Information, Entropy, Cross-Entropy: ML perspective

Information

- Information is any entity or form that provides the answer to a question of some kind or resolves uncertainty.
- Information reduces uncertainty. The uncertainty of an event is measured by its probability of occurrence and is inversely proportional to that. The more uncertain an event, the more information is required to resolve uncertainty of that event.

Fundamental properties of Information

1. Less likely events should carry more information than more likely events, we want a monotonic function that decreases in value as the probability of an event increases
2. An event that is guaranteed to happen carries no information
3. Information is a non-negative quantity
4. Information due to independent event is additive $I(p_1 p_2) = I(p_1) + I(p_2)$

Self-Information

- To satisfy these criterion the proper choice of function is logarithmic, i.e.,

  $I(p) = \log\left(\frac{1}{p}\right) = -\log(p)$
- According to this definition information contained in two independent event is

  $I(p_1 p_2) = \log\left(\frac{1}{p1} * \frac{1}{p2}\right) = \log\left(\frac{1}{p1}\right) + \log\left(\frac{1}{p2}\right) = I(p_1) + I(p_2)$

  so it preserves the additive property of information. Its also obvious that it also preserves first three properties.

Entropy(Shannon Entropy)

- Average rate at which information is produced by a stochastic source of data
- Entropy is a measure of unpredictability of the state, or equivalently, of its average information content
- For a random variable X entropy is defined as the expected value of the information content of X:

  $H(X) = \mathbb{E}[I(X)] = \mathbb{E}\left[-\log(P(X))\right] = -\sum_i P(x_i)\log\left(P(x_i)\right)$
- The unit of entropy depends on base of the logarithm used. Common values of b are 2, Euler's number e, and 10, and the corresponding units of entropy are the 'bits'(or 'shanonns') for b = 2, 'nats' for b = e, and 'bans' (or 'dits) for b = 10.
- In the case of P(x_i)=0 the information content $[0\log(0)]$ is taken to be 0 which is consistent with limit $lim_{p\to 0^+}\ p\,log(p) = 0$
- ***Example Coin Toss:*** Consider tossing a coin with the probability of getting head P(H) is p and tail P(T) is (1-p) then the entropy of the process is:

  $H(X) = -\sum_i P(x_i)\log\left(P(x_i)\right) = -[P(H)\log(P(H)) + P(T)\log(P(T))$

  $= -[p\log(p) + (1-p)\log(1-p)]$

  $To\ obtain\ maximum\ amount\ of\ entropy\ we\ take\ derivative\ with\ respect\ to\ p$:

  $\frac{dH(x)}{dp} = -\left[p * \frac{1}{p} + \log(p) * 1 + \frac{(1-p)}{1-p} * (-1) + \log(1-p) * (-1)\right]$

  $= -[1 + \log(p) - 1 - \log(1-p)] = \log(1-p) - \log(p)$

$For\ maximum: \frac{dH(x)}{dp} = 0 \Rightarrow \log(1-p) - \log(p) = 0 \Rightarrow \log(1-p) = \log(p) \Rightarrow 1 - p = p$

$\Rightarrow 2p = 1 \Rightarrow p = \frac{1}{2}$

Which aligns with our intuition that when each event is equally likely, the uncertainty about an outcome is maximum and so is the entropy of the random variable.

And the maximum value is

$$H(X) = -\left[\frac{1}{2}\log\left(\frac{1}{2}\right) + \left(1-\frac{1}{2}\right)\log\left(1-\frac{1}{2}\right)\right] = -\left[\log\left(\frac{1}{2}\right)\right] = -[\log(1) - \log(2)] = 1\ bit$$

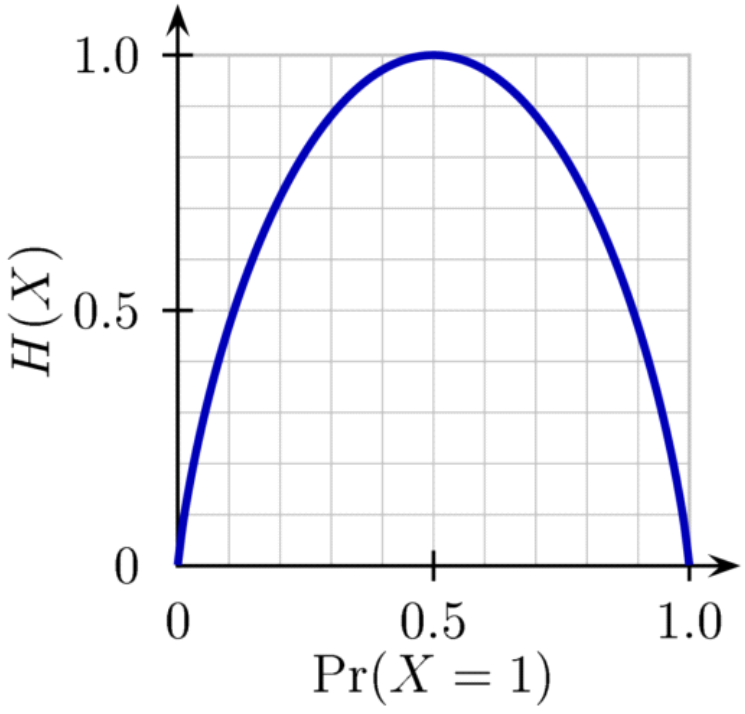
Which makes sense because we need 1 bit to encode the outcome for a single coin toss(let's say 1 for head, 0 for tail).

If we toss the coin n times there are 2^n possible outcomes. If the coin is fair $(p = 0.5)$ then the entropy is

$$H(X) = -\sum_{i=1}^{2^n} p^n\, log(p^n) = -\sum_{i=1}^{2^n} \left(\frac{1}{2}\right)^n log\left(\frac{1}{2}\right)^n = 2^n * \left(\frac{1}{2}\right)^n * log(2)^n = n$$

So it also checks out for n coin toss when we need n bit to represent an outcome.

Below is a graph for entropy of a coin flip where $Pr(X = 1)$ represents probability of getting head(or tail equivalently)



Communication system (Source coding) perspective of entropy

- Source coding from a sequence of symbols from an information source to a sequence of alphabet symbols (like bits) such that source symbols can be recovered exactly or with some distortions. Shannon's source coding theorem shows that is impossible to compress the data such that code rate(average number of bits per symbol) is less than the Shannon entropy of the source without information being lost. Or simply the optimal coding rate is *H(X)* bits per letter.
- One simple(but impractical) way to achieve this to assign $\log\left(\frac{1}{P(x_i)}\right)$ bits to the i^{th} symbol. Because in this case the average number of bits required is:

$$b = \sum_{i=1}^{N} P(x_i) b_i = \sum_{i=1}^{N} P(x_i) \log\left(\frac{1}{P(x_i)}\right) = -\sum_{i=1}^{N} P(x_i) \log(P(x_i)) = H(X)$$

Cross Entropy

- Cross entropy between two probability distributions *p* and *q* over the same underlying set of events measures the average number of bits needed to identify an event drawn from the set, if a coding scheme is used that is optimized for an artificial probability distribution *q* instead of true distribution *p*.
- It is defined as:

  $$H(p, q) = \mathbb{E}_p[I(q)] = \mathbb{E}_p\left[\log\left(\frac{1}{q(x_i)}\right)\right] = \sum_{x_i} p(x_i) \log\left(\frac{1}{q(x_i)}\right) = -\sum_{x_i} p(x_i) \log q(x_i)$$

- *Example Coin Toss:*

  Suppose we have a biased coin with $P_H = \frac{3}{4}$ *and* $P_T = \frac{1}{4}$. So true distribution is

  $$p = \begin{cases} P_H = 0.7 \\ P_T = 0.3 \end{cases}$$

  But if our predicted distribution is

  $$q = \begin{cases} P_H = 0.6 \\ P_T = 0.4 \end{cases}$$

  So entropy of the actual event or equivalently the average bit length(for optimized coding) is

  $$H(X) = -\sum_i P(x_i) \log p(x_i) = -[0.7 \log(0.7) + 0.3 \log(0.3)] = 0.8813$$

  But the cross entropy the distributions or average bit length(for optimized coding) when we assume distribution *q* is

  $$H(p, q) = -\sum_i p(x_i) \log q(x_i) = -[0.7 * \log(0.6) + 0.3 * \log(0.4)] = 0.9125$$

- **The entropy will always increase when we assume a wrong distribution.** Here's the proof for coin toss event:

  $p = \{P_H = m, 0 \le m \le 1$
  $q = \{P_H = n, 0 \le n \le 1$

  $$H(p, q) = -\sum_i p(x_i) \log q(x_i) = -[m \log(n) + (1 - m)\log(1 - n)]$$

  Partial derivative with respect to n (since we want to find optimum value for *n*):

  $$\frac{\partial H(p, q)}{\partial n} = -\left[m * \frac{1}{n} + (1 - m) * \frac{1}{1 - n} * (-1)\right] = -\left[\frac{m}{n} - \frac{1 - m}{1 - n}\right] = -\left[\frac{m - mn - n + mn}{n(1 - n)}\right]$$
  $$= \frac{n - m}{n(1 - n)}$$

  $\therefore n - m = 0 \Rightarrow n = m$

  To show that it's a minima we take second derivative with respect to n:

  $$\frac{\partial^2 H(p, q)}{\partial n^2} = -\left[m * \left(-\frac{1}{n^2}\right) - (1 - m) * \frac{-1}{(1 - n)^2} * (-1)\right] = \left[\frac{m}{n^2} + \frac{1 - m}{(1 - n)^2}\right] \ge 0$$

  It's because every term in the expression is non-negative.

  So it follows that $n = m$ is indeed a minima. Which in term means that every other probability distribution other than $n = m$ will result in a higher cross entropy.

Kullback–Leibler divergence(Relative Entropy)

- The Kullback–Leibler divergence is a measure of how a probability distribution is different from a second, reference probability distribution.

- For discrete probability distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$ defined on the same probability space, the KL divergence from **Q** to **P** is defined to be

  $$D_{KL}(P||Q) = -\sum_i P(i)\log\left(\frac{Q(i)}{P(i)}\right) = \sum_i P(i)\log(\frac{P(i)}{Q(i)})$$

- For distributions **P** and **Q** of a continuous random variable, the KL divergence is defined to be the integral

  $$D_{KL}(P||Q) = \int_x P(x)\log\frac{p(x)}{q(x)}dx \; where \; p \; and \; q \; probability \; ensities \; of \; P \; and \; Q$$

- A value of zero KL divergence indicates that the two probability distributions in question are similar.
- In the context of coding theory, $D_{KL}(P||Q)$can be construed as measuring the expected number of extra bits required to code samples from **P** using a code optimized for **Q** rather than the code optimized for **P**.
- *Relation with cross entropy:* $H(p,q) = H(p) + D_{KL}(P||Q)$
  So minimizing KL divergence is equivalent to minimizing cross entropy between the two probability distributions.

Likelihood

- Likelihood is a measure of the extent to which a sample provides support for particular values of a parameter in a parametric model. In probability we start with an assumed parameter and estimate the probability of a given sample. In statistics we start with the observation and make inference about our parameter.
- For a discrete random variable **X** with probability mass function *p* dependent on parameter θ the likelihood function is
  $\mathcal{L}(\theta|x) = p_\theta(x) = P_\theta(X = x) = P(X = x;\theta)$
  It should not be confused with conditional probability.
- *Example:* We toss a coin 10 times and end up with 4 heads. If we assumed the coin is fair i.e. $p_H = 0.5$ then the likelihood of the event is(it follows a binomial distribution):

  $$P\left(head_{count} = 4 \middle| p_H = 0.5\right) = \binom{n}{k}p^k(1-p)^{n-k} = \binom{10}{4} * 0.5^4 * (1-0.5)^{0.6} = 0.2051$$

  Or equivalently the likelihood that model parameter p_H equals to 0.5 is 0.2051.
  $\mathcal{L}\left(p_H = 0.5 \middle| head_{count} = 4\right) = 0.2051$
  For $p_H = 0.4$:

  $$P\left(head_{count} = 4 \middle| P_H = 0.4\right) = \binom{10}{4} * (0.4)^4 * (1-0.4)^6 = 0.2508$$

  $\mathcal{L}\left(p_H = 0.4 \middle| head_{count} = 4\right) = 0.2508$
  It should be obvious that this is the maximum likelihood obtainable in this case because it matches the observed data.

Log-Likelihood

- For many applications, the natural logarithm of the likelihood function, called the log-likelihood, is more convenient to work with. It's because the likelihood of several independent event is multiplicative and taking log makes them additive which is easier to work with(specially taking derivative).
- $l(\theta|x) = \log\,\mathcal{L}(\theta|x)$
- For n independent events:

  $$\mathcal{L}(\theta|x) = \prod_{i=1}^{n} P\left(x_i|\theta\right) = \prod_{i=1}^{n} \mathcal{L}(\theta|x_i)$$

  $$l(\theta|x) = \log\prod_{i=1}^{n} P\left(x_i|\theta\right) = \sum_{i=1}^{n}\log P\left(x_i|\theta\right) = \sum_{i=1}^{n}\log\mathcal{L}\left(\theta|x_i\right) = \sum_{i=1}^{n} l(\theta|x_i)$$

Motivation for using cross entropy loss in machine learning

- Cross entropy is a measure of how our assumed probability distribution is similar to the true probability distribution. The lower the value the less is the entropy/uncertainty between the two distributions. So we want to minimize cross entropy loss.
- For a classification task suppose we have *n* training samples for *m* class. Then the negative log likelihood over the entire training data is

  $$NLL = -\sum_{i=1}^{n} l(\theta|\boldsymbol{y}_i) \; where \; \boldsymbol{y_i} \; is \; the \; ground \; truth \; vector \; of \; the \; i^{th} \; sample$$

  Lets say for a particular example we have ground truth vector $\boldsymbol{y_i} = [0, 0, 1, 0]^T$ and the prediction vector is $\widehat{\boldsymbol{y_i}} = [0.1, 0.3, 0.4, 0.2]^T$ then what is the likelihood of model parameters being θ for this event? Well it's just the probability the model predicted for the correct outcome which is 0.4. So we can express the likelihood of a particular example in the following way

  $$\mathcal{L}(\theta|\boldsymbol{y_i}) = \hat{y}_i^k \; where \; k \; indicates \; the \; ground \; truth \; class$$

  $$l(\theta|\boldsymbol{y_i}) = \log \hat{y}_i^k = \sum_{j=1}^{m} y_i^j \log \hat{y}_i^j \; where \; each \; trem \; is \; zero \; except \; when \; j = k \; (y_j^k = 1)$$

  $$\therefore NLL = -\sum_{i=1}^{n}\sum_{j=1}^{m} y_i^j \log \hat{y}_i^j = \sum_{i=1}^{n}\left(-\sum_{j=1}^{m} y_i^j \log \hat{y}_i^j\right) = \sum_{i=1}^{n} H(\boldsymbol{y}_i, \widehat{\boldsymbol{y}_i}) \; which \; is \; total \; cross \; entropy.$$

  So it turns out minimizing negative log likelihood(or maximizing likelihood) is the same as minimizing cross entropy loss over the training dataset.
- The gradient of a cost function needs to be large and predictable enough to serve as a good guide for the learning algorithm. Functions that saturate (become very flat) undermine this objective because they make the gradient become very small. In many cases this happens because the activation functions used to produce the output of the hidden units or the output units saturate. The negative log-likelihood helps to avoid this problem for many models. Many output units involve an exp function that can saturate when its argument is very negative. The log function in the negative log-likelihood cost function undoes the exp of some output units. Some other cost functions like mean squared error produce very small gradients on saturated outputs.

References

1. https://en.wikipedia.org/wiki/Information_theory
2. https://en.wikipedia.org/wiki/Entropy_(information_theory)
3. https://en.wikipedia.org/wiki/Cross_entropy
4. https://en.wikipedia.org/wiki/Shannon's_source_coding_theorem
5. https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
6. https://en.wikipedia.org/wiki/Likelihood_function
7. https://stats.stackexchange.com/questions/2641/what-is-the-difference-between-likelihood-and-probability
8. https://en.wikipedia.org/wiki/Binomial_distribution
9. https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/
10. https://www.deeplearningbook.org/