Information Gain and Mutual Information for Machine Learning

by **Jason Brownlee** on October 16, 2019 in **Probability**

Last Updated on December 10, 2020

Information gain calculates the reduction in entropy or surprise from transforming a dataset in some way.

It is commonly used in the construction of decision trees from a training dataset, by evaluating the information gain for each variable, and selecting the variable that maximizes the information gain, which in turn minimizes the entropy and best splits the dataset into groups for effective classification.

Information gain can also be used for feature selection, by evaluating the gain of each variable in the context of the target variable. In this slightly different usage, the calculation is referred to as mutual information between the two random variables.

In this post, you will discover information gain and mutual information in machine learning.

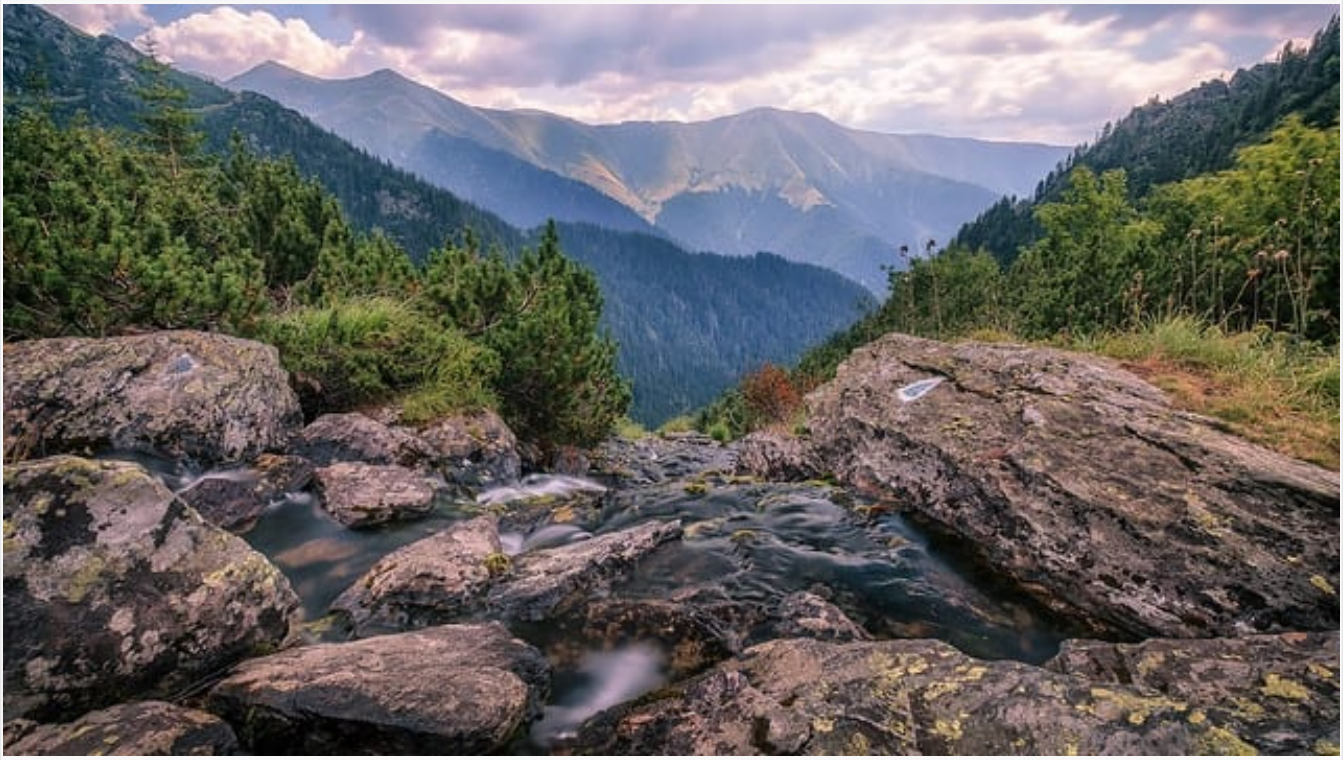
After reading this post, you will know:

- Information gain is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees.
- Information gain is calculated by comparing the entropy of the dataset before and after a transformation.
- Mutual information calculates the statistical dependence between two variables and is the name given to information gain when applied to variable selection.

Kick-start your project with my new book Probability for Machine Learning, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

- **Update Nov/2019**: Improved the description of info/entropy basics (thanks HR).
- **Update Aug/2020**: Added missing brackets to equation (thanks David)



What is Information Gain and Mutual Information for Machine Learning
Photo by Giuseppe Milo, some rights reserved.

Overview

This tutorial is divided into five parts; they are:

1. What Is Information Gain?
2. Worked Example of Calculating Information Gain
3. Examples of Information Gain in Machine Learning
4. What Is Mutual Information?
5. How Are Information Gain and Mutual Information Related?

What Is Information Gain?

Information Gain, or IG for short, measures the reduction in entropy or surprise by splitting a dataset according to a given value of a random variable.

A larger information gain suggests a lower entropy group or groups of samples, and hence less surprise.

You might recall that **information** quantifies how surprising an event is in bits. Lower probability events have more information, higher probability events have less information. **Entropy** quantifies how much information there is in a random variable, or more specifically its probability distribution. A skewed distribution has a low entropy, whereas a distribution where events have equal probability has a larger entropy.

In information theory, we like to describe the "*surprise*" of an event. Low probability events are more surprising therefore have a larger amount of information. Whereas probability distributions where the events are equally likely are more surprising and have larger entropy.

- **Skewed Probability Distribution** (*unsurprising*): Low entropy.
- **Balanced Probability Distribution** (*surprising*): High entropy.

For more on the basics of information and entropy, see the tutorial:

- A Gentle Introduction to Information Entropy

Now, let's consider the entropy of a dataset.

We can think about the entropy of a dataset in terms of the probability distribution of observations in the dataset belonging to one class or another, e.g. two classes in the case of a binary classification dataset.

> *One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S (i.e., a member of S drawn at random with uniform probability).*

— Page 58, Machine Learning, 1997.

For example, in a binary classification problem (two classes), we can calculate the entropy of the data sample as follows:

- Entropy = -(p(0) * log(P(0)) + p(1) * log(P(1)))

A dataset with a 50/50 split of samples for the two classes would have a maximum entropy (maximum surprise) of 1 bit, whereas an imbalanced dataset with a split of 10/90 would have a smaller entropy as there would be less surprise for a randomly drawn example from the dataset.

We can demonstrate this with an example of calculating the entropy for this imbalanced dataset in Python. The complete example is listed below.

```
# calculate the entropy for a dataset
from math import log2
# proportion of examples in each class
class0 = 10/100
class1 = 90/100
# calculate entropy
entropy = -(class0 * log2(class0) + class1 * log2(class1))
# print the result
print('entropy: %.3f bits' % entropy)
```

Running the example, we can see that entropy of the dataset for binary classification is less than 1 bit. That is, less than one bit of information is required to encode the class label for an arbitrary example from the dataset.

```
entropy: 0.469 bits
```

In this way, entropy can be used as a calculation of the purity of a dataset, e.g. how balanced the distribution of classes happens to be.

An entropy of 0 bits indicates a dataset containing one class; an entropy of 1 or more bits suggests maximum entropy for a balanced dataset (depending on the number of classes), with values in between indicating levels between these extremes.

Information gain provides a way to use entropy to calculate how a change to the dataset impacts the purity of the dataset, e.g. the distribution of classes. A smaller entropy suggests more purity or less surprise.

> *… information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.*

— Page 57, Machine Learning, 1997.

For example, we may wish to evaluate the impact on purity by splitting a dataset *S* by a random variable with a range of values.

This can be calculated as follows:

- IG(S, a) = H(S) – H(S | a)

Where *IG(S, a)* is the information for the dataset *S* for the variable a for a random variable, *H(S)* is the entropy for the dataset before any change (described above) and *H(S | a)* is the

conditional entropy for the dataset given the variable *a*.

This calculation describes the gain in the dataset *S* for the variable a. It is the number of bits saved when transforming the dataset.

The conditional entropy can be calculated by splitting the dataset into groups for each observed value of a and calculating the sum of the ratio of examples in each group out of the entire dataset multiplied by the entropy of each group.

- H(S | a) = sum v in a Sa(v)/S * H(Sa(v))

Where *Sa(v)/S* is the ratio of the number of examples in the dataset with variable a has the value *v*, and *H(Sa(v))* is the entropy of group of samples where variable a has the value *v*.

This might sound a little confusing.

We can make the calculation of information gain concrete with a worked example.

Worked Example of Calculating Information Gain

In this section, we will make the calculation of information gain concrete with a worked example.

We can define a function to calculate the entropy of a group of samples based on the ratio of samples that belong to class 0 and class 1.

```
# calculate the entropy for the split in the dataset
def entropy(class0, class1):
    return -(class0 * log2(class0) + class1 * log2(class1))
```

Now, consider a dataset with 20 examples, 13 for class 0 and 7 for class 1. We can calculate the entropy for this dataset, which will have less than 1 bit.

```
...
# split of the main dataset
class0 = 13 / 20
class1 = 7 / 20
# calculate entropy before the change
s_entropy = entropy(class0, class1)
print('Dataset Entropy: %.3f bits' % s_entropy)
```

Now consider that one of the variables in the dataset has two unique values, say "*value1*" and "*value2*." We are interested in calculating the information gain of this variable.

Let's assume that if we split the dataset by value1, we have a group of eight samples, seven for class 0 and one for class 1. We can then calculate the entropy of this group of samples.

```
...
# split 1 (split via value1)
s1_class0 = 7 / 8
s1_class1 = 1 / 8
# calculate the entropy of the first group
s1_entropy = entropy(s1_class0, s1_class1)
print('Group1 Entropy: %.3f bits' % s1_entropy)
```

Now, let's assume that we split the dataset by value2; we have a group of 12 samples with six in each group. We would expect this group to have an entropy of 1.

```
...
# split 2  (split via value2)
s2_class0 = 6 / 12
s2_class1 = 6 / 12
# calculate the entropy of the second group
s2_entropy = entropy(s2_class0, s2_class1)
print('Group2 Entropy: %.3f bits' % s2_entropy)
```

Finally, we can calculate the information gain for this variable based on the groups created for each value of the variable and the calculated entropy.

The first variable resulted in a group of eight examples from the dataset, and the second group had the remaining 12 samples in the data set. Therefore, we have everything we need to calculate the information gain.

In this case, information gain can be calculated as:

- Entropy(Dataset) – (Count(Group1) / Count(Dataset) * Entropy(Group1) + Count(Group2) / Count(Dataset) * Entropy(Group2))

Or:

- Entropy(13/20, 7/20) – (8/20 * Entropy(7/8, 1/8) + 12/20 * Entropy(6/12, 6/12))

Or in code:

```
...
# calculate the information gain
gain = s_entropy - (8/20 * s1_entropy + 12/20 * s2_entropy)
print('Information Gain: %.3f bits' % gain)
```

Tying this all together, the complete example is listed below.

```
# calculate the information gain
from math import log2

# calculate the entropy for the split in the dataset
def entropy(class0, class1):
    return -(class0 * log2(class0) + class1 * log2(class1))

# split of the main dataset
class0 = 13 / 20
class1 = 7 / 20
# calculate entropy before the change
s_entropy = entropy(class0, class1)
print('Dataset Entropy: %.3f bits' % s_entropy)

# split 1 (split via value1)
s1_class0 = 7 / 8
s1_class1 = 1 / 8
# calculate the entropy of the first group
s1_entropy = entropy(s1_class0, s1_class1)
print('Group1 Entropy: %.3f bits' % s1_entropy)

# split 2  (split via value2)
s2_class0 = 6 / 12
s2_class1 = 6 / 12
# calculate the entropy of the second group
s2_entropy = entropy(s2_class0, s2_class1)
print('Group2 Entropy: %.3f bits' % s2_entropy)

# calculate the information gain
gain = s_entropy - (8/20 * s1_entropy + 12/20 * s2_entropy)
print('Information Gain: %.3f bits' % gain)
```

First, the entropy of the dataset is calculated at just under 1 bit. Then the entropy for the first and second groups are calculated at about 0.5 and 1 bits respectively.

Finally, the information gain for the variable is calculated as 0.117 bits. That is, the gain to the dataset by splitting it via the chosen variable is 0.117 bits.

```
Dataset Entropy: 0.934 bits
```

```
Group1 Entropy: 0.544 bits
Group2 Entropy: 1.000 bits
Information Gain: 0.117 bits
```

Examples of Information Gain in Machine Learning

Perhaps the most popular use of information gain in machine learning is in decision trees.

An example is the Iterative Dichotomiser 3 algorithm, or ID3 for short, used to construct a decision tree.

> *Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree.*

— Page 58, Machine Learning, 1997.

The information gain is calculated for each variable in the dataset. The variable that has the largest information gain is selected to split the dataset. Generally, a larger gain indicates a smaller entropy or less surprise.

> *Note that minimizing the entropy is equivalent to maximizing the information gain …*

— Page 547, Machine Learning: A Probabilistic Perspective, 2012.

The process is then repeated on each created group, excluding the variable that was already chosen. This stops once a desired depth to the decision tree is reached or no more splits are possible.

> *The process of selecting a new attribute and partitioning the training examples is now repeated for each non terminal descendant node, this time using only the training examples associated with that node. Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.*

— Page 60, Machine Learning, 1997.

Information gain can be used as a split criterion in most modern implementations of decision trees, such as the implementation of the Classification and Regression Tree (CART) algorithm in the scikit-learn Python machine learning library in the DecisionTreeClassifier class for classification.

This can be achieved by setting the criterion argument to "*entropy*" when configuring the model; for example:

```
# example of a decision tree trained with information gain
from sklearn.tree import DecisionTreeClassifier
model = sklearn.tree.DecisionTreeClassifier(criterion='entropy')
...
```

Information gain can also be used for feature selection prior to modeling.

It involves calculating the information gain between the target variable and each input variable in the training dataset. The Weka machine learning workbench provides an implementation of information gain for feature selection via the InfoGainAttributeEval class.

In this context of feature selection, information gain may be referred to as "*mutual information*" and calculate the statistical dependence between two variables. An example of using information gain (mutual information) for feature selection is the mutual_info_classif() scikit-learn function.

What Is Mutual Information?

Mutual information is calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable.

> *A quantity called mutual information measures the amount of information one can obtain from one random variable given another.*

— Page 310, Data Mining: Practical Machine Learning Tools and Techniques, 4th edition, 2016.

The mutual information between two random variables *X* and *Y* can be stated formally as follows:

- I(X ; Y) = H(X) – H(X | Y)

Where *I(X ; Y)* is the mutual information for *X* and *Y*, *H(X)* is the entropy for *X* and *H(X | Y)* is the conditional entropy for *X* given *Y*. The result has the units of bits.

Mutual information is a measure of dependence or "*mutual dependence*" between two random variables. As such, the measure is symmetrical, meaning that *I(X ; Y) = I(Y ; X)*.

> *It measures the average reduction in uncertainty about x that results from learning*

> *the value of y; or vice versa, the average amount of information that x conveys about y.*

— Page 139, Information Theory, Inference, and Learning Algorithms, 2003.

Kullback-Leibler, or KL, divergence is a measure that calculates the difference between two probability distributions.

The mutual information can also be calculated as the KL divergence between the joint probability distribution and the product of the marginal probabilities for each variable.

> *If the variables are not independent, we can gain some idea of whether they are 'close' to being independent by considering the Kullback-Leibler divergence between the joint distribution and the product of the marginals […] which is called the mutual information between the variables*

— Page 57, Pattern Recognition and Machine Learning, 2006.

This can be stated formally as follows:

- I(X ; Y) = KL(p(X, Y) || p(X) * p(Y))

Mutual information is always larger than or equal to zero, where the larger the value, the greater the relationship between the two variables. If the calculated result is zero, then the variables are independent.

Mutual information is often used as a general form of a correlation coefficient, e.g. a measure of the dependence between random variables.

It is also used as an aspect in some machine learning algorithms. A common example is the Independent Component Analysis, or ICA for short, that provides a projection of statistically independent components of a dataset.

How Are Information Gain and Mutual Information Related?

Mutual Information and Information Gain are the same thing, although the context or usage of the measure often gives rise to the different names.

For example:

- Effect of Transforms to a Dataset (*decision trees*): Information Gain.
- Dependence Between Variables (*feature selection*): Mutual Information.

Notice the similarity in the way that the mutual information is calculated and the way that information gain is calculated; they are equivalent:

- $I(X ; Y) = H(X) – H(X | Y)$

and

- $IG(S, a) = H(S) – H(S | a)$

As such, mutual information is sometimes used as a synonym for information gain. Technically, they calculate the same quantity if applied to the same data.

We can understand the relationship between the two as the more the difference in the joint and marginal probability distributions (mutual information), the larger the gain in information (information gain).

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Books

- Information Theory, Inference, and Learning Algorithms, 2003.
- Machine Learning: A Probabilistic Perspective, 2012.
- Pattern Recognition and Machine Learning, 2006.
- Machine Learning, 1997.
- Data Mining: Practical Machine Learning Tools and Techniques, 4th edition, 2016.

API

- scipy.stats.entropy API

Articles

- Entropy (information theory), Wikipedia.
- Information gain in decision trees, Wikipedia.
- ID3 algorithm, Wikipedia.
- Information gain ratio, Wikipedia.
- Mutual Information, Wikipedia.

Summary

In this post, you discovered information gain and mutual information in machine learning.

Specifically, you learned:

- Information gain is the reduction in entropy or surprise by transforming a dataset and is often used in training decision trees.
- Information gain is calculated by comparing the entropy of the dataset before and after a transformation.
- Mutual information calculates the statistical dependence between two variables and is the name given to information gain when applied to variable selection.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.

More On This Topic