K-Nearest Neighbors
Decision Trees

PHYS 243 Lecture # 9

K-Nearest Neighbors (KNN)

The K-nearest Neighbors is a non-parametric method used for classification and regression.

The input consists of the k closest training examples in the feature space.

The output depends on whether k-NN is used for classification or regression.

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

KNN Classification Method

In KNN, we have an existing set of training data- the training set. We have labels for all these data (we know what class each set of data should fall into).

When we are given a new piece of data without a label, we compare that new piece of data to the existing data.

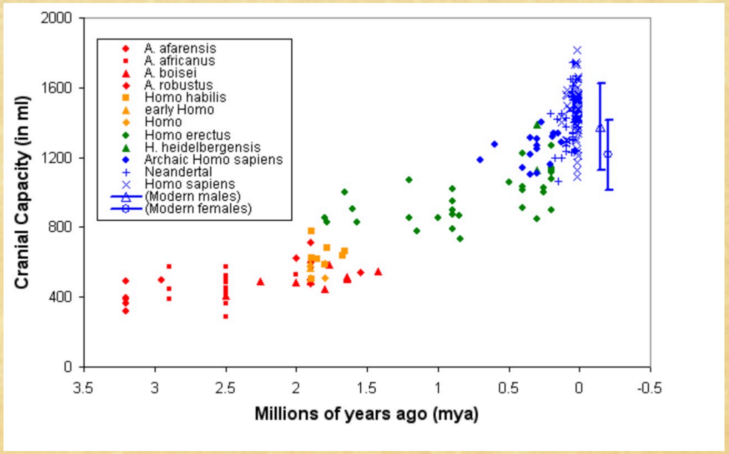
We then take the most similar pieces of data (the nearest neighbors) and look at their labels.

We look at the top k most similar pieces of data from our known dataset. This is where k comes from (k is an integer).

We then take a “majority vote” from the k most similar pieces of data and the majority is the new class we assign to the data we were asked to classify.

Example

The figure shows the evolution of mammal's brain size with time. For different spices, the size increased with time. Now, we consider the spices as the labels (*astrolapitus, Homo Ergaster, Homo Sapien* etc). Now, if we are given the brain size of a spices and the geological time it lived, how could we assign the label to this?



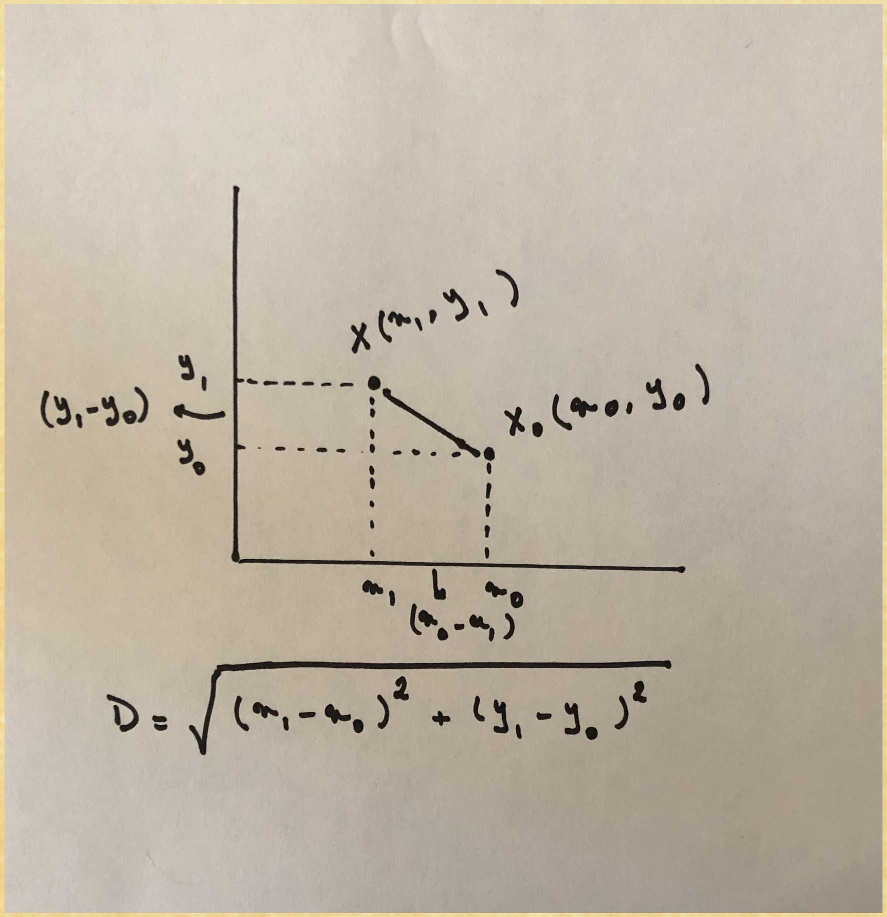
The KNN Method

We find the Euclidian distance between the point in question and all the other points in the training sample. This is the distance between any two vectors (x_o, y_o) and (x_1,y_1) measured by the relation:

$$d = [(x_1-x_0)^2 + (y_1-y_0)^2]^{1/2}$$

This distance is measured between the input point in question and all the rest of the points and the result sorted in terms of increasing distance. The sources within the k'*th* closest distance are selected. The vote between the labels of these points in the training sample will indicate the label to be assigned to the input point.

How to calculate distance between two points in 2 dimensions



$$D = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

How to Test a KNN Classifier

We need to find out if a classifier is right and if it has classified the point correctly.

To test a classifier we start with some data so that we could hide the answer from the classifier and ask the classifier to perform the classification on the data for which we know the answer. We then find the number of times the classifier was wrong and divide it by the total number of tests we gave it.

This will give us the error rate which is a common measure to decide how good a classifier is working. An error rate of zero means we have a perfect classifier while an error rate of 1 means all the classifications are wrong.

KNN Approach

To apply KNN one needs to take the following steps:

- Collect data within a file
- Prepare: Numeric values are needed for distance calculation. A structured data set would be the best.
- Analyze: apply the distance measurement method and find distances to each point.
- Training does not apply to KNN algorithms.
- Test: calculate the error rate
- The application runs the KNN algorithm on the input data and determines what class the input data should belong to.

K- Nearest Neighbors

Advantages: High accuracy, insensitivity to outliers, no assumptions about the data

Disadvantages: Computationally expensive, requires a lot of memory

The algorithm works with numeric or nominal values

KNN is a non-parametric learning algorithm. It uses a data set in which the data are separated into different classes and to predict the class of a new point.

Non-parametric means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data.

KNN does not use the training data to do any generalizations. KNN algorithms are based on feature similarities- how closely out of sample features resemble our training set determines how closely we determine a give data point.

KNN is used for classification with the output a class membership when it predicts a class. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors. It can also be used for regression where output is the value of the object. This value is the average or median of the values of its k nearest neighbors.

Applications of KNN

Credit ratings—collecting financial characteristics vs. comparing people with similar financial features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer’s credit rating, without having to perform all the calculations.

In political science—classing a potential voter to a “will vote” or “will not vote”, or to “vote Democrat” or “vote Republican”.

Pros and Cons of the KNN

Pros:

No assumptions about data—useful, for example, for nonlinear data

Simple algorithm—to explain and understand/interpret

High accuracy (relatively)—it is pretty high but not competitive in comparison to better supervised learning models

Versatile—useful for classification or regression

Cons:

Computationally expensive—because the algorithm stores all of the training data

High memory requirement

Stores all (or almost all) of the training data

Prediction stage might be slow (with big N)

Sensitive to irrelevant features and the scale of the data

A positive integer k is specified, along with a new sample

We select the k entries in our database which are closest to the new sample

We find the most common classification of these entries

This is the classification we give to the new sample

KNN stores the entire training dataset which it uses as its representation.

KNN does not learn any model.

KNN makes predictions just-in-time by calculating the similarity between an input sample and each training instance.

Types of KNN

KNN for Regression:

When KNN is used for regression problems the prediction is based on the mean or the median of the K most similar instances

KNN for Classification:

When KNN is used for classification, the output can be calculated as the class with the highest frequency from the K most similar instances. Each instance vote for their class and the class with most votes is taken as the prediction.

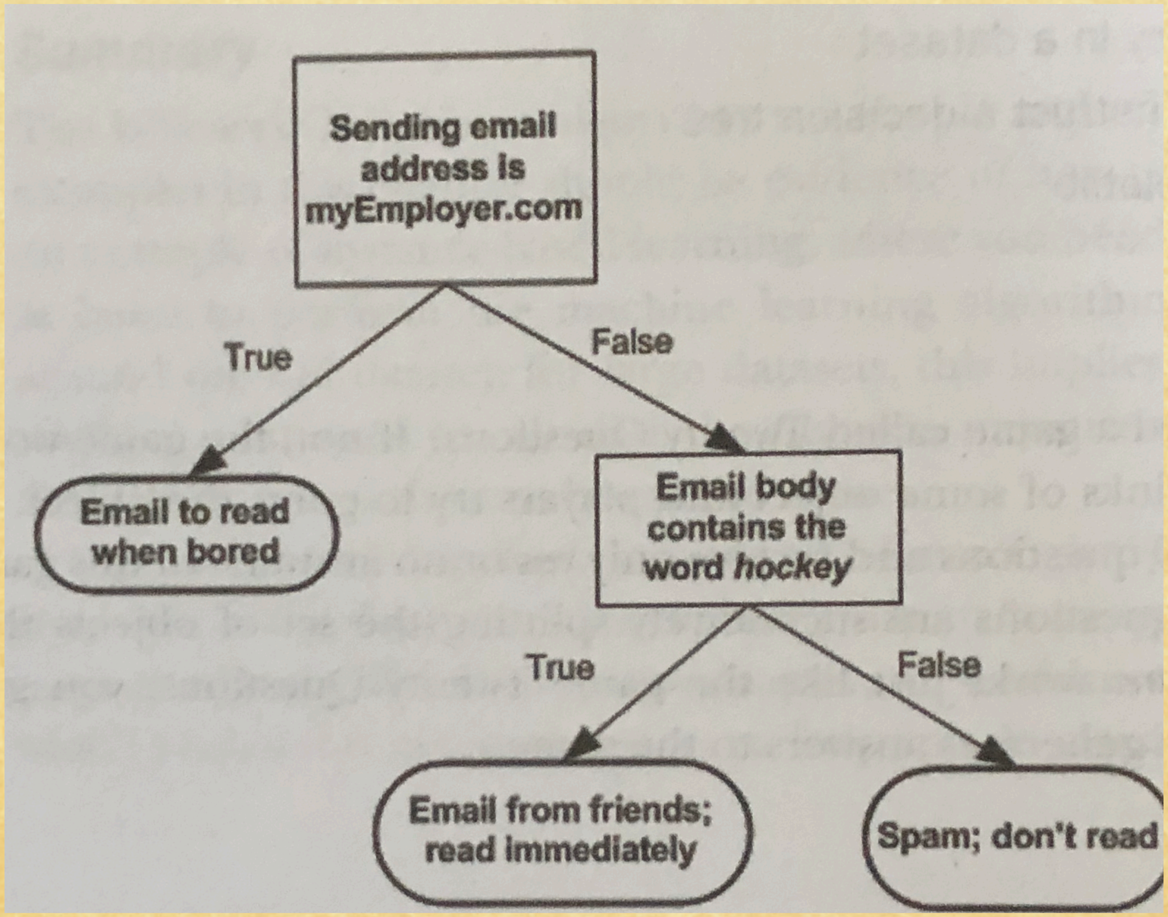
Decision Tree Classifier

The decision tree is one of the most commonly used classifiers.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules. A decision tree consists of three type of nodes

Decision node (represented by a square), chance node (represented by a circle) and end node (represented by ovals).

Example of a decision Tree


Sending email address is myEmployer.com
True
False
Email to read when bored
Email body contains the word *hockey*
True
False
Email from friends; read immediately
Spam; don't read

How to Build a Decision Tree

To build a decision tree we first need to understand the mathematics that decide how to split a dataset. This is based on information theory. Decision tree codes are written based on the principles in information theory.

- Data set we will split

- Feature we will split on

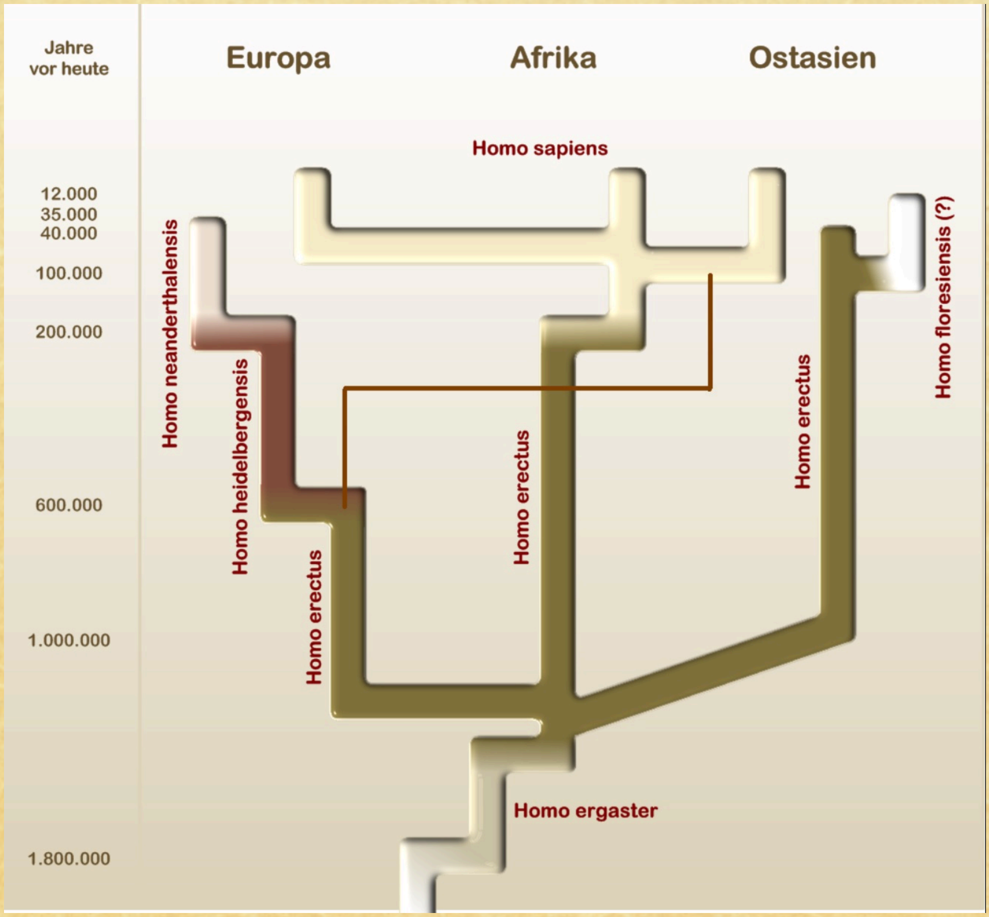
- Value of the feature

When to Stop the Splitting

The recursive splitting of the tree could continue until there is a single point per node. This is inefficient as it results in O(N) in computational cost. A common criterion for stopping the recursion is to cease splitting the nodes when either a node contains only one class of objects, when the split does not improve the information gain or when the number of points per node reaches a predefined value.

The recursive process could lead to overfitting the data. In decision tree complexity is defined by the depth of the tree or number of levels. A the depth increases the error on the training set decrease and at some point the tree will cease to represent the correlations within the data and will reflect the noise within the training data. We can use the entropy or misclassification error to optimize the depth of the tree.

Example of Decision Tree



Step-by-Step Method

To make a decision tree, you first need to make a first decision on a dataset to dictate what features to use to split the data.

To do this, you try every feature and measure which split will give you the best results.

After that, you will split the dataset into subsets. The subsets then transverse down the branches of the first decision node.

If the data on branches are of the same class, you don’t need to continue splitting it.

If the data isn’t the same, then need to repeat the splitting process on this subset.

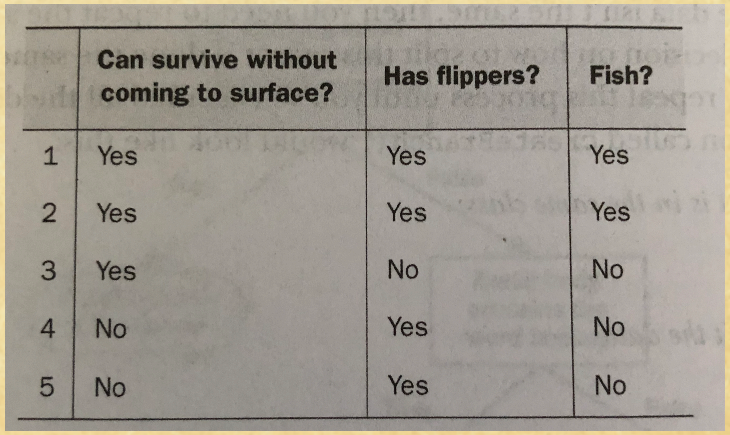
The decision as how to split the subset is made the same way as the original one. You repeat this process until you have classified all the data. Some decision trees make a binary split of the data.

Splitting the Dataset

1. Measure the entropy, split the dataset, measure the entropy on the split dataset$\rightarrow$ see if the feature used for the split was the right one
2. Do the above for all the features to determine the best feature to split on

Example

The table contains five animals pulled from the sea and asked if they could survive without coming to the surface and if they have flippers. We need to classify these animals into two classes- fish and not fish. Now we want to decide if we want to split the data based on the first feature or second. To answer this question, we need some quantitative way of determining how to split the data.



	Can survive without coming to surface?	Has flippers?	Fish?
1	Yes	Yes	Yes
2	Yes	Yes	Yes
3	Yes	No	No
4	No	Yes	No
5	No	Yes	No

To do this, we need to make our data more organized. One way to do this is to measure the information, using information theory.

This way you can measure and quantify the information before and after the split. The change in information before and after the split is called information gain. When you know how to calculate the information gain, you can split your data across every feature to see which split gives you the highest information gain. The split with the highest information gain is your best option.

Measuring Information

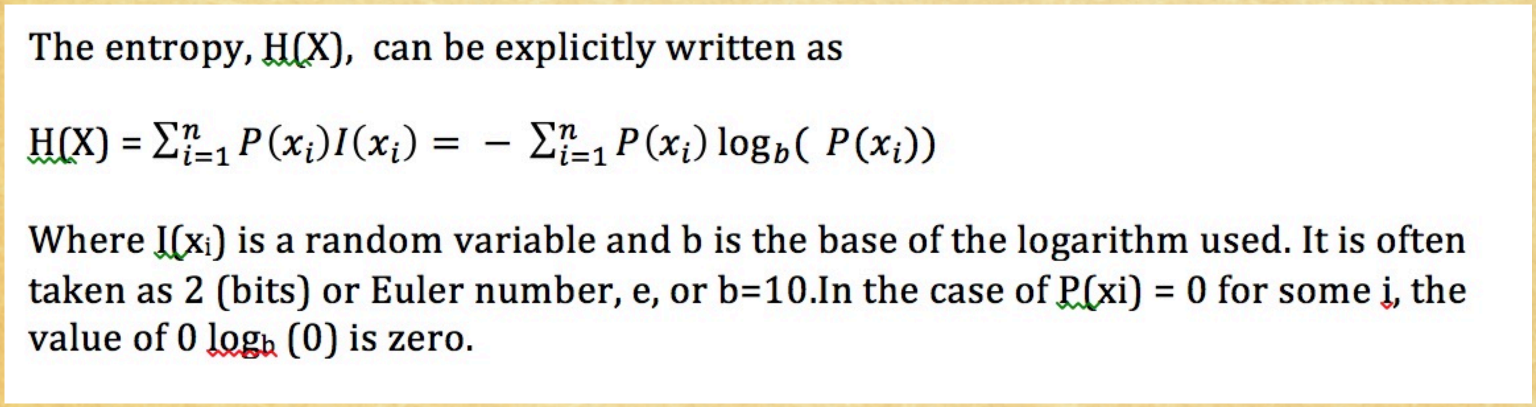
The measure of information of a set is known as entropy

Entropy is defined as the expected value of the information. First we need to define information. If you are classifying something that could take multiple values, the information for symbol x_i is defined as

$I(x_i)=\log_2 p(x_i)$

where $p(x_i)$ is the probability of choosing this class.

To calculate entropy, you need the expected value of all the information of all possible values of our class.



The entropy, H(X), can be explicitly written as

$$H(X) = \sum_{i=1}^{n} P(x_i) I(x_i) = -\sum_{i=1}^{n} P(x_i) \log_b(P(x_i))$$

Where $I(x_i)$ is a random variable and b is the base of the logarithm used. It is often taken as 2 (bits) or Euler number, e, or b=10.In the case of P(xi) = 0 for some i, the value of 0 $\log_b$ (0) is zero.

Entropy

The entropy is given by

$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where n is the number of classes. This measures the amount of disorder in a dataset. The higher the entropy, the more mixed up the data is.

Splitting the Dataset

You measure the amount of disorder in the dataset. For our classifier algorithm to work, you need to measure the entropy, split the dataset, measure the entropy on the split sets and see if splitting it was the right thing to do. You do this on all the features to see what feature is the best to split on. This is analogous to a 2 dimensional plot. You want to draw a line through the points to separate one class from another. Should you do this on the X or Y axis? The answer is what you are trying to find out here.

Using Decision Tree for Classification

You learned the tree from the training data. The algorithm takes the data under test and compares it against the values in the decision tree. It will do this recursively until it hits a leaf node. Then it will stop because it has arrived at a conclusion.

Decision Tree Approaches

Collect data

Prepare: the tree building algorithm only works on the nominal values. Therefore, any continuous value needs to be quantized.

Analyze: need to visually inspect the tree after it is built

Train: construct a tree data structure

Test: calculate the error rate with the learned tree

This can be used in any supervised task.

Decision Tree Characteristics

Advantages: Computationally cheap, easy to understand the learned results, it is OK to miss values, can deal with irrelevant features

Disadvantages: prone to overfitting, works with numeric and nominal values

A Brief Review of Information Theory

Information Theory

Information theory is the average rate with which information is produced by a stochastic data. This rate is measured by entropy that, for each value, is defined as the negative logarithm of the probability for that value. Therefore, when the data source has a lower probability value (when a low probability event occurs), the event carries more information than when the source data has a higher probability value. Generally entropy refers to disorder or uncertainty. The definition in information theory is the same. The concept of information entropy were introduced by Claude Shannon in 1948.

Entropy

Information entropy is the average amount of information conveyed by an event when considering all the possible outcomes. It could be looked at as follows: Data communication systems consist of three components: a source of data, a communication channel and a receiver. According to Shannon, the fundamental problem with communication is for the receiver to be able to identify what data was generated by the source, based on the signal it receives through the channel. The entropy provides an absolute limit on the shortest possible average length of a lossless compression encoding of the data produced by a source. If the entropy of the source is less than the channel capacity of the communication channel, the data generated by the source can be reliably communicated to the receiver.

Principles of Information Theory

Information theory is measured in bits. The unit of the measurement depends on the base of the logarithm that is used to define the entropy.

The logarithm of the probability is useful as a measure of the entropy because it is additive for independent sources. For example, the entropy of a fair coin is 1 bit, and the entropy of m tosses is m bits. Log_2 (n) bits are needed to represent a variable that can take one of n values if n is a power of 2.

If these values are equally probable, the entropy (in bits) is equal to this number. If one of the numbers is more probable than others, an observation that this value occurs is less informative than if some less common outcome had occurred.

Conversely, rare events provide more information when observed. Since observations of less probable events occurs more rarely, the net effect is that the entropy (thought of as average information) received from non-uniformly distributed data is always less than or equal to log2 (n). Entropy is zero when an event is certain to occur. The entropy quantifies these considerations when a probability distribution of the source data is known.

Formulating Information theory

The basic idea of information theory is that the more one knows about one event, less information is likely to get from it. If an event is very probable, when it happens it provides little new information. Inversely, if an event was improbable, when it happens it is much more informative. Therefore, the information content is an increasing function of the inverse of the probability of the event (1/p). Now, if more events happen, entropy measures the average information content one can expect to get if one of the events actually happens. This implies that casting a dice has more entropy than tossing a coin because each outcome of the dice has smaller probability than each outcome of the coin. In other words, entropy is a measure of the unpredictability of the state.

Consider the example of a coin toss. Assuming the probability of heads and tails are the same, the entropy of the coin toss is as high as it could be because there is no way one could predict the outcome- tail or head. The best we can do is to predict 50% probability for the coin to land as head or tail. Such a coin has one bit of entropy since there are two possible outcomes that occur with equal probabilities and learning the actual outcome has one bit of information. In contrast if both sides of the coin were tail, then we know what we are going to get. Therefore, there is no information in the coin toss. The entropy in this case is zero. Therefore, one binary outcome has the probability $\log_2 2 = 1$ bit. Similarly, one trit with equal probability contains $\log_2 3$ bits of information because it has one of three values.

Summary of the Concepts of Information Theory

Try to define an information function, *I*, in terms of an event, *i*, with probability p_i. How much information is acquired due to observation of event i?

- $I(p_i)$ is monotonically decreasing in p – an increase in the probability of an event decreases the information from an observed event
- $I(p) > 0$ – information is non-negative value
- $I(1) = 0$ – events that always occur do not communicate information
- $I(p1,p2)=I(p1)+I(p2)$ - information due to independent events is additive.

From the last condition it becomes clear that the proper choice of function to quantify information, preserving this additivity, is logarithmic

$I(p) = \log(1/p) = -\log(p)$

Finding the entropy

Now suppose we have a distribution where event i can happen with probability p_i. Suppose we have sampled it N times and outcome *i* was, accordingly, seen $n_i = Np_i$ times. The total amount of information we can receive is

$$\sum_i n_i I(p_i) = -\sum_i N p_i \log\,(p_i)$$

The average amount of information that we receive per event is therefore

$$-\sum_{\square} p_i \log(p_i)$$

Sources for further reading

Machine Learning in Action (chapters 2 & 3)

Peter Harrington

The Data Science Handbook (Chapters 8 & 19)

By Field Caddy

Machine Learning: A Probabilistic Perspective

By Kevin P. Murray