Information Theory in Machine Learning

Jaspreet Singh

08 June, 2020

Contents

1 Abstract

Machine Learning deals with systems that can learn from data. The resulting flow and storage of information in this process can be represented mathematically. Information Theory is a branch of applied mathematics that helps us answer some key questions like: What should the model learn? How do we evaluate the model? How does the model learn? What should we adjust for a better performing model? There are various metrics and measures that have helped us create and optimize machine learning models over the years. This paper covers all these metrics and their applications. It also attempts to explain and implement entropy based modifications in clustering, classification and regression techniques.

2 Introduction

Information can be represented as bits. One bit of information allows us to choose between two equally probable, or equiprobable, alternatives. In other words, in a scenario with 2 equiprobable choices, if you get an instruction to choose one of the choices, that is one bit of information. Say, each instruction can be represented as a binary digit (0='choice 1' and 1='choice 2') then this binary digit provides you with one bit of information. In the case of 'n' sequential forks, with 'm' final choices (destinations), then $m= 2^n$. In other words, $n= \log_2 m$. Information is thus, an ordered symbol of sequences to interpret its meaning.

Please note: A binary digit is the value of a binary variable, whereas a bit is an amount of information. A binary digit (when averaged over both of its possible states) can convey between zero and one bit of information.

By now, we know what information means. It is a mathematical representation of uncertainty. Let us now understand what information theory is.

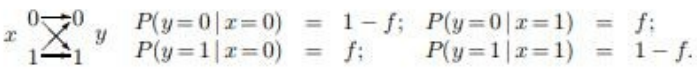


Figure 2.1: A binary symmetric channel showing probability of incorrect transmission of message given the transmitted symbol is x and the received symbol is y

When we transmit data, for instance, a string of bits, over a communication channel or any computational medium, there is a probability (say 'p') that the received message will not be identical to the transmitted message. An ideal channel is where this probability (p) is zero (or almost 0). However, most real world channels have a non zero probability 'f' of incorrect transmission of information and a probability of (1 minus f) that each bit of information will be transmitted correctly. Given this probability of error, one needs to find solutions to reduce the same in order to transmit information with minimum error. One can use the 'physical solution' where one needs to revamp the physical attributes of the communication channel, for instance the circuitry. However, this might increase the operational cost. An alternative solution is to update the system using 'information theory' and 'coding theory'. Under these solutions, we accept the given noisy physical channel in its current form. We then add communication systems to it so that we can detect and correct the errors introduced by the channel. This system normally comprises of an encoder and decoder. The 'system' solutions can help you design a reliable communication channel with the

only additional cost of computations of an encoder and a decoder. While coding theory helps you design the appropriate decoder and encoder, information theory helps you define the quality of information or content you can transmit. It can help you study the theoretical limitations and potentials of a system.

Information theory treats information as a physical entity, like energy or mass. It deals with theoretical analyses of how information can be transmitted over any channel: natural or man-made. Thus, it defines a few laws of information. Let us assume a basic system for information flow as follows:

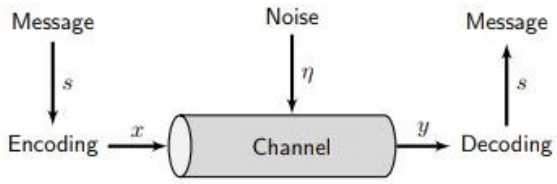


Figure 2.2: Information Channel: A message (data) is encoded before being fed into a communication channel, which adds noise. The channel output has the encoded message with noise that is decoded by a receiver to recover the message.

There are a few laws on information that can be derived from the above channel:

- There is a definite upper limit, the channel capacity, to the amount of information that can be communicated through that channel.
- This limit shrinks as the amount of noise in the channel increases.
- This limit can very nearly be reached by judicious packaging, or encoding, of data.

Essentially, information theory entails two broad techniques: 1. Data Compression (source coding): More frequent events should have shorter encodings 2. Error Correction (channel coding): Should be able to infer encoded event even if message is corrupted by noise

Both these methods require you to build probabilistic models of the data sources. This is why information theory is relevant to machine learning and data analytics. It not only helps you measure the accuracy of information contained in a data source, but also helps you improve results of predictive models that might be built on this data. Before we study how information theory can be applied to, let us study a few basic terms.

3 Few important terms of Information Theory and Machine Learning

3.1 Marginal Probability

The probability of an event occurring is also called marginal probability. It may be thought of as an unconditional probability. It is not conditioned on another event. It can be represented as *P* (*x*) for an event ‘x’.

3.2 Conditional Probability

Probability Theory defines conditional probability as the measure of the probability of an event (some particular situation occurring) given that another event has occurred.
If we have two variables ‘x’ and ‘y’, the conditional probability that event ‘x’ occurred, given ‘y’ has already occurred is represented as: $P(x \mid y)$

3.3 Joint Probability

Joint Probability is defined as the probability of two events occurring together. If ‘x’ and ‘y’ are two events, the likelihood of both the events occurring can be represented as $A \cap B$. This values can be defined as

$$P(x \cap y) = P(x \mid y) * P(y)$$

3.4 Codewords

Information theory represents data in the form of codewords. These codewords are representation of the actual data elements in the form of sequence of binary digits or bits. There are various techniques to map each symbol (data element) with the corresponding codeword.

3.5 Variable Length Codes

One traditional way of assigning codewords to data elements would be to assign codes with fixed lengths, or every data element gets assigned a code of the same code length. Let us look at a visual representation of the same. If we have 4 values of a given variable, say ‘Discount type’ with the following 4 values: ‘Promotion’, ‘Priority Customer’, ‘Repeat buy’ and ‘None’. If we assume that the 4 types of discounts are equally probable, they can safely be mapped to codewords of equal length, say 00, 01, 10 and 11, as shown in the image below:

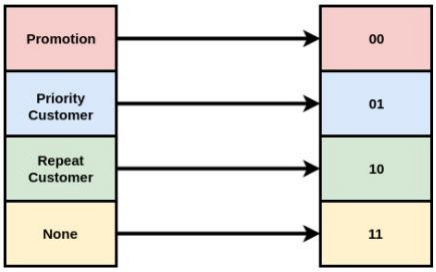


Figure 3.1: Fixed Length Encoding: Every data element here is assumed to have equal likelihood of occurrence. Hence, every code word have equal and fixed length (i.e. 2)

Now, if we know that discounts under ‘Promotion’ are more frequent than the rest and have been assigned the following probability values.
We can then, design codes weighted according to these probability values. We design the code to reduce the total length of the message. Since one would transmit the most frequent terms

Promotion	1/2
Priority Customer	1/4
Repeat Customer	1/8
None	1/8

Figure 3.2: Varying probabilities of each data element indicating the likelihood of occurrence of the event

(data elements) most number of times, you would want to associate the least length with the most frequent term. Hence, we design a code with the least number of bits for the most frequent data element. Let us look at an example of the same.

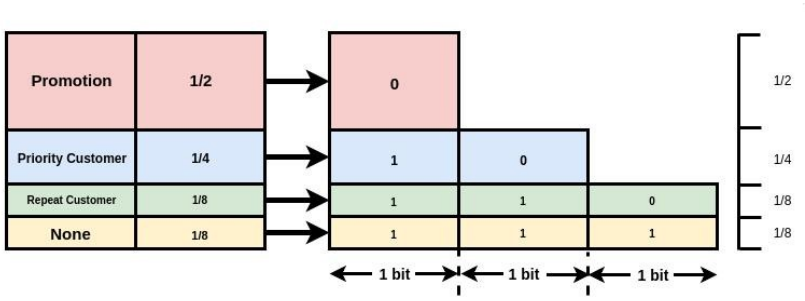


Figure 3.3: Variable Length Encoding: Every data element has been mapped to a sequence of binary codes of varying length according to the probability of occurrence

This variable-length code will now represent each data element in a unique and exclusive manner. Moreover, if you consider the vertical axis to visualize the probability of each word, p(x), and the horizontal axis to visualize the length of the corresponding codeword, L(x), let us compute the area covered by the encoding. This amounts to 1.75 bits, which is a reduction from a fixed-length encoding of 2 bits each for all the above terms. The fixed-length encoding would amount to 2 bits.

3.6 Optimal Encoding

The most optimal encoding has to strike a balance between the length of the message and the cost of the codeword. Let us look at how one affects the other. The cost of buying a codeword of length 0 is 1. This is because you are buying all possible codewords, and hence, if you want to have a codeword of length 0, you can't have any other codeword. The cost of a codeword of length 1, like "0" or "1", is (1/2) because half of possible codewords start with "0" or "1" respectively. The cost of a codeword of length 2, like "01", is (1/4) because a quarter of all possible codewords start with "01". **In general, the cost of codewords decreases exponentially with the length of the codeword.**

Refer to the image below:

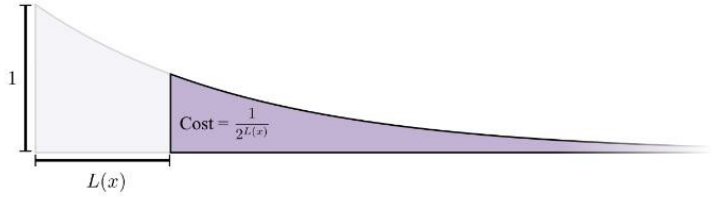


Figure 3.4: Cost of a codeword as a function of the length of the codeword

We know that the average length of the total code string is the function of the probability of each word and the length of the corresponding codeword. It can be represented as $p(x) * L(x)$. This average length contribution is related to the cost through the length of the codeword. The amount we pay decides the length of the codeword. The length of the codeword controls how much it adds to the average message length. We can picture the two of these together in the following way:

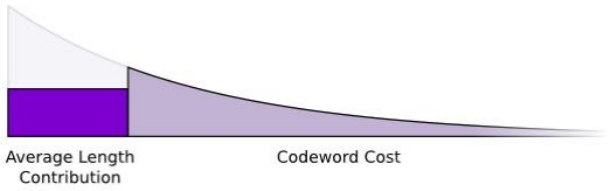


Figure 3.5: Relationship between Average Length Contribution and Cost of the Codeword

Thus, we can clearly establish that shorter codewords cost higher and vice versa.

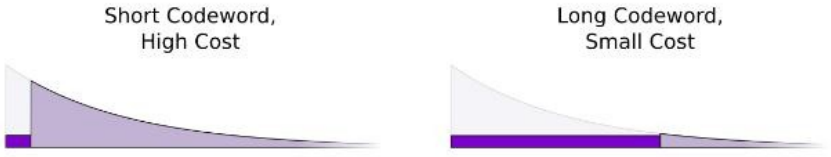


Figure 3.6: Costs of long and short codewords

3.7 Shannon Information Content

Let us consider a set of discrete random variables ‘X’ which contains a possible set of events $\{a_i\}$ where ‘i’ represents the range of elements contained in the set ‘X’. Shannon Information Content or Information Content is the measure of information contained in the event: $X = a_i$. Information content actually quantifies the surprisal of each outcome of any experiment. Higher the surprisal of an outcome, higher is the information content.

Thus, the information you can gain when an event occurs which had some probability value asso-ciated with it, can be represented as:

$$h(X = a_i) = log_2(1/p(X = a_i)) \quad (1)$$

The above equation will show that as the value of probability ‘p’ increases, the information content decreases. You get a curve that looks something like this:

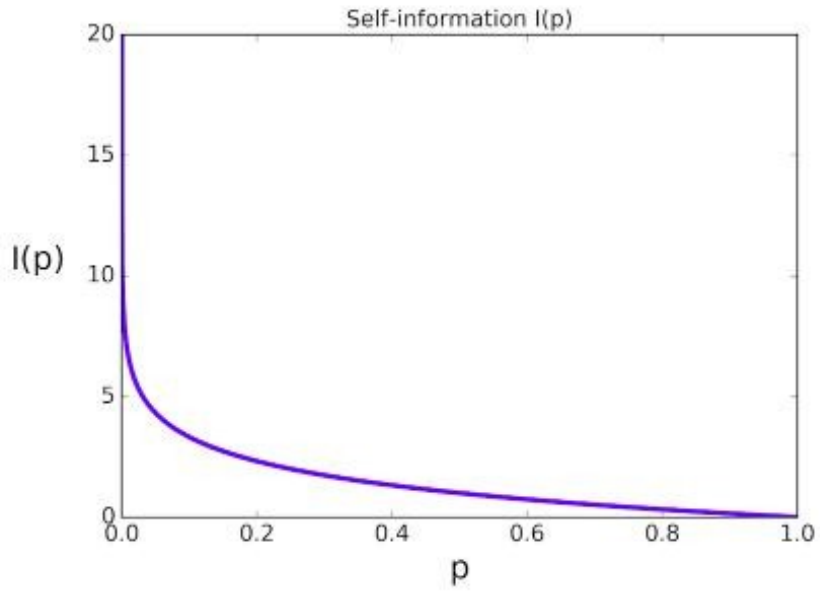


Figure 3.7: The information content function versus probability values

The information content is measured in bits. It is also known as *self information*.

3.8 Entropy

While designing an optimal codeword for our data elements above, we see that the average length contribution of a message and the cost of the same line up well. In other words, the height of the rectangle denoting the average length contribution of messages is almost equal to the maximum height of the exponential curve denting the cost of the codeword. Thus, if we slightly increase the length of the codeword, the message length contribution will increase in proportion to its height at the boundary, while the cost will decrease in proportion to its height at the boundary.

Figure 3.8: Proportionality of cost with the height of the message length contribution

We can also say that the cost to make the codeword for a data element ‘a’ shorter is p(a). Now, let us generalize this to any message ‘X’. For a message ‘X’ with the likelihood of occurrence $p(X)$, the cost can be approximated as $p(X)$ too. We also know that the cost of a message of length ‘L’ is $1/2^L$. If we invert this to obtain the length of a message that costs a given amount (cost), we get: $L = log(1/cost)$. If we substitute the value of cost by $p(X)$, we derive ‘L’ as $log(1/p(X))$. Thus, the length of a message ‘X’ can be denoted in terms of its probability in the following equation:

$$L(X) = log_2(1/p(X))$$

As we discussed earlier, there is a limit to how short an average message can get to communicate events effectively, from a particular probability distribution 'p' of events 'X'. This limit, the average message length using the best possible code, is called the entropy or Shannon entropy of 'p', $H(X)$. In other words, entropy is a measure of unpredictability of the state (X), or equivalently, of its average information content. The formula to represent entropy is as follows:

$$H(X) = \sum_{X} p(X) log_2(1/p(X)) \quad (2)$$

$H(X)$ can also be denoted by $H(p)$ where 'p' is the vector containing probabilities of each of the discrete events. Entropy is measured in bits too. Let us look at one example that shall clarify this completely.

Let us assume 'X' is a set of randomly selected letters (a_i) from the English alphabet. Each value in 'X' has a probability value $p(X = a_i)$ associated with itself. Following is a representation of the same:

```
# import dataset
alphabet <- read.csv('./data/alphabet_data.csv')

#  create column with Shannon Information Content for each event in the set X
alphabet$info_a_i_p <- log2(1/alphabet$p)

#  print the dataset
print(alphabet)
```

```
#          i a_ip info_a_i_p
  ## 1     1    a 0.0575     4.120294
  ## 2     2    b 0.0128     6.287712
  ## 3     3    c 0.0263     5.248793
  ## 4     4    d 0.0285     5.132894
  ## 5     5    e 0.0913     3.453241
  ## 6     6    f 0.0173     5.853084
  ## 7     7    g 0.0133     6.232430
  ## 8     8    h 0.0313     4.997694
  ## 9     9    i 0.0599     4.061300
   ## 10 10    j 0.0006    10.702750
   ## 11 11    k 0.0084     6.895395
   ## 12 12    l 0.0335     4.899695
   ## 13 13    m 0.0235     5.411195
   ## 14 14    n 0.0596     4.068544
   ## 15 15    o 0.0689     3.859352
   ## 16 16    p 0.0192     5.702750
   ## 17 17    q 0.0008    10.287712
```

```
## 18 18   r 0.0508      4.299028
## 19 19   s 0.0567      4.140507
## 20 20   t 0.0706      3.824188
## 21 21   u 0.0334      4.904008
## 22 22   v 0.0069      7.179188
## 23 23   w 0.0119      6.392895
## 24 24   x 0.0073      7.097888
## 25 25   y 0.0164      5.930160
## 26 26   z 0.0007     10.480357
## 27 27   - 0.1928      2.374823
```

```r
# compute Entropy of dataset
Entropy_p <- sum(alphabet$p*alphabet$info_a_i_p)

# print entropy value
cat("Entropy of the above data is: ",Entropy_p)
```

```
## Entropy of the above data is:  4.109446
```

In this dataset, 'p' is one of the probability distributions of each of the events (probability of occurrence of the alphabets). As we see above the entropy of the dataset is 4.11 approximately. This is also known as marginal entropy. The column 'info_a_i_p' is the information content corresponding to each of the elements (a_i) in 'X'.

3.9 Conditional Entropy

The average uncertainty about the output value (say 'Y') given an input value ('X') is the conditional entropy $H(Y|X)$. Here, $H(Y|X)$ is, 'the residual uncertainty (entropy) of 'Y' given that we know the value of 'X''. Let us assume a channel with noise η and input 'X' and output 'Y'. In this case, $H(Y|X)$ can also be represented as $H(\eta)$. It can be represented in the following form:

$$H(Y|X) = \sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) log_2(1/p(y|x))$$

The above equation can further be condensed into the following form:

$$H(Y|X) = \sum_{x \in X, y \in Y} p(y, x) log_2(1/p(y|x)) \quad (3)$$

You could modify this equation for a given value of the input, say $X = a_i$ in the following manner:

$$H(Y|X = a_i) = \sum_{y \in Y} p(y, x = a_i) log_2(1/p(y|x = a_i))$$

3.10 Cross Entropy or Joint Entropy

If X and Y are discrete random variables and p(X,Y) is the value of their joint probability distri-bution at (, y), then the joint entropy or cross entropy of X and Y is:

$$H(Y, X) = \sum_{x \in X} \sum_{y \in Y} p(y, x) \log_2(1/p(y, x))$$

Let's say you have two discrete distributions: p(x) and q(x) with common elements. One way of defining cross entropy would be: the average length of code for communicating an event from one distribution with the optimal code for another distribution is called the cross-entropy. It can be represented so:

$$H_p(q) = \sum_{x} q(x) \log_2(1/p(x))$$

For continuous distributions, you can simply replace the discrete summation with integral functions. Cross-entropy is an important measure. It gives us a way to show how different two probability distributions are. The more different the distributions 'p' and 'q' are, the more the cross-entropy of 'p' with respect to 'q' will be bigger than the entropy of 'p'.

3.11 Mutual Information

Let us consider a channel with inputs 'x', output 'y' and noise η.

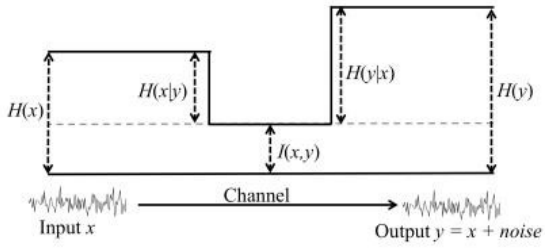


Figure 3.9: Channel design demonstrating an ideal communication channel

In the above channel $H(x)$ and $H(y)$ are the marginal entropies of the inputs and outputs respectively. $H(x|y)$ and $H(y|x)$ are the conditional entropies of the inputs given the outputs and vice versa respectively. We can now derive mutual information from this system. The mutual information $I(x, y)$ between two variables, such as a channel input 'x' and output 'y', is the average amount of information that each value of 'x' provides about 'y'. It can represented so:

$$I(x, y) = H(y) - H(y|x) = H(y) - H(\eta) = H(x) + H(y) - H(x, y)$$

In terms of 2 two random variables 'X' and 'Y' , whose joint distribution is defined by $p(X, Y)$, mutual information can be represented in the following form:

$$I(X,Y) = H(Y) - H(Y|X) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2(p(x, y)/p(x)p(y)) \quad (4)$$

Another metric that can be derived from the mutual information is the **variation of information**. The variation of information is the information which isn't shared between these two variables ('x' and 'y'). We can define it like so:

$$V(x, y) = H(x, y) - I(x, y) \tag{5}$$

The variation of information between two variables is zero if knowing the value of one tells you the value of the other and increases as they become more independent.

3.12 Conditional Mutual Information

The conditional mutual information is defined as the expected value of the mutual information of two random variables given the value of a third. For random variables X, Y, and Z, it can be represented as $I(X; Y|Z)$. Let us look at a representation of these three variables in the form of a Venn diagram:

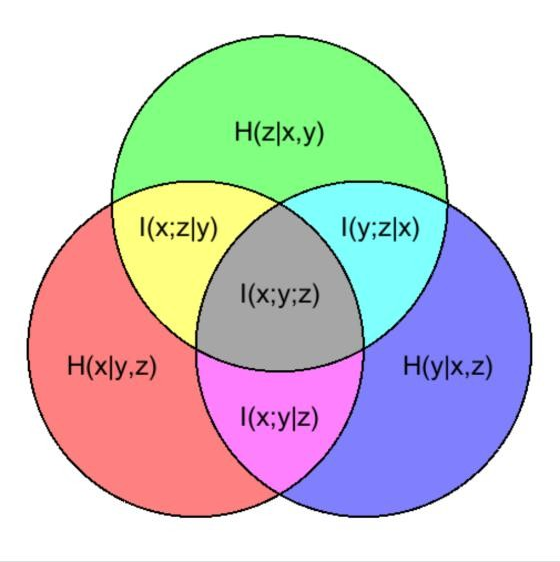


Figure 3.10: Channel design demonstrating an ideal communication channel

Mathematically, it can be defined as:

$$I(X; Y|Z) = \sum_{z \in Z} p_Z(z) \sum_{x \in X} \sum_{y \in Y} p_{X,Y|Z}(x, y|z) \log\left(p_{X,Y|Z}(x, y|z) / \left(p_{X|Z}(x|z) p_{Y|Z}(y|z)\right)\right)$$

This can be simplified to:

$$I(X; Y|Z) = p_{X,Y,Z}(x, y, z) \log\left(\left(p_Z(z) p_{X,Y,Z}(x, y, z)\right) / \left(p_{X,Z}(x, z) p_{Y,Z}(y, z)\right)\right) \tag{6}$$

The above entity is greater than '0' for discrete, jointly distributed random variables X, Y and Z. The above probability mass functions represented by 'p' can be evaluated using conventional definitions of probability.

3.13 Multivariate mutual information

Multivariate mutual information (MMI) or Multi-information is the amount of information about Z which is yielded by knowing both X and Y together is the information that is mutual to Z and the X,Y pair, written as $I(X, Y ; Z)$. In the above Venn diagram, it is the central part of the diagram. Mathematically, a general form of multi-information can be represented as a sum of entropies:

$$I(X_1; X_2; .....; X_n) = - \sum_{T \subseteq (1,2,...,n)} (-1)^{|T|} H(T)$$

Positive MMI is typical of common-cause structures. For example, clouds cause rain and also block the sun; therefore, the correlation between rain and darkness is partly accounted for by the presence of clouds, $I(rain; dark|cloud) \leq I(rain; dark)$. The result is positive MMI $I(rain; dark; cloud)$. A condition where $I(Y ; Z|X) \geq I(Y ; Z)$ indicates a negative MMI $I(X; Y ; Z)$.

3.14 Channel Capacity

The measure channel capacity basically helps us answer how fast can we transmit information over a communication channel. One of the most common and convenient channels used is the *additive channel*. This channel adds noise (η) to the encoded input values as they pass through the channel, so that the channel output is a noisy version 'y' of the channel input 'x'. It looks something like this:

$$y = x + \eta$$

The channel capacity 'C' is the maximum amount of information that a channel can provide at its output about the input. It is often represented as:

$$C = max(I(x, y)) \tag{7}$$

where $(I(x, y))$ is the mutual information of the inputs and the output.

The rate at which information is transmitted through a channel can be determined using the entropies of three variables:

1. the entropy $H(x)$ of the input,
2. the entropy $H(y)$ of the output,
3. the entropy $H(\eta)$ of the noise in the channel.

A output entropy is high then this provides a large potential for information transmission. It depends on the input entropy and the level of noise. If the noise is low, then the output entropy can be as close to the channel capacity. his further explains and reinforces equation (6).However, channel capacity gets progressively smaller as the noise increases. Capacity is usually expressed in bits-per-usage (bits per output), or bits-per-second (bits/s). The **Shannon-Hartley theorem** states that the channel capacity (C) is given by:

$$C = Blog_2(1 + S/N)$$

Here, (S/N) is the signal-to-noise ratio of the channel. 'B' is the bandwidth of the channel in Hertz and C is represented in bits-per-second.

3.15 Information Divergence

Information divergence is a measure of dissimilarity of content of two probability distributions. This is a general concept that is used across various domains to compare distributions and the information content of the same. In a lot of machine learning objectives, it can be used as an approximation of the form $x \sim \mu$, where $x > 0$ is the observed data (input) and μ is the approximation given by the model.

There is a large variety of information divergences. Most of them are distance metrics. We are covering one of them in the next section.

3.16 Relative entropy or Kullback–Leibler divergence

KL divergence stands for Kullback-Leibler divergence. If we have two distributions $p(x)$ and $q(x)$, defined for the same set of events, KL divergence helps you determine the difference between these distributions. It is closely related to relative entropy, and information divergence. It is a non-symmetric measure of the difference between two probability distributions. This means that the divergence of $q(x)$ from $p(x)$, denoted by $D_{KL}(p||q)$ or $D_{KL}(p(x), q(x))$, is a measure of the information lost when q(x) is used to approximate p(x). It implies that $D_{KL}(p(x), q(x))! = D_{KL}(q(x), p(x))$. For two discrete distributions, such that $p(x) > 0$ and $q(x) > 0$, it can be represented in the following way:

$$D_{KL}(p(x), q(x)) = D_{KL}(p||q) = H(p, q) - H(p) = \sum_{x} p(x) log(p(x)/q(x)) \quad (8)$$

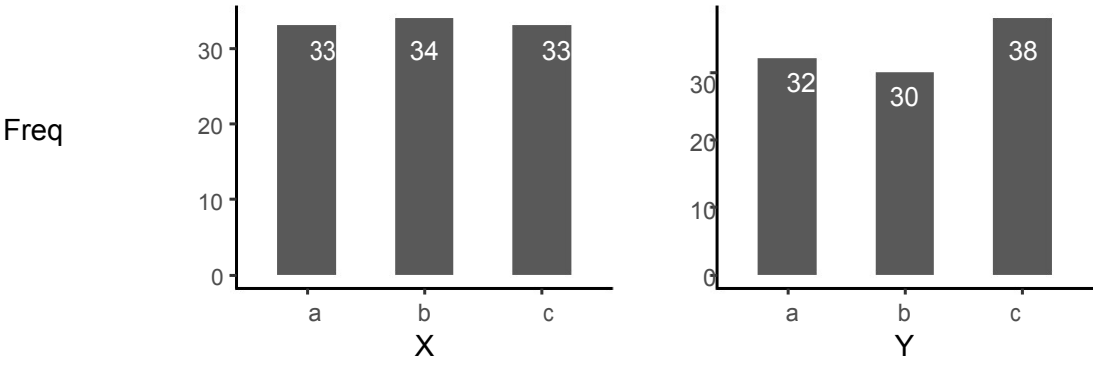
Thus, KL divergence or relative entropy measures the expected number of extra bits required to code samples from $p(x)$ when using a code based on $q(x)$, instead of using a code based on $p(x)$. Since KL divergence is a measure of dissimilarity or a distance, it is closely related to cross entropy and mutual information. If you remember from the last section, 'variation of information' quantifies the notion of distance, between different variables. However, it differs slightly from KL divergence.

KL divergence gives us a distance between two distributions over the same variable or set of vari-ables. In contrast, variation of information gives us distance between two jointly distributed vari-ables. KL divergence talks about divergence between distributions, while variation of information does the same within a distribution. *Please note that distance is not exactly the best term to be used in this context, since distance is symmetric and divergence is not. However, we can use it to draw an analogy in this case.*

Now let us look at how KL divergence can be used in machine learning. Let us start with creating two random distributions. Below is a snippet to create 2 random distributions of events named ('a','b' and 'c'):

```
#using the 'entropy' library for KL divergence
library(entropy)
library(ggplot2)
library(gridExtra)

#create 2 random lists of the alphabets('a','b' and'c') with varying distributions
```

```
X <- sample(letters[1:3], 100, 1)
Y <- sample(letters[1:3], 100, 1)

#plot both distributions to compare
#for X
ggplot(data=as.data.frame(table(X)), aes(x=X, y=Freq), color=X) +
   geom_bar(stat="identity", width=0.5)+
   geom_text(aes(label=Freq), vjust=1.6, color="white", size=3.5)+
   theme_classic() -> plot_x

#for Y
ggplot(data=as.data.frame(table(Y)), aes(x=Y, y=Freq), color=Y) +
   geom_bar(stat="identity", width=0.5)+
   geom_text(aes(label=Freq), vjust=1.6, color="white", size=3.5)+
   theme_classic() -> plot_y

grid.arrange(plot_x, plot_y, ncol = 2)
```



Let us now look at how to compute the KL divergence for these two distributions:

```
#create a contingency table showing the distribution of these alphabets
#  A contingency table is a type of table in a matrix format
#  It displays the (multivariate) frequency distribution of the variables
#  It is highly used in information theory for multivariate statistics
t    <- table(X,Y)
print(t)
```

```
#         Y
#   Xa  b  c
#      a 12  9 12
#      b 11 10 13
#      c  9 11 13
```

```
#create contingency table with fractional values tp <-
t/100
print(tp)
```

```
#         Y
## X          a       b       c
#       a 0.12 0.09 0.12
#       b 0.11 0.10 0.13
#       c 0.09 0.11 0.13
```

```
#normalize the above table
tn    <- tp/sum(tp)

#obtain distributions from the contingency table #for X

p_x <- rowSums(tn)
#for Y
p_y <- colSums(tn)

#  manually deriving KL divergence P
<- tn
Q <- p_x %o% p_y
kl_1 <- sum(P*log(P/Q))
library(entropy)

#KL divergence of p_y from p_x using the library 'entropy' kl_2<-
KL.empirical(P,Q)
#check if both values are equal
compare<- (kl_1 == kl_2)

#KL divergence of p_y from p_x using the library 'entropy' kl_3<-
KL.empirical(Q,P)

#KL divergence of p_y from itself using the library 'entropy' kl_4<-
KL.empirical(P,P)

#printing results
cat("The KL divergence of distribution p_y from p_x is: ",kl_2)
```

```
## The KL divergence of distribution p_y from p_x is:                0.003395495
```

```
ifelse(compare==TRUE,
          "Results from manual calculation and R package entropy have matched", "Results
          from manual calculation and R package entropy have not matched")
```

```
## [1] "Results from manual calculation and R package entropy have matched"
```

```
cat("The KL divergence of distribution p_x from p_y is: ",kl_3)
```

```
## The KL divergence of distribution p_x from p_y is:  0.003416819
```

```
cat("The KL divergence of distribution p_x from itself is: ",kl_4)
```

```
## The KL divergence of distribution p_x from itself is:  0
```

The above results give us the following takeaways:

- KL divergence is not symmetric. As we see above the value of KL divergence of 'p_x' from 'p_y' and vice versa differs a lot.
- KL divergence is 0 where $P = Q$. In other words, the divergence of one distribution from itself in '0'. This is because $log(P/Q) = log(1) = 0$.
- If P is not equal to Q i.e. for two distinct distributions, the KL divergence is always positive because the entropy is the minimum average loss-less encoding size.
- Higher the value of the KL divergence, higher is the information lost when a distribution (q(x)) is used to approximate another distribution (p(x)). Thus to reduce the divergence or this loss of information, we should reduce the KL divergence.

The above points state some key characteristics of KL divergence. These characteristics of KL diver-gence make it widely applicable in machine learning applications. In a lot of cases, KL divergence is used in optimization functions for problems like optimizing learning algorithms in supervised learning etc.

Because of the fact that KL divergence is not a symmetric measure (i.e. $D_{KL}(p(x), q(x))! = D_{KL}(q(x), p(x))$), we get two objective functions to optimize while approximating p(x) in a loss-less manner efficiently. These functions are:

1. Minimizing the **forward KL divergence**: $argmin_\theta D_{KL}(P \,||Q_\theta)$
2. Minimizing the **reverse KL divergence**: $argmin_\theta D_{KL}(Q_\theta||P)$

Both the above optimization functions actually cause different types of approximations. Let us look at each one of them in a little detail:

3.16.1 Forward KL

Let us start with the base equation of Forward KL:

$$argmin_\theta D_{KL}(P \,||Q_\theta) \tag{9}$$

Let us start expanding equation (7). On substituting the value of KL divergence, with its expansion from equation (6):

$$argmin_\theta D_{KL}(P \,||Q_\theta) = E_{x\sim P}\,[log(1/q(X))] + E_{x\sim P}\,[log(1/p(X))]$$

This can also be written as:

$$argmin_{\theta} D_{KL}(P \,||Q_{\theta}) = E_{x\sim P}\,[log(1/q(X))] - H(p(X)) \quad (10)$$

In the above equation, the first element is the cross entropy between P and Q (also denoted by H(p,q) while the second element $H(p(X))$ is the entropy of 'P'(also represented as $E_{x\sim P}$ $[-log(p(X))]$). Since the entropy of $p(X)$ is fixed the final equation takes the following form:

$$argmax_{\theta} D_{KL}(P \,||Q_{\theta}) = E_{x\sim P}\,[log(1/q(X))] \quad (11)$$

The above equation looks similar to the maximum likelihood estimation objective in machine learn-ing exercises. This objective will sample points from p(X) and try to maximize the probability of occurrence of these points under q(X). **mean-seeking behaviour**, because the approximate dis-tribution Q *must* cover all the modes (frequent events) and regions of high probability in P. In short, **"Wherever P has high probability, Q must also have high probability."**Let us look at an example of an approximate distribution for the same:

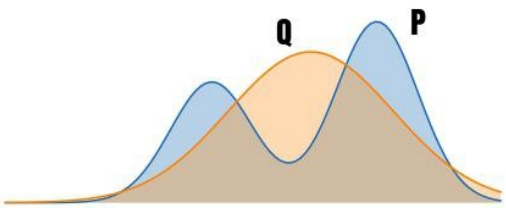


Figure 3.11: Approximate distribution for minimization of forward KL divergence

In the above diagram, the approximate distribution Q centers itself between the two modes of P, so that it can have high coverage of both. The 'forward KL divergence' optimization technique does not penalize Q for having high probability mass where P does not.

3.16.1.1 Application of forward KL divergence

Now, in supervised learning techniques employing 'empirical risk minimization', we use a dataset of samples $D = (x_i, y_i)$ from some ground-truth data distribution $P(x, y) = P(x)P(y|x)$. In supervised learning we aim to learn a model that maps data elements X to Y in the following manner: $f : X ß Y$. This model should eventually minimize the empirical risk of the model, which is determined by a parameter defining loss of information, also called *loss function"" $L(f(x), y)$. A loss function is a mapping function that associates an event to a real number or a value with some "cost". We will talk about the loss functions in detail later. We need to optimize the loss function (reduce the cost) over some distribution of models f_{θ}. This creates an optimization objective function as follows:

$$argmin_{\theta} E_{(x,y)\sim D}[L(f_{\theta}(x), y)]$$

Optimizing this objective is equivalent to minimizing the divergence from an approximate distribution(Q_θ) to the true data distribution(P). This concept can be extended to both regression and classification in the following ways:

1. Regression with Mean-Squared Error Loss: In regression, one of the significant loss functions is the mean-squared error. We aim to reduce the loss in order to get the most accurate predictions. If the distribution of your estimations can be represented as $q_\theta(y|x)$ which is normally distributed. The negative log-likelihood of this normal distribution can be defined as: $-log(q_\theta) = (-1/2)||y-f_\theta(x)||^2+C$, where $f_\theta(x)$ are the results from the regression function and y are the actuals. Minimizing the negative log-likelihood of this normal distribution is hence, equivalent to the mean-squared error loss.
2. Classification with Cross Entropy Loss: In this case, the approximate distribution (Q) is the result of the classification model. It is represented as a discrete event distribution parameterized by a probability vector (probability of an event belonging to a particular class). In classification you aim to reduce the cross entropy loss(or log loss) of the predicted results against the ground-truth.

The above logic can be applied to any loss function. This way, we can improve the accuracy of the machine learning models. This concept is widely applied to supervised learning techniques.

3.16.2 Reverse KL

Let us start with the base equation of Reverse KL:

$$argmin_\theta D_{KL}(Q_\theta||P) \tag{12}$$

The equation for reverse KL can be written as:

$$argmax_\theta D_{KL}(Q_\theta||P) = E_{x\sim Q_\theta}[log(p(X))] + H[Q_\theta]$$

The above equation will sample points from Q and try to maximize the probability of these points under P. The entropy term encourages the approximate distribution to be as wide as possible. A good approximation under the reverse KL objective thus satisfies the below condition: **“Wherever Q has high probability, P must also have high probability.”**

It is known as the **mode-seeking behaviour** because any sample from the approximate distribu-tion Q *must* lie within a mode of P (since it’s required that samples from Q are highly probable to occur under P). Let us look at an approximate distribution to visualize the same.

As we see here, the approximate distribution essentially encompasses the right mode of P. The reverse KL divergence does not penalize Q for not placing probability mass on the other mode of P.

3.16.2.1 Application of reverse KL divergence

Reverse KL divergence finds its application in reinforcement learning. The maximum-entropy reinforcement learning objective which is used in reinforcement learning models uses this principle extensively.

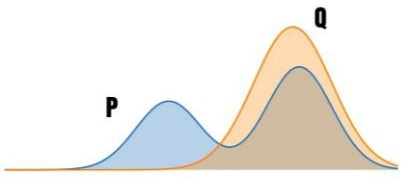
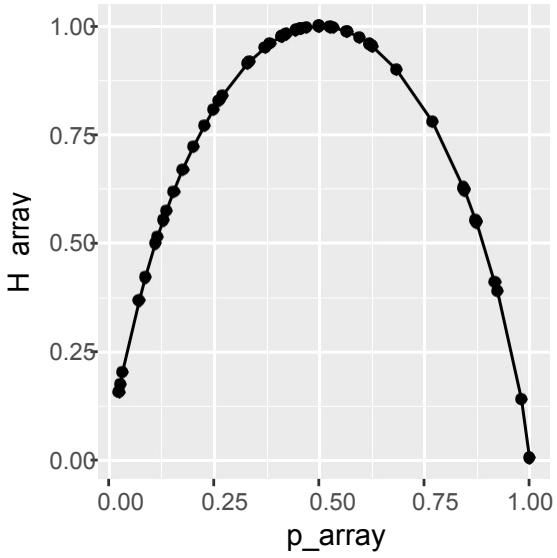


Figure 3.12: Approximate distribution for minimization of reverse KL divergence

3.17 Differential Entropy

While applying statistics to information theory, we come across a concept of maximum entropy probability distributions. Let us begin with a binomial event of flipping a coin. If a random variable 'X' represents the toss of a fair coin we have, $P(X = H) = p$ and $P(X = T) = (1 - p)$ with $p \in [0, 1]$. If we were to compute the entropy of all possible probability values, we will obtain a distribution. Let us look at the same.

```
#generate random fractional numbers in the range [0,1]
p_array<-runif(n=50, min=0, max=1)

#create an empty array for story entropy values
H_array=numeric(0)

#  compute the entropy using equation 2
for(i in 1:length(p_array)){
   H_i=(p_array[i]*(log2(1/p_array[i])))+((1-p_array[i])*(log2(1/(1-p_array[i])))) H_array[i]<- H_i

}

#create a dataframe
distr<-cbind(as.data.frame(p_array),as.data.frame(H_array))

# Plot the above distribution
ggplot(data=distr, aes(x=p_array, y=H_array)) +
   geom_line()+
   geom_point()
```



The above distribution of entropy with varying 'p' depicts a continuous analogue to discrete entropy is called **differential entropy** (or continuous entropy). There are a few differences between discrete entropy and differential entropy. It uses the integral function instead of the discrete aggregation. Moreover, differential entropy can be negative. For a random variable $X = f(x)$, the differential entropy can be defined as:

$$h(f(x)) = \int_{X} f(x) log(1/f(x)) dx$$

The above equation when applied to a uniform distribution $U(0, 1/2)$, gives the value $-log(2)$. This proves that differential entropy can be negative. Furthermore, differential entropy for normal distributions often assume the form of $[contant + log(\sigma)]$ where σ is the standard deviation of the distribution.

3.18 Entropy Distributions

When we study entropy distributions, we often aim to get a maximum entropy distribution. This is because we want to use a variable that can transmit as much information as possible. Since information is quantified the entropy, we want the variable to have maximum entropy. In other words, A distribution of values that has the maximum information (or entropy) possible, is a maximum entropy distribution. Maximum entropy distributions are important because they help us provide with the maximum amount of prior information and then build the most accurate or precise systems when applied to real-world problems. A maximum entropy probability distribution has entropy that is at least as great as that of all other members of a specified class of probability distributions.

According to the principle of maximum entropy, if nothing is known about a distribu-tion except its parent class, then the distribution with the largest entropy should be chosen as the least-informative default. For a given variable, the precise form of its maximum entropy distribution depends on the constraints placed on the values of that variable. We will cover a few distributions that depend on these constraints.

The motivation of building and applying maximum entropy distributions is twofold: first, maximizing entropy minimizes the amount of prior information built into the distribution; second, many physical systems tend to move towards maximal entropy configurations over time. Let us look at a few common maximum entropy distributions:

3.18.1 Gaussian Distribution

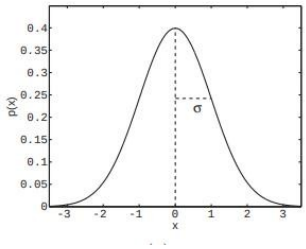


Figure 3.13: Maximum Entropy Distribution: Gaussian distribution, with mean = 0 and a standard deviation = 1

If a variable has **fixed variance** and has no other constraints, the maximum entropy distribution for this variable will be a Gaussian distribution. The key characteristic of a Gaussian distribution is that it is highly energy efficient because no other distribution can provide as much information at a lower energy cost per bit.

If a variable has a Gaussian or normal distribution then the probability of observing a particular value 'x' is $p(x)$ and is represented by the following equation:

$$p(x) = (1/\sqrt{(2\pi v_x)})e^{(\mu_x - x_2)/2v_x}$$

Here, e=2.72. Intuitively, the above probability distribution equation above will form a bell curve as normal distributions do. The term μ_x here indicates the mean of the variable 'x', and defines the central value of the distribution. The variance of the variable, v_x is the square of the standard deviation σ_x of 'x'. It defines the width of the bell curve as shown above.

3.18.2 Exponential Distribution

A variable that has no values below zero (**no negative values**), and has a **fixed mean** (μ), but has no other constraints, has an exponential maximum entropy distribution. The probability distribution can be represented by the following equation:

$$p(x) = (1/\mu)e^{-x/\mu}$$

The above has a variance of μ^2 and looks like the curve below (Figure 3.14).

3.18.3 Uniform Distribution

The uniform maximum entropy distribution represents those variables that have a fixed lower bound x_{min} and upper bound x_{max}. These variables are otherwise unconstrained. The probability distribution function is represented by:

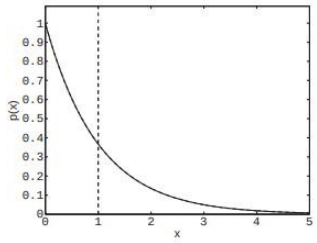


Figure 3.14: Maximum Entropy Distribution: Exponential distribution, with mean indicated by the dotted vertical line

$$p(x) = 1/(x_{max} - x_{min})$$

The curve for the same is depicted by the plot below (Figure 3.15):

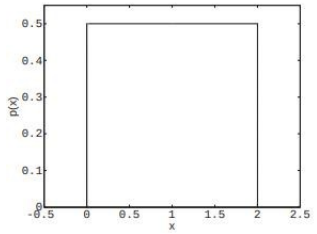


Figure 3.15: Maximum Entropy Distribution: A uniform distribution with a range between zero and two

3.19 Loss functions

Loss functions are basically just objective functions that are applied to a lot of machine learning models. The functions take from the concepts of information theory. While a lot of loss functions are designed on the basis of information content or the entropy of datasets, there are a few others that use simple mathematical operations to measure the accuracy and performance of machine learning models. Let us refer to this image below:

Let us talk about a few loss functions in detail.

3.19.1 Cross Entropy Loss and Log Loss

To understand the definition and application of this loss function, let us first understand that it is used in classification problems. This entropy-based loss function is widely used to measure the performance of classification algorithms that give the probability of a record belonging to a particular class as an output. Cross entropy is the more generic form of another loss function, called the **logarithmic loss** or log loss, when it comes to machine learning algorithms. While log loss is used for binary classification algorithms, cross-entropy serves the same purpose for multiclass classification problems. In other words, log loss is used when there are 2 possible outcomes and cross-entropy is used when there are more than 2 possible outcomes.

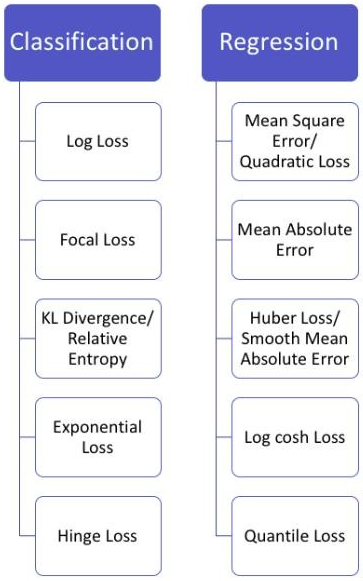


Figure 3.16: Some key loss functions in classification and regression models

These loss functions quantify the price paid for the inaccuracy of predictions in classification prob-lems by penalizing false classifications by taking into account the probability of classification. The generic equation of cross entropy loss looks like the following:

$$CrossEntropyLoss = -(1/N) \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} log_2 p_{ij} \quad (13)$$

Here, 'M' is the number of outcomes or labels that are possible for a given situation and 'N' is the number of samples or instances in the dataset. 'p_{ij}' is the model's probability of assigning label 'j' to instance 'i' and 'y_{iy}' depicts the outcome of the i-th instance with respect to the possible labels (j). Now, if you see the equations looks similar to the cross entropy equation above. Let us take an example for the same.

Let us assume a problem statement where one has to predict the range of grades a student will score in an exam given his attributes. If there are three possible outcomes: High, Medium and Low represented by [(1,0,0) (0,1,0) (0,0,1)]. Now, for a particular student, the predicted probabilities are (0.2, 0.7, 0.1). This indicates the predicted range of scores will most likely be 'Medium' as the probability is the highest there. Hence, the cross-entropy error would be:

```
CE_loss=  -(((0)*log2(0.2)) + ((1)*log2(0.7)) + ((0)*log2(0.1)))
cat("Cross entropy loss of the above prediction is: ", CE_loss)
```

```
## Cross entropy loss of the above prediction is:  0.5145732
```

To convert the equation (13) into log loss, we take into account only two possible outcomes. Thus the formula looks something like this:

$$LogLoss = -(1/N) \sum_{i=1}^{N} (y_i log_2 p_i + (1 - y_i) log_2 (1 - p_i))$$

Following the same conventions as equation (13), here y_i depicts the outcome (the class) of the i-th instance. p_i is the probability (outcome of the classifier) of the i-th instance assuming the value 'y_i'. So if y_i assumes the value '1', $(1 - y_i)$ would be equal to 0: the two classes of a binary classification problem.

We have already spoken about **KL divergence** as a loss function for both regression and classifi-cation problems. Let us look at a few other loss functions.

3.19.2 Focal Loss

Focal loss is an improvement of cross-entropy loss often used for object detection and neural net-works. In problems like object detection, a major issue is caused by class imbalance due to the presence of large number of easily-classified background examples. Focal loss is one such loss function that tries to accommodate for this class imbalance.

$$F\,L(p_t) = (1 - p_t)^{\gamma} log_2(p_t)$$

Here, p_t is defined the following way:

$$p_t = p : when(y = 1); (1 - p) : otherwise$$

This basically indicates the two possible outcomes of a binary classifier.

This loss function is a dynamically scaled log loss function, where the scaling factor $(1 - p)^{(\gamma)}$ decays to zero as confidence in the correct class increases. This scaling factor can automatically down-weight the contribution of easy examples (examples present in large numbers) during training and rapidly focus the model on hard examples (sparse examples). Focal loss enables the user to train a high-accuracy, one-stage detector that significantly outperforms the alternatives of training with hard example mining. Focal loss can be depicted used the below graph:

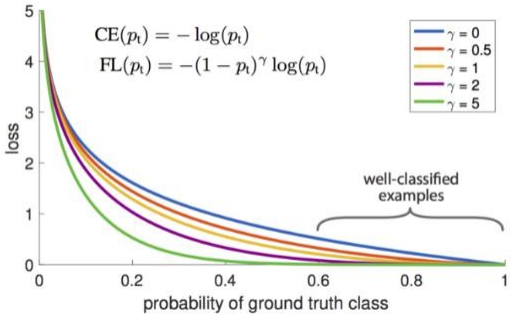


Figure 3.17: Focal loss adds a scaling factor to standard cross entropy loss

Here, we see that the focal loss function adds a factor $(1-p_t)^{\gamma}$ to the standard binary loss function. The relative loss for well-classified easy examples (having $(p_t > 0.5)$) is reduced for all $\gamma > 0$. As the value of γ increases this loss further reduces towards the tail of the plots. In the above curve,

the line corresponding to (γ = 0) represents the cross entropy (CE) loss or more specifically, the log loss function.

There are several other functions that use probability-based concepts, not necessarily based on entropy and information content. You can explore their use and application according to your requirements and the kind of model you are trying to evaluate.

3.20 Learning Rate

Learning rate, also called step-size, is a model-hyperparameter that tells us how to adjust the weights of our model network with respect to the loss gradient function. This concept comes from the optimization functions we saw previously. When applied to machine learning, learning rate helps determine how to change the parameters of a model for it to converge the best, or have the most accuracy. In other words, if we have a hypothesis or model represented in the following manner:

$$h_\theta = \theta_0 + \theta_1 x$$

We have to alter the weights (θ_0, θ_1) so that the above equation best estimates h_θ. We thus apply the gradient descent (partial derivative) operation here and find the following relationship between the old and new weights:

$$new_weight = existing_weight - learning_rate * gradient$$

More precisely, it can be represented in the following manner:

$$\theta_1 := \theta_1 - \alpha(\delta J(\theta_1)/\delta\theta_1) \quad (14)$$

In the above equation 'α' is the learning rate. This basically depicts to what extent the newly acquired information about a dataset or a model overrides old information. The aim is to arrive at an optimal learning rate. This is because an optimal learning rate will give the best performance of the model i.e. least loss of information.

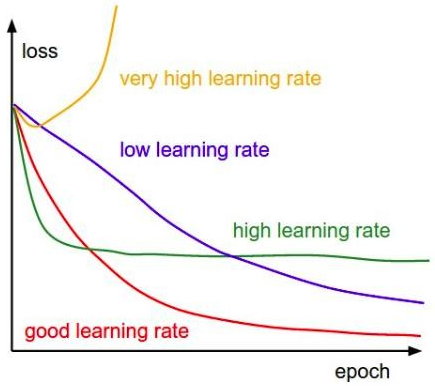


Figure 3.18: Effect of learning rate on loss function of a model

As we see above (Figure 3.18), the learning rate cannot be set either be too high or too low. This is because, with a very high learning rate, the next point in the equation (14) above will perpetually bounce haphazardly across the minima of the curve (refer to Figure 3.19 below). With the value of *α* too low, it might take too much time trying to converge the model. This will cost you computational power and time.

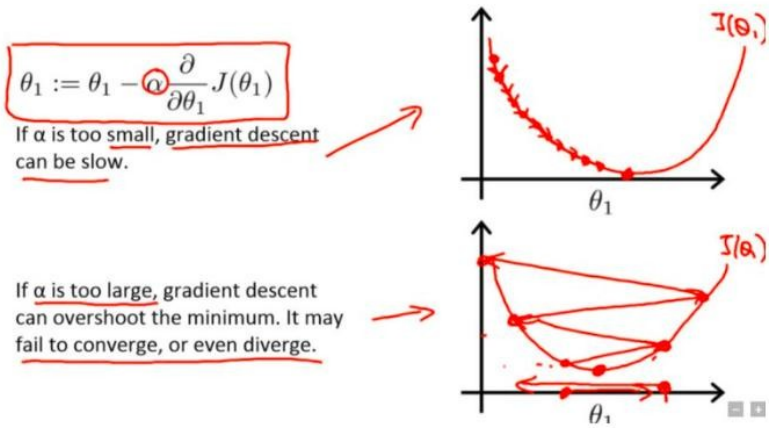


Figure 3.19: Effect of learning rate on loss function of a model

3.21 Information Gain

Information Gain is a key concept of information theory that applies significantly to machine learn-ing. While building a machine learning model on an existing dataset, there are several attributes or fields that contribute to the information content of the dataset and thus the accuracy of the model. The addition or removal of these features can alter the performance of the model.

Information gain is the amount of information or entropy that is gained by knowing the value of the attribute. This is given by calculating the difference of the entropy of the distribution after the attribute is added from the entropy of the distribution before it. It basically measures how much “information” a feature gives us about the class. The largest information gain is a desirable state. Often, information gain is a key concept used in building *decision trees*. Decision Trees algorithms always try to maximize information gain. Information gain of a dataset can be represented by the following formula:

$$InformationGain = entropy(ParentDataset) - [WeightedAverage] * (entropy(ChildDataset)) \quad (15)$$

Let us look at an example:

```
# import dataset
speed_df <- read.csv('./data/information_gain.csv')
print(speed_df)
```

```
#       Grade Bumpiness Speed.Limit Speed
   ## 1 steep bumpiness         yes  slow
  ## 2 steep      smooth         yes  slow
  ## 3      flat bumpiness        no  fast
  ##  4 steep      smooth         no  fast
  ##  5      flat bumpiness       yes  slow
```

```
#probability of occurrence of each class
P_slow<-nrow(speed_df[speed_df$Speed == "slow",])/nrow(speed_df)
P_fast<-nrow(speed_df[speed_df$Speed == "fast",])/nrow(speed_df)

#entropy of the Speed column in the parent dataset
H<- (P_slow*(log2(1/P_slow)))+(P_fast*(log2(1/P_fast)))
cat("The entropy of the parent dataset is: ",H)
```

```
## The entropy of the parent dataset is:        0.9709506
```

Now let us start looking at entropy contributed by each of the attributes. Let us start with the first attribute: Grade. The attribute divides the dataset in the following manner:

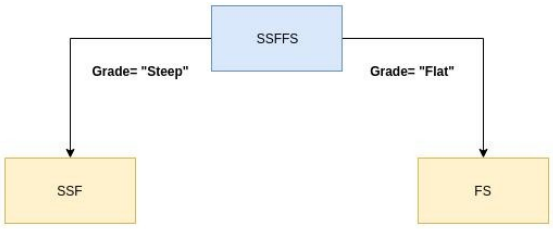


Figure 3.20: Split of label 'Speed' across various values of the attribute 'Grade'

```
#The split of classes across the various values of the first attribute: Grade #For Grade=
"Steep"
P_slow_gs<- 2/3
P_fast_gs<- 1/3

#Entropy of the dataset with respect to Grade='Steep'
entropy_grade_steep<-((P_slow_gs)*(log2(1/P_slow_gs)))+(P_fast_gs*(log2(1/P_fast_gs)))

#Doing a similar exercise for Grade='Flat'
P_slow_gf<- 1/2
P_fast_gf<- 1/2

#Entropy of the dataset with respect to Grade='Steep'
entropy_grade_flat<-((P_slow_gf)*(log2(1/P_slow_gf)))+(P_fast_gf*(log2(1/P_fast_gf)))

#Applying equation (15) to find the entropy of the child dataset:
#[Weighted avg]Entropy(children) =
```

```
#(no. of examples in left child node) /
#(total no. of examples in parent node) * (entropy of left node)
#+
#(no. of examples in right child node)/
#(total no. of examples in parent node) * (entropy of right node)

H_child<- ((3/5)*entropy_grade_steep)+((2/5)*entropy_grade_flat)

#print results
cat("Entropy of the child dataset w.r.t Grade is: ", H_child)
```

```
## Entropy of the child dataset w.r.t Grade is:  0.9509775
```

```
#Information Gain due to 'Grade'
Info_gain<- H-H_child
cat("Information Gain due to the attribute Grade is: ", Info_gain)
```

```
## Information Gain due to the attribute Grade is:  0.01997309
```

We can do a similar exercise for all the attributes. The aim is to select attributes that maximize information gain. We get:

- Information Gain from attribute ‘Bumpiness’: 0.0199
- Information Gain from attribute ‘Speed.Limit’: 1 (This can clearly be seen as all of the examples in each node of this attribute belong to the same class)

Thus, the final decision tree for this dataset would look like this:

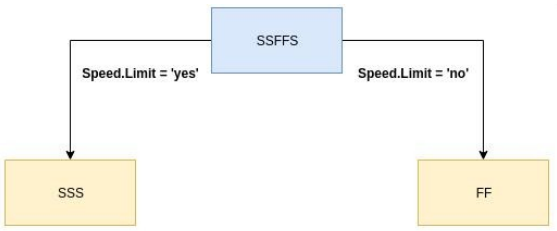


Figure 3.21: Split of label ’Speed’ across various values of the attribute ’Speed.Limit’

3.22 Information Bottleneck Method

The *Information Bottleneck* (IB) method is a technique of maximization of information that has a lot of applications across various fields. In machine learning, it is used in dimensionality reduction, clustering and even in deep learning. Let us first look at how this principle works.

Suppose you have a system with input $x \in X$ and a corresponding output (or another signal) such that $y \in Y$. We first start with computing how much information X gives about Y. This would be

the mutual information ($I(X, Y)$). The mutual information should be a positive variable (i.e. Y must not be independent from the original signal X). Now, we need to encode and find a short code (X_E) for X such that the relevant information about Y is preserved to its maximum capacity. This is the Information Bottleneck (IB) method. IB method attempts to minimize the quantity $I(X, X_E)$ to gain maximum compression while maximizing the mutual information $I(X_E, Y)$. Here, X_E is the information bottleneck.

The information between these variables can be best represented in the following manner:

$$I(X_E, Y) = \sum_{y \in Y} \sum_{x_E \in X_E} p(y, x_E) log(p(y, x_E)/(p(y)p(x_E))) \leq I(X, Y)$$

This is because most real-world compression systems are lossy compressions. Lossy compressions cannot convey more information than the original data (X). In this process, there is a trade-off between *compressing the representation* and *preserving meaningful information*. There is no single right solution for the trade-off. We are looking for a solution that keeps a fixed amount of meaningful information about the relevant signal 'Y' while minimizing the number of bits from the original signal 'X' (maximizing the compression). The solution to the same is as follows:

$$min_{p(X_E|X)}[(I(X, X_E)) - (\beta(I(X_E, Y)))] \quad (16)$$

Here β is the Lagrange multiplier attached to the constrained meaningful information while maintaining the normalization of the mapping $p(X_E|X)$ for every x. β reflects the trade-off between compression and preservation of mutual information.

3.22.1 Application of IB method in Machine Learning

1. Information Bottleneck method can widely be used in **dimensionality reduction**. This is because we need to compress the base dataset into its most concise form i.e. reduce the dimensions (attributes) in a dataset so as to retain the maximum information contained in the base dataset. We need to strike a balance between the number of attributes in the dataset and the information those attributes carry collectively.
2. IB method also finds its application in **clustering**. Clustering is viewed as lossy data compression because the identity of individual points is replaced by the identity of the cluster to which they are assigned. Information theory is relevant in this context as two points can be clustered together if this merger does not lose too much information about the relevant variable. For instance, you can improve the performance of K-means clustering to converge at the best possible clusters with maximum information using the minimization constraint as defined in equation (15).
3. IB method can also be used in another specific case of clustering i.e. **topic modelling and document summarization and clustering**. For this, it first generates a partitioning of the words (with encoded words: $p(W_E|W)$), which is supposed to preserve information about the documents. Then, the original document representation is replaced by a representation based on the word-clusters. Then, a partitioning of the documents $p(D_E|D)$ is found that preserves information about the words.

4 Information Theory in Feature Engineering

Knowledge discovery is a process that helps you engineer features or attributes in a base dataset to improve the performance of the machine learning models. Information theory has a lot of 'information measures' or metrics that can help you evaluate the importance and relevance of each new attribute that you develop. For instance, if you develop new attributes in a dataset, you should use measures like mutual information and conditional entropy to evaluate how significant an explanatory variable is with respect to the target or the dependent variable.

We have already seen in the sections above that conditional entropy determines how two variables are correlated. If $H(Y|X = x_i)$ represents the conditional entropy of a variable 'Y' given $X = x_i$. Let us assume that 'Y' represents the target variable in a machine learning model and 'X' is an explanatory variable, say a categorical variable with several labels (x_i) where 'i' specifies the number of unique labels of the variable 'X'. In this case, we need to ensure that $H(Y|X = x_i)$ when aggregated for all 'i' is close to zero. This is because $H(Y|X) = 0$ *if and only if the value of 'Y' is completely determined by the value of X.* The value of conditional entropy $H(Y|X)$ is equal to $H(Y)$ if and only if 'Y; and 'X' are independent random variables. Our aim while designing and collecting variables is to achieve a conditional entropy value of close to '0'. In terms of mutual information, we need to collate and engineer features such that the mutual information between 'Y' and 'X' is maximized. Below is a table of some important information measures that we have already covered so far, that can be used for this purpose.

Name	Formula	(Dis)similarity	(A)symmetry
Joint Information	$H(T,Y) = -\sum_t \sum_y p(t,y) \log_2 p(t,y)$	Inapplicable	Symmetry
Mutual Information	$I(T,Y) = \sum_t \sum_y p(t,y) \log_2 \frac{p(t,y)}{p(t)p(y)}$	Similarity	Symmetry
Conditional Entropy	$H(Y\|T) = -\sum_t \sum_y p(t,y) \log_2 p(y\|t)$	Dissimilarity	Asymmetry
Cross Entropy	$H(T;Y) = -\sum_z p_t(z) \log_2 p_y(z)$	Dissimilarity	Asymmetry
KL Divergence	$KL(T,Y) = \sum_z p_t(z) \log_2 \frac{p_t(z)}{p_y(z)}$	Dissimilarity	Asymmetry

Figure 4.1: Information Measures that can be used as learning and content measures

5 Information Theory in Feature Selection

While feature engineering includes generation of data, feature selection is a key step in machine learning where you select the most important and relevant features for the given problem statement. Information theory is a key concept used in this step. You can design your optimization function using measures from information theory. Let us look at a few possible techniques.

5.1 Mutual Information

As we see in equation (2), Shannon's definition of entropy relies on class prior probabilities. Let us also factor condition entropy in from equation (3). Let us assume a classification problem with class identities of the target variable represented as 'C'. If 'X' represents your feature space vector, the uncertainty of the class identity can be quantified using the conditional entropy:

$$H(C|X) = \sum_{x \in X} p(x) \sum_{c \in C} p(c|x) log_2(1/p(c|x))$$

The amount by which the class uncertainty is reduced after having observed the feature vector 'X' is called the mutual information, $I(C, X) = H(C) - H(C|X)$. Mutual Information can also be defined using equation (4) above. Since mutual information measures independence between two variables, in this case between C and X. It is 0 when $p(c, x) = p(c)p(x)$ i.e. when the joint density of C and X can be defined as a direct product of marginal densities, which is the condition for independence of the two variables. (This condition can also be defined as the Kullback-Leibler divergence measure between $p(c, x)$ and $p(c)p(x)$.) Now, we need to design an optimization objective function such that the mutual information between C and X is maximized or $H(C|X)$ is minimized. There are a few constraints for the same.

1. **Lower Bound**: A lower bound to the probability of error when estimating a discrete random variable 'C' from another random variable 'X' can be defined using Fano's bound. It can be represented as:

$$P\,r(c \neq \hat{c}) \geq (H(C|X) - 1)/log(N_c) = (H(C) - I(C, X) - 1)/log(N_c)$$

   Here, $\hat{c}$ is the estimate of C after observing a sample of X, which can be scalar or multivariate and N_c represents the number of different classes. We are trying to optimize the probability that ($\hat{c} \neq c$). This lower bound on error probability is minimized when the mutual information between C and X is maximized. If we find such features, we can achieve the lowest possible bound to the error of a classifier.

2. **Upper Bound**: The upper bound that can be set on the probability error can be represented as follows:

$$P\,r(c \neq \hat{c}) \leq 1/2(H(C|X)) = 1/2(H(C) - I(C, X))$$

   Both the upper and lower bounds can be minimized by either maximizing the mutual information between C and X or by minimizing H(C|X).

5.1.1 Application of optimization of mutual information for feature selection

The idea of using mutual information is to design an objective function that runs on your high dimensional training data. Let us represent the training dataset as (x_i, c) where x_i is the feature space and c represents the various classes. The objective is to find a transformation g (or its parameter or weight vector w) such that $\hat{x}_i = g(w, xi)$ maximizes $I(C, X)$, the mutual information between transformed data X and class labels C. For this process to work, we need to estimate $I(C, X)$ as a function of the dataset in a differentiable form. This process will be a reiterative and recursive process till you find the most optimized solution. To find the factor for the learning process, a gradient ascent (representing the learning parameter) can be performed on $I(C, X)$. This process can be represented best using the below diagram.

5.2 Renyi Entropy

We know that mutual information can be computed using probability mass functions. Histograms are one of the most popularly used probability mass functions. However, they are parametric. The issue with parametric methods of estimating information measures is that they work well with 2 or 3 variables. However, with non-parametric methods, you can evaluate and compare multiple

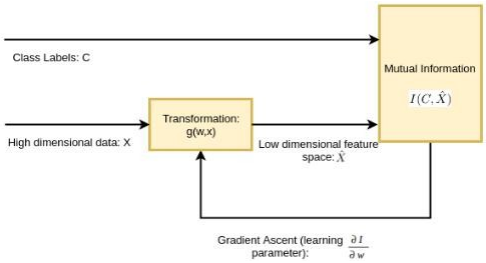


Figure 5.1: Feature selection by maximizing the mutual information between class labels and transformed features.

variables. This is ideal for high-dimensional data. One of the non-parametric evaluations of entropy in the Renyi Entropy.

Renyi's quadratic entropy is a generalized form of Shannon's entropy that we have covered earlier. It can be represented in the following manner:

$$H_\alpha(X) = (1/(1-\alpha))log_2(\sum_{i=1}^{n_x}(p_i^\alpha)) \quad (17)$$

As we see above, for the value (α = 1), the equation provides Shannon entropy. α can assume values [0, ∞]. However, Renyi's entropy has computational advantages in the sense that this parametric measure helps you find a distribution (instead of a particular entropy value) that maximizes or minimizes the entropy given some constraints. For continuous variables, you can use the integral operation instead of the discrete aggregation.

The above metrics can then be used to compute mutual information of a class variable with respect to the features. The aim would be to select features such that the mutual information is maximized.

6 Information Theory in Model Selection

Model selection is an important bit in machine learning. With an increasing number of algorithms available for both supervised and unsupervised, it is necessary we choose the best model. This means we also need to penalize model for complexity when the model introduces more complexity than what is needed to fit the regularities of the data. We need to find a model that has an optimal goodness-of-fit and generalization (or generalizability). Generalization basically helps you avoid overfitting of the model. It is claimed that as the model complexity increases, the goodness-of-fit increases but the generalization ability of the model decreases. In other words, a very complex model shows a tendency of overfitting. Below is an image representing the same.

Let us now look at a few methods of model selection. A few common ways of selecting the best model are as follows:

6.1 AIC: Akaike Information Criterion

The Akaike Information Criterion (AIC) is a way of selecting a good model from a set of models. It is based on the concept of KL divergence. In simple terms, it ensures that the best model hence

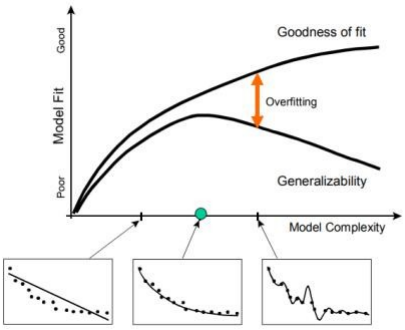


Figure 6.1: Relationship between generalizability and model fit for varying degrees of model com-plexity

selected minimizes the Kullback-Leibler (KL) divergence between the model and the truth. In general, AIC can be defined in the following manner.

$$AIC = 2K - 2(log(\hat{L}))$$

Here, K is the number of model parameters (the number of variables in the model plus the intercept) required to make the model estimations. $\hat{L}$ denotes the maximum likelihood function of the model and thus $log(\hat{L})$ is the log-likelihood of the model. It can also be represented as:

$$AIC = 2K + T\, log(RSS)$$

Here, RSS denotes the residual sum of squared errors, T denotes the number of observations and K stands for the model parameters or regressors. If you have a set of models for your given data, the preferred model is the one with the smallest AIC value. In the above equations we see that AIC rewards goodness-of-fit (represented by the element with the likelihood function). However, it also penalizes a model by a factor that is an increasing function of the number of parameters (K). This penalty discourages overfitting, because increasing the number of parameters in the model generally improves the goodness of its fit.

The above equation is slightly modified for small sample sizes (i.e. $T/K \leq 40$). It can be represented as a second-order AIC (or AIC_C)

$$AIC_C = 2K - 2(log(\hat{L})) + (2K(K+1)/(T - K - 1))$$

The above equation follows the same conventions as the previous equation. Every model will thus have an AIC. Let us briefly see how to understand and use an AIC score. In a given set of R models, let us list the AIC values of each of these models by $AIC_1, AIC_2, AIC_3, ..., AIC_R$. Let AIC_{min} be the minimum of those values. Then the quantity $exp((AIC_{min} - AIC_i)/2)$ is proportional to the probability that the i-th model minimizes the (estimated) information loss.

While AIC seems to be a reliable measure for selecting models, there are a few caveats in the same. AIC can only compare a given set of models and not determine the absolute quality of a model. In other words, it will give you the best model out of the set of models you have (relative quality), but there might be another model outside your sample set that is a better model. Moreover, it is advised to not use AIC for comparing a large number of models.

6.2 BIC: Bayesian information criterion

Bayesian information criterion (BIC) or Schwarz information criterion (also SIC, SBC, SBIC) is yet another information measure for model selection. It is closely related to AIC and can be interpreted in the same manner i.e. the model with the smallest BIC value is the best model. It can be defined as:

$$BIC = log(T)K - 2log(\hat{L})$$

As we see, there is not much difference between the equations for AIC and BIC. Following the same conventions as earlier, we see that the penalty term has an extra weightage by a factor of the number of observations. As a result, penalty for additional parameters is more in BIC than AIC.

One advantage of using BIC over AIC is that AIC generally tries to find an unknown model that has a high dimensionality. This means that all the models might not be true models in AIC. On the other hand, the Bayesian Information Criteria or BIC comes across only true models. This also implies that BIC is more consistent as compared to AIC while estimating for the best model.

6.3 MDL: Minimum Description Length

It is a complex concept based on algorithmic coding theory that represents machine learning models and data as compressible codes. The working principle of MDL for model selection is as follows:

The best model is the one that provides the shortest description length of the data in bits by "compressing" the data as tightly as possible.. In other words, this means that the model that requires the least number of bits to describe the actual data is the best. Like the other two methods described above, you need to choose a model with the least MDL (minimum description length).

There are multiple methods to estimate the value of an MDL for a model. Some of them are: 1. Fisher Information Approximation (FIA) 2. Normalized Maximum Likelihood (NML)

The detailed description of these techniques are beyond the scope of the paper currently.

7 Information Theory in Regression

We have studied how KL divergence can be used in regression to reduce the mean -squared loss error. Let us also cover briefly how mutual information and linear regression might be highly related.

We know from equation (4) that mutual information can be defined as $H(Y) - H(Y|X)$. Since regression is a form of continuous estimation, we would like to assess an information measure for continuous variables, like differential entropy. We also saw earlier that differential entropy of a Gaussian is equal to a constant plus the log of its standard deviation. We can extend this concept to the expression for mutual information, where both $H(Y)$ and $H(Y|X)$ are Gaussian distributions. While $H(Y)$ has a standard deviation of 1, $H(Y|X)$ represents the loss of information while estimating for Y from X (or the error) and hence has a standard deviation value of σ_e i.e. standard deviation of the error term. On substituting the same in equation (4), we get:

$$I(X, Y) = H(Y) - H(Y|X) = [constant + log(1)] - [constant + log(\sigma_e)] = -log(\sigma_e) \quad (18)$$

Now, we know that r-squared (R^2) is a significant measure in regression that represents the explained variance in a target variable by the features/explanatory variables. Higher the value of R^2, lower is the error term or the loss of information while estimating Y from X. R-squared can also be represented as :

$$R^2 = 1 - \sigma_e^2$$

This indicates, σ_e can be defined as $\sqrt{(1 - R^2)}$. We can simply substitute this in the equation for mutual information above (equation (18)) and say that:

$$I(X, Y) = log(1/\sqrt{(1 - R^2)})$$

The above equations shows a direct relation between linear regression and mutual information.

8 Information Theory in Classification

We have covered the concept of cross entropy loss and log loss as standard loss functions for a lot of classification algorithms. These loss functions are used both in binary classification as well multiclass classifications.

These loss functions can be used to refine the performance of your classification algorithms. You can refer to equation (13) for the generic form of cross entropy loss. A more custom form of the equation for binary classification is the log loss function.

We also saw how entropy is used to build decision trees. Entropy and information gain are used to construct each step of a decision tree. This can help you find the most relevant feature for a particular split in a decision tree. KL divergence is yet another significant information measure used in classification.

9 Information Theory in Clustering

Clustering is a kind of unsupervised learning problem in machine learning. The goal of (hard) clustering is to assign data instances into one of the groups (clusters) such that the instances in the same cluster exhibit similar properties.We saw in the earlier sections how the Information Bottleneck method can be used for clustering by maximizing compression of individual data points into clusters while ensuring that the loss of information is reduced (or the mutual information is maximized). Furthermore, we can use the concept of mutual information for information theoretic clustering.

9.1 Mutual Information Criterion for Information Theoretic Clustering

Let us assume that 'Y' represents the cluster identities for your data elements and 'X' defines the features or the variables defining the attributes of these data points. The objective of clustering would be to maximize the mutual information of X and Y i.e. $I(X, Y)$.

From equation (4), we know that $I(X, Y) = H(X) - H(X|Y)$. Now the entropy of the features (X) is independent of the type of clustering mechanism used. We thus need to evaluate the conditional entropy $H(X|Y)$ to evaluate our clusters. The objective thus boils down to minimizing $H(X|Y)$.

10 Conclusion

The use of information theory extends beyond the applications described in this paper. Information theory can also be used in applications like selection of distance metrics for algorithms like KNN, creation of factor graphs for Bayesian Networks etc.

There are basically two modes of applying information theory in the machine learning: 1. Apply fundamental information measures and concepts like entropy, mutual information, KL-divergence as objective functions or regularization terms in an optimization problem. This will help you improve the performance an existing algorithm or model. 2. Develop new algorithms and techniques using concepts from sophisticated information theory, such as rate-distortion theory and coding theory. This might help you provide additional insights for existing machine learning techniques and in that process, develop improvised versions of the existing algorithms.

11 References

1. *Visual Information Theory*. Retrieved from http://colah.github.io/posts/2015-09-Visual-Information/
2. *Information Theory: A Tutorial Introduction* https://arxiv.org/pdf/1802.05968.pdf
3. *KL Divergence for Machine Learning*. Retrieved from https://dibyaghosh.com/blog/probability/kldivergence.html
4. *Information Theory, Inference, and Learning Algorithms*. Retrieved from https://www.inference.org.uk/itprnn/book.pdf
5. *Focal Loss for Dense Object Detection* https://arxiv.org/pdf/1708.02002.pdf
6. *5 Regression Loss Functions All Machine Learners Should Know* https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
7. *What is Entropy and why Information gain matter in Decision Trees?*. Retrieved from https://medium.com/coinmonks/what-is-entropy-and-why-information-gain-is-matter-4e85d46d2f01
8. *The Information Bottleneck Method*. Retrieved from https://arxiv.org/pdf/physics/0004057.pdf
9. *Rényi Entropy*. Retrieved from https://www.johndcook.com/blog/2018/11/21/renyi-entropy/
10. *How do I interpret the AIC*. Retrieved from https://www.r-bloggers.com/how-do-i-interpret-the-aic/
11. *Model Selection Methods*. Retrieved from https://faculty.psy.ohio-state.edu/myung/personal/model%20selection%20tutorial.pdf
12. *Information Theory and Machine Learning*. Retrieved from https://pdfs.semanticscholar.org/d6d9/b285738560963810bf15c68210a21b05de23.pdf