Introduction to XEntropy for Machine Learning

J Hope

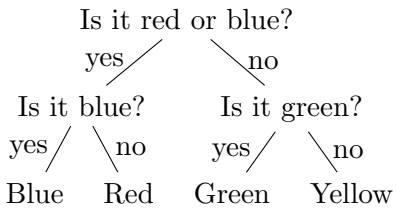
July 25, 2017

Introduction

Entropy and Cross Entropy is used extensively in machine learning to evaluate the effectiveness of a given strategy under different situations. Cross entropy is, for instance, used in classification algorithms including neural networks specifically as a cost function, for the purposes of back propagation and model training.

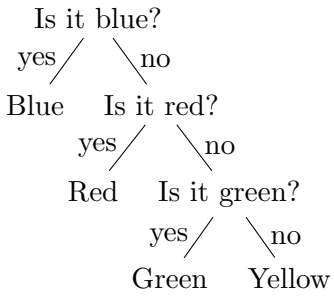
Entropy explained

Entropy can be explained in the context of a simple game. The game works as follows. A player draws a ball from a bag of balls which contains: a blue ball, a red ball, a green ball, and an yellow ball. The objective of the game is to guess the correct colour drawn from the bag, with as fewer guesses or 'questions' as possible. One strategy, is described by the tree below.



Under this strategy, each ball has a $1/4$ of probability of getting chosen ($1/2 \times 1/2$), and takes 2 questions to guess the colour of the ball drawn. So the expected number of questions to guess the ball is 2.

Let's consider a new game. Ball are now drawn from a different bag: $1/2$ of them are blue, $1/4$ are red, $1/8$ are green, and $1/8$ are yellow. The strategy that yields the fewest questions - the optimal strategy - is described by the tree below.

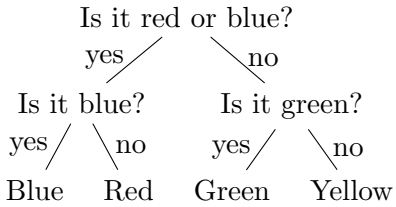


1/2 of the time it is blue, and it takes 1 question to correctly guess the colour of the ball drawn. 1/4 of the time it is red, and it takes 2 questions to correctly guess the colour of the ball drawn. The expected (mean) number of questions to guess a ball is calculated as 1/2 × 1 question (blue) + 1/4 × 2 question (red) + 1/8 × 3 questions (green) + 1/8 × 3 questions (yellow) = 1.75.

When we consider the minimum number of questions that must be asked under varying ball probabilities, a pattern emerges. When p = 1, it takes 0 questions to guess the colour of my ball, or $\log_2(1)$ questions. When p = 1/2, it takes 1 question to guess the colour of my ball, or $\log_2(2)$ questions. In general terms, a ball with p probability takes $\log_2\left(\frac{1}{p}\right)$ to guess its colour correctly. When p = 1/4, $\log_2(1) = 2\log_2(4) = 2$ questions. The expected (mean) number of questions for this setup is $\sum_i p_i \log_2\left(\frac{1}{p_i}\right)$. And that is the expression for entropy. Intuitively, it is the expected number of questions to guess the colour under the optimal setup. The more uncertain the setup is, the higher the entropy.

XEntropy explained

Cross entropy (often written as XEntropy) is a related idea. Consider again the bag where 1/2 of the balls are blue, 1/4 are red, 1/8 are green, and 1/8 are yellow, under the first strategy (shown again below):



Under this strategy, 1/8 of the time, the ball is yellow, and it takes 2 questions to get it right. 1/2 of the time, it?s blue but it still takes 2 questions to guess the correct colour. On average, it takes 1/8 × 2 + 1/4 × 2 + 1/2 × 2 = 2 questions to guess the correct colour. So, 2 is the cross entropy for this strategy.

Cross entropy for a given strategy is simply the expected number of questions to guess the colour

under that strategy. For a given setup, the better the strategy is, the lower the cross entropy. The lowest cross entropy is that of the optimal strategy, which is just the entropy defined above. This is why in machine learning classification problems, we want to minimize the cross entropy.

More formally, cross entropy is $\sum_i p_i \log_2 \left(\frac{1}{\hat{p}_i}\right)$, where p_i is the true probability (for example, 1/8 for yellow and green, 1/4 for red and 1/2 for blue) and $\hat{p}_i$ is the wrongly assumed or predicted probability (for example, using the first strategy, we are assuming or predicting p = 1/4 for all colours). It may be easy to confuse whether it's p_i or $\hat{p}_i$ that is in the log. The logarithm is used to calculate the number of questions under the strategy you are applying; in other words, what is under the log is your predicted probability, $\hat{p}_i$.

XEntropy in Machine Learning context

Returning to Softmax Layers in neural networks, the idea is quite simple. Given any instance x, a score $s_k(x)$ for each class k is computed as $s_k(x) = \theta_k^T \cdot x$, where T represents the transposition and theta represent the model parameters. The probability $\hat{p_k}$ that the instance belongs to a class k is then computed as

$$\hat{p_k} = \sigma(s(x))_k = \frac{exp(s_k(x))}{\sum_{j=1}^{K} exp(s_j(x))}$$

Note that

- K is the number of classes
- s(x) is a vector containing the scores of each class for the instance x
- $\sigma(s(x))_k$ is the estimated probability that the instance x belongs to the class k given the scores of each class for that instance.

For the purposes of training in supervised machine learning, cross entropy is used to measure how well a set of estimated probabilities match the target class. The cross entropy, expressed as a cost function is given as:

$$J(\phi) = -\frac{1}{m_i} \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^i log(\hat{p}_k^i)$$

Note that

- m in the number of instances
- y_k^i is the target class for the i^{th} instance for class k.
- $-\frac{1}{m_i}$ is simply saying for all instances m, and is expressed negatively since it is a function of cost
- otherwise, cross entropy is expressed as given above.

References

[1] Intuitive explanations of technical concepts
`https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy`

[2] Hands-On Machine Learning with SciKit-Learn & Tensorlow Aurenlien Geron, Oreilly.