# IN THIS NOTE,

- Cross-Entropy
- Log-Loss
- MLE

Information for a discrete event = Measuring information = how surprising/ unsurprising an event

Low Probability = rare event = surprising = More information ( High Entropy ??)

High Probability = common event = unsurprising = Less information ( Low Entropy ??)

## Shannon Information = To measure information

= information (x) = - log( p(x)) = h(x)

- log() base 2 due to binary bits
- base e ( Euler's number) can be used and the unit is nats
- negative sign ensures the result in positive or zero.

Information for a random variable = Information for an event

## Shannon Entropy = Entropy = H(X) = - sum ( each k in K p(k) * log(p(k))) = -sum( p * log(p))

- the average info is the same as the lower bound on info as all outcomes are equally likely esp uniform distribution eg. A roll of a fair dice

**The intuition of Entropy** : the average number of bits required to represent of transmit an event drawn from the probability distribution for the random variable

## CROSS ENTROPY

The average number of bits needed to encode data coming from a source with distribution p when we use model q

We have the target distribution P and the approximation of target distribution Q and the cross entropy of Q from P is the average number of total bits required to represent an event using Q instead of P.

Cross Entropy = difference between two probability distributions

= the average number of TOTAL BITS required to represent a message

= can be used as loss function when optimizing

= H( P, Q) = - sum x in X P(x) * log( Q(x))

*We can use this for continuous probability distributions using the integral across the events instead of the sum.*

KL Divergence = Relative Entropy between two probability distributions

= the average number of EXTRA BITS required to represent a message ( coz we used Q rather P)

KL ( P || Q ) = - sum x in X P(x) * log ( Q(x) / P(x))

H ( P, Q ) = H(P) + KL ( P||Q)

H ( P, Q) != H( Q, P) , H( P, P) = H(P) , H( Q, Q) = H (Q)

Therefore, H ( P, Q ) = H(P) + KL ( P||Q) = - sum x in X P(x) * log( Q(x))

*Using the cross —entropy error function rather than the sum of squares for a classification problem provides faster training and better generalization*

---

*How about we try a random variable with a probability distribution with idea of the classification example ?!*

**Random Variable** : *The example for which we require a predicted class label*

**Events** : *Each class label that could be predicted*

*We know the target probability distribution P for an input as the class label 0 or 1 ( "impossible" or "certain").*

*But these probabilities have no surprise at all, and hence no information content or zero entropy there is.*

*For language Classification,*

**Expected Probability** **( y ):** The known probability of each class label for an example in the dataset ( P )

**Predicted Probability** **( yhat )** : The probability of each class label for an example predicted by the model ( Q )

*To calculate the cross-entropy for a single prediction,*

H ( P, Q ) = - sum x in X P(X) * log ( Q (x) )

Each x in X is a class label that could be assigned to the example

P(x) will be 1 for known label and 0 for all other labels.

For binary classification task, the cross-entropy for a single example is as follows:

H ( P, Q) = - ( P(class0) * log( Q(class0)) + P(class1) * log(Q(class1)))

According Bernoulli distribution for the positive class label,

Predicted P(class0) = 1 – yhat, Predicted P( class1) = yhat

We minimize the cross-entropy for the model across the entire training dataset which can be done by calculating the average cross-entropy across all training examples.

Therefore, known class label; the entropy of this variable -> log (1) = 0

ZERO Entropy = zero loss means that the predicted class probabilities are identical to probabilities in the training dataset.

HOWEVER, this means that the model is overfitting the training dataset.

A good cross-entropy score ?

H ( P, Q ) < 0.2 is a good start.

H (P, Q) < 0.1 or 0.05 is better.

Cross-entropy is not log loss, which is the Negative Log Likelihood. ( NLL)

For classification problems, " log loss", "cross-entropy" and "negative log-likelihood" are used interchangeably. Calculation is done in the same way.

???

"Any loss consisting of a negative log-likelihood is a cross-entropy between the empirical distribution defined by the training set and the probability distribution defined by model. For example, mean squared error is the cross-entropy between the empirical distribution and a Gaussian model." – — Page 132, Deep Learning, 2016.

---

Logistic regression is a model( a classical linear method) used in binary classification.

We use maximum likelihood estimation to give good parameters of a logistic regression model.

Linear Regression = fits the line to the data to predict a new quantity

Logistic Regression = fits a line to best separate the two classes

Input= X with n examples

Output = yhat

yhat = model ( X )

given input = weighted sum of the inputs for the example and the coefficient symbolized as beta ( intercept or bias )

yhat = beta0 + beta1 * x1 + beta2 * x2 + .....+ betan * xn

y = X * Beta

But this gives the real value instead of a class label and so we need to squash it to become a value between 0 and 1.

f (x) = 1 / ( 1 + exp (-x))

This is linear regression.

In case of logistic regression, x is replaced with the weighted sum.

Therefore,

yhat = 1 / ( 1 + exp ( - (X * Beta)))

**"The output being a number between 0 and 1, can be interpreted as a probability of belonging to the class labeled 1." –** — Page 726, Artificial Intelligence: A Modern Approach, 3rd edition, 2009.

TO estimate the parameters,

- Least Squares Optimization ( iteratively reweighted least squares)
- Maximum Likelihood Estimation

can be used.

Both are optimization procedures that involve searching for different model parameters.

log-odds = beta0 + beta1 * x1 + beta2 * x2 + .....+ betan * xn = yhat = predicted values

**What are odds and log-odds ???**

Odds came from the field of gambling.

Odds = (wins : losses )

Odds of success = p / ( 1 —p)

Log-odds = log ( p / (1 — p ))

Log-odds = beta0 + beta1 * x1 + beta2 * x2 + ….. + betan * xn

Odds = exp(log-odds)

Odds = exp (beta0 + beta1 * x1 + beta2 * x2 + ….. + betan * xn)

Odds — p * odds = p

Odds = p + p * odds

Odds = p (odds + 1)

P = odds / (odds + 1)

p = exp(log-odds) / ( exp(log-odds) + 1)
p = 1/ ( (exp(log-odds) + 1) /exp(log-odds))
p = 1 / (1+ 1/ exp(log-odds))
p = 1 / ( 1+ exp(- log-odds )) , in case; 1/exp(x) =exp(-x)

All in all,
Cross-entropy or log-loss measures the performance of a classification model whose output is a probability value between 0 and 1.
Cross-entropy loss increases as the predicted probability diverges from the actual label.
Log-loss increases rapidly as the predicted probability decreases.
Although both are slightly different depending on the context, but in machine learning, when calculating error rates between 0 and 1 they resolve to the same thing.

## MAXIMUM LIKELIHOOD ESTIMATION OR MLE

Aim : To maximize the conditional probability of observing the data ( X) given a specific probability distribution and its parameters ( theta)

P ( X ; theta) = P ( x l, x2, x3, ….., xn ; theta)
= L ( X; theta)

L() is the likelihood function.

To get much more stability, the above equation is fixed as follows:

$$\text{sum i to n log ( P (xi ; theta))}$$

Now it becomes log-likelihood function from likelihood function.

It is used in optimization problems to prefer to minimize the cost function rather than to maximize it.

Therefore, the negative of the log-likelihood function is use, referred to generally as a Negative Log-Likelihood(NLL) function.

$$\text{Minimize } -\text{sum i to n log ( P (xi; theta))}$$

Let's see how we use this in ML example.

A modeling hypothesis h includes

- The choice of model
- The model parameters

Finding the modeling hypothesis means maximizing the likelihood function.

$$\text{Maximize } -\text{sum i to n log ( P (xi; theta))}$$

In supervised learning.

P( y | x ) = probability of the output given the input

As such,

$$\text{Maximize } -\text{sum i to n log ( P (yi | xi ; h))}$$

*h is replaced with our logistic model and we use the Bernoulli distribution as a single parameter.*

*The probability of a successful outcome (p).*

*P(y = 1 ) = p*

*P(y = 0 ) = 1-p*

The expected value (mean) of the Bernoulli distribution is as follows:

$$mean = P(y = 1) * 1 + P(y = 0) * 0$$

$$= p * 1 + (1 - p) * 0$$

$$Likelihood = yhat * y + (1 - yhat) * (1 - y)$$

Coz the predicted value from the model deals with the target value- yhat deals with y.

```
y=1.0, yhat=0.9, likelihood: 0.900
y=1.0, yhat=0.1, likelihood: 0.100
y=0.0, yhat=0.1, likelihood: 0.900
y=0.0, yhat=0.9, likelihood: 0.100
```

This means that when yhat is more likely to become y = 1 or yhat is less likely to become y = 0, the likelihood is higher: the model and its parameters are great!

We can update the likelihood function using the log to transform it into a log-likelihood function.

$$Log\text{-}Likelihood = log(yhat) * y + log(1 - yhat) * (1 - y)$$

Then, we can sum the likelihood function across all examples in the dataset to maximize the likelihood:

$$Maximize \sum_{i}^{n} log(yhat\_i) * y\_i + log(1 - yhat\_i) * (1 - y\_i)$$

To minimize the cost function, we can change the above equation as negative log-likelihood and minimize it.

$$Minimize \sum_{i}^{n} -(log(yhat\_i)) * y\_i + log(1 - yhat\_i) * (1 - y\_i)$$

Calculating the negative of the log-likelihood function for the Bernoulli distribution is equal to calculating the cross-entropy function for the Bernoulli distribution, where p() represents the probability of class 0 or class 1, and q() represents the estimation of the probability distribution, in this case by our logistic regression model.

Cross entropy = - ( log (q(class0)) * p(clas0) + log(q(class1)) * p(class1))

Unlike linear regression, we can't write in down the MLE in the closed form. Instead, an iterative optimization algorithm must be used. For optimization, we need to derive the gradient and Hessian.

To conclude,

Cross Entropy ( log loss ) is much better than MSE. We use cross entropy to see the error and calculate the gradient decent to update the parameters of weights and biases. Then we measure the cross entropy again to know if it's minimized.

For more details, you better read these links.

https://towardsdatascience.com/why-and-how-to-use-cross-entropy-4e983cbdd873

https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

References :
https://machinelearningmastery.com/what-is-information-entropy/
https://machinelearningmastery.com/cross-entropy-for-machine-learning/
https://machinelearningmastery.com/logistic-regression-with-maximum-likelihood-estimation/
https://ml-cheatsheet.readthedocs.io/