call $C$ an "hypothesis" for $d$.

- Some $C$ might be shorter than the data, so compress the data.
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C=i_1 \ i_2 \ \ldots \ i_n$.

We may call $C$ an "hypothesis" for $d$.

- Some $C$ might be shorter than the data, so compress the data.

Now consider the following principle: "the best hypothesis for a given set of data is the one that leads to the best compression of the data"
Consider an encoding of $d$ as a sequence of computer instructions generating $d$, $C=i_1 \ i_2 \ ... \ i_n$.

We may call C an "hypothesis" for $d$.

- Some C might be shorter than the data, so compress the data.

Now consider the following principle: "the best hypothesis for a given set of data is the one that leads to the best compression of the data"

- Named Minimum description length principle (due to Jorma Rissanen).
Consider an encoding of \( d \) as a sequence of computer instructions generating \( d \), \( C=i_1 \ i_2 \ \ldots \ i_n \).

We may call \( C \) an "hypothesis" for \( d \).

Some \( C \) might be shorter than the data, so compress the data.

Now consider the following principle: "the best hypothesis for a given set of data is the one that leads to the best compression of the data"

Named Minimum description length principle (due to Jorma Rissanen).

But this is an instance of Occam’s Razor, in which we define "simplest" as "shortest"
Minimum Description Length

Now we can use it for the initial example:

Recall: \( x := 1, 3, 5, 7 \)

\( h_1 := x_n := 2^n - 1 \),

\( h_2 := x_n := 2^n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \).

We want to pick \( \hat{h} = \max \{ h_1, h_2 \} \).

Since \( p_2 \) is longer than \( p_1 \) (encodes 9 more arithmetic operations) we have \( l(p_1) < l(p_2) \) and therefore \( P(h_1) > P(h_2) \).

Both \( h_1 \) and \( h_2 \) are equally consistent with the data, so \( P(H(d|h_1)) = P(H(d|h_2)) \).

Since \( P(D(d)) \) is constant we prefer \( h_1 \).
Minimum Description Length

- Now we can use it for the initial example:

Recall: \( x := 1, 3, 5, 7 \)

\( h_1 \): \( x^n := 2^n - 1 \), \( h_2 \): \( x^n := 2^n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \).

We want to pick \( \hat{h} = \max(h_1, h_2) \) \( \{ P_D | H(d| h_1) P_H(h_1) P_D(d), P_D | H(d| h_2) P_H(h_2) P_D(d) \} \)

Since \( p_2 \) is longer than \( p_1 \) (encodes 9 more arithmetic operations) we have \( l(p_1) < l(p_2) \) and therefore \( P(h_1) > P(h_2) \).

Both \( h_1 \) and \( h_2 \) are equally consistent with the data, so \( P_H(d|h_1) = P_H(d|h_2) \).

Since \( P_D(d) \) is constant we prefer \( h_1 \).
Now we can use it for the initial example:

Recall: $x := 1, 3, 5, 7$

$h_1$: $x_n := 2n - 1$, $h_2$: $x_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4)$. 
Now we can use it for the initial example:

Recall: \( x := 1, 3, 5, 7 \)

\( h_1: x_n := 2n - 1 \), \( h_2: x_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \).

We want to pick \( \hat{h} = \max_{h_1, h_2} \left\{ \frac{P_{D|H}(d|h_1)P_H(h_1)}{P_D(d)}, \frac{P_{D|H}(d|h_2)P_H(h_2)}{P_D(d)} \right\} \)
Now we can use it for the initial example:

Recall: \( x := 1, 3, 5, 7 \)

\( h_1: x_n = 2n - 1 \), \( h_2: x_n = 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \).

We want to pick \( \hat{h} = \max \{ \frac{P_{D|H}(d|h_1)P_H(h_1)}{P_D(d)}, \frac{P_{D|H}(d|h_2)P_H(h_2)}{P_D(d)} \} \)

since \( p_2 \) is longer than \( p_1 \) (encodes 9 more arithmetic operations) we have \( l(p_1) < l(p_2) \) and therefore \( P(h_1) > P(h_2) \).
Minimum Description Length

Now we can use it for the initial example:

Recall: $x := 1, 3, 5, 7$

$h_1: x_n := 2n - 1$, $h_2: x_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4)$.

We want to pick $\hat{h} = \max_{h_1, h_2} \left\{ \frac{P_D|_H(d|h_1)P_H(h_1)}{P_D(d)}, \frac{P_D|_H(d|h_2)P_H(h_2)}{P_D(d)} \right\}$

since $p_2$ is longer than $p_1$ (encodes 9 more arithmetic operations) we have $l(p_1) < l(p_2)$ and therefore $P(h_1) > P(h_2)$.

Both $h_1$ and $h_2$ are equally consistent with the data, so $P_H(d|h_1) = P_H(d|h_2)$.

Since $P_D(d)$ is constant we prefer $h_1$. 

Now we can use it for the initial example:

Recall: \( x:= 1, 3, 5, 7 \)

\( h_1: x_n := 2n - 1 \)
\( h_2: x_n := 2n - 1 + (n - 1)(n - 2)(n - 3)(n - 4) \).

We want to pick \( \hat{h} = \max_{h_1, h_2} \left\{ \frac{P_{D|H}(d|h_1)P_H(h_1)}{P_D(d)}, \frac{P_{D|H}(d|h_2)P_H(h_2)}{P_D(d)} \right\} \)

since \( p_2 \) is longer than \( p_1 \) (encodes 9 more arithmetic operations) we have \( l(p_1) < l(p_2) \) and therefore \( P(h_1) > P(h_2) \).

Both \( h_1 \) and \( h_2 \) are equally consistent with the data, so \( P_H(d|h_1) = P_H(d|h_2) \).

Since \( P_D(d) \) is constant we prefer \( h_1 \).
Kolmogorov Complexity

The length of the shortest code for $d$ is the Kolmogorov Complexity of $d$.

But the Kolmogorov complexity of any data depends on the language of the program that generates it, so cannot be "universal"...
Kolmogorov Complexity

- The length of the shortest code for $d$ is the *Kolmogorov Complexity* of $d$.
Kolmogorov Complexity

- The length of the shortest code for $d$ is the *Kolmogorov Complexity of $d$*

- But the Kolmogorov complexity of any data depends on the language of the program that generates it, so cannot be "universal"...
Kolmogorov Complexity

- The length of the shortest code for $d$ is the *Kolmogorov Complexity* of $d$.

- But the Kolmogorov complexity of any data depends on the language of the program that generates it, so cannot be "universal"...
Kolmogorov Complexity

▶ Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).

▶ Proof:
Consider \( K_p(o) = \min_{p: U(p) = o} \) \( l(p) \) and \( K_{p'}(o) = \min_{p': U(p') = o} \) \( l(p') \) for arbitrary \( p, p' \) with \( |K_p(o) - K_{p'}(o)| = d \geq 1 \).

By definition \( K_p(o), K_{p'}(o) < l(o) \), so that \( m < l(o) \).
Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).
Kolmogorov Complexity

- Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).
- Proof:
Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).

**Proof:** Consider $K_p(o) = \min_{p: U(p) = o} \{l(p)\}$ and $K_{p'}(o) = \min_{p': U(p') = o} \{l(p')\}$ for arbitrary $p, p'$ with $|K_p(o) - K_{p'}(o)| = d \geq 1$. 
Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).

Proof: Consider $K_p(o) = \min_{p:U(p)=o} \{l(p)\}$ and $K_{p'}(o) = \min_{p':U(p')=o} \{l(p')\}$ for arbitrary $p, p'$ with $|K_p(o) - K_{p'}(o)| = d \geq 1$

$$d \geq 1 \Rightarrow |\frac{K_p(o)}{m} - \frac{K_{p'}(o)}{m}| < 1 \text{ for } m = \max(K_p(o), K_{p'}(o)).$$
Good news is that the error we can commit is bounded by the length of the data itself (Invariance Theorem).

**Proof:** Consider $K_p(o) = \min_{p: U(p) = o} \{l(p)\}$ and $K_{p'}(o) = \min_{p': U(p') = o} \{l(p')\}$ for arbitrary $p, p'$ with $|K_p(o) - K_{p'}(o)| = d \geq 1$

\[ d \geq 1 \Rightarrow \left| \frac{K_p(o)}{m} - \frac{K_{p'}(o)}{m} \right| < 1 \text{ for } m = \max(K_p(o), K_{p'}(o)).\]

By definition $K_p(o), K_{p'}(o)) < l(o)$ so that $m < l(o)$
Solomonoff’s inference and Machine Learning

Kolmogorov Complexity is central in Ray Solomonoff’s Inductive Inference Theory. The theory formalizes the sequence prediction procedure we did at the beginning of the talk. But sequence prediction is quite a small subset of real-world prediction problems... Nevertheless, some Machine Learning problems can be reduced to it.
Solomonoff’s inference and Machine Learning

- Kolmogorov Complexity is central in Ray Solomonoff’s Inductive Inference Theory.
Kolmogorov Complexity is central in Ray Solomonoff’s Inductive Inference Theory.

The theory formalizes the sequence prediction procedure we did at the beginning of the talk.
Solomonoff’s inference and Machine Learning

- Kolmogorov Complexity is central in Ray Solomonoff’s Inductive Inference Theory.
- The theory formalizes the sequence prediction procedure we did at the beginning of the talk.
- But sequence prediction is quite a small subset of real-world prediction problems...
Solomonoff’s inference and Machine Learning

- Kolmogorov Complexity is central in Ray Solomonoff’s Inductive Inference Theory.
- The theory formalizes the sequence prediction procedure we did at the beginning of the talk.
- But sequence prediction is quite a small subset of real-world prediction problems...
- Nevertheless, some Machine Learning problems can be reduced to it.
Discrete regression is a good example: Training dataset (feature,value): 

\((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).

Goal: Find \(f\) 

Equivalent to find \(f(i) = f(x_{n+1})\) given 

\(x_1 f(x_1), x_2 f(x_2), \ldots, x_n f(x_n), x_{n+1}\).

Recall first example: Find next in 1,1,2,3,3,5,4,7,5

But way to prefer such weird “representation”?

Suppose some scattered outliers in a dataset.

Traditional ML techniques typically risk fitting the regression function too much (high \(P(h—d)\)) with complex models (low \(P(h)\) as estimated by Occam’s Razor).

In contrast, MDL principle tends to gain a balance with \(P(d|h) \approx P(h)\), therefore maximizing \(P(d|h) P(h)\).
Solomonoff’s inference and Machine Learning

- Discrete regression is a good example:
Discrete regression is a good example:

- Training dataset (feature, value): 
  \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
Solomonoff’s inference and Machine Learning

- Discrete regression is a good example:
  - Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
  - Goal: Find \(f\)
Discrete regression is a good example:
- Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
- Goal: Find \(f\)
- Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1) x_2 f(x_2) x_n f(x_n) x_{n+1}\)
Discrete regression is a good example:

- Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
- Goal: Find \(f\)
- Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1) x_2 f(x_2) x_n f(x_n) x_{n+1}\)
  - Recall first example: Find next in 1, 1, 2, 3, 3, 5, 4, 7, 5

But way to prefer such weird “representation”?

Suppose some scattered outliers in a dataset.

Traditional ML techniques typically risk fitting the regression function too much (high \(P(h|d)\)) with complex models (low \(P(h)\) as estimated by Occam’s Razor).

In contrast, MDL principle tends to gain a balance with \(P(d|h) \approx P(h)\), therefore maximizing \(P(d|h)\)P(h).
Discrete regression is a good example:

- Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
- Goal: Find \(f\)
- Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1) x_2 f(x_2) x_n f(x_n) x_{n+1}\)
  - Recall first example: Find next in 1, 1, 2, 3, 3, 5, 4, 7, 5

But way to prefer such weird ”representation”?
Discrete regression is a good example:

- Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\).
- Goal: Find \(f\)
- Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1)x_2 f(x_2)x_n f(x_n)x_{n+1}\)
  - Recall first example: Find next in 1, 1, 2, 3, 3, 5, 4, 7, 5

But way to prefer such weird ”representation”?
- Suppose some scattered outliers in a dataset.
Discrete regression is a good example:
- Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots (x_n, f(x_n))\).
- Goal: Find \(f\)
- Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1) x_2 f(x_2) x_n f(x_n) x_{n+1}\)
  - Recall first example: Find next in 1, 1, 2, 3, 3, 5, 4, 7, 5

But way to prefer such weird ”representation”?
- Suppose some scattered outliers in a dataset.
- Traditional ML techniques typically risk fitting the regression function too much (high \(P(h—d)\)) with complex models (low \(P(h)\) as estimated by Occam’s Razor).
Solomonoff’s inference and Machine Learning

- Discrete regression is a good example:
  - Training dataset (feature, value): \((x_1, f(x_1)), (x_2, f(x_2)), \ldots (x_n, f(x_n))\).
  - Goal: Find \(f\)
  - Equivalent to find \(f(i) = f(x_{n+1})\) given \(x_1 f(x_1) x_2 f(x_2) x_n f(x_n) x_{n+1}\)
    - Recall first example: Find next in 1,1,2,3,3,5,4,7,5

- But way to prefer such weird ”representation”?
  - Suppose some scattered outliers in a dataset.
  - Traditional ML techniques typically risk fitting the regression function too much (high \(P(h—d)\)) with complex models (low \(P(h)\) as estimated by Occam’s Razor).
  - In contrast, MDL principle tends to gain a balance with \(P(d|h) \approx P(h)\), therefore maximizing \(P(d|h)P(h)\).
Conclusions

Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.

MDL pple performs better than typical ML models in very noisy data.

Unlike entropy, kolmogorov complexity cannot be computed in general. However Kolmogorov Complexity is "approximately" computable. We can avoid infinite loops at the cost of an approximated solution:

\[ \hat{K}(o) := \min_{p : U(p) = o} \{ l(p) + \log t \} \]
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
- MDL pple performs better than typical ML models in very noisy data.
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
- MDL pple performs better than typical ML models in very noisy data.
- Unlike entropy, kolmogorov complexity cannot be computed in general.
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
- MDL pple performs better than typical ML models in very noisy data.
- Unlike entropy, kolmogorov complexity cannot be computed in general.
  - However Kolmogorov Complexity is "approximately" computable.
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
- MDL pple performs better than typical ML models in very noisy data.
- Unlike entropy, kolmogorov complexity cannot be computed in general.
  - However Kolmogorov Complexity is ”approximately” computable. We can avoid infinite loops at the cost of an approximated solution:
Conclusions

- Sometimes little training data is available, and MDL pple assumes nothing about the distributions it tries to compress, so has a general advantage to Maximum Entropy models, which assume a uniform distribution of a restricted set of parameters.
- MDL pple performs better than typical ML models in very noisy data.
- Unlike entropy, kolmogorov complexity cannot be computed in general.
  - However Kolmogorov Complexity is "approximately" computable. We can avoid infinite loops at the cost of an approximated solution:
    - e.g. Time-bounded "Levin" complexity:
      \[ \hat{K}(o) := \min_{p: U(p) = o \text{ in } t \text{ steps}} \{ l(p) + \log t \} \]
Conclusions

Entropy increases.
Complexity first increases, then decreases.