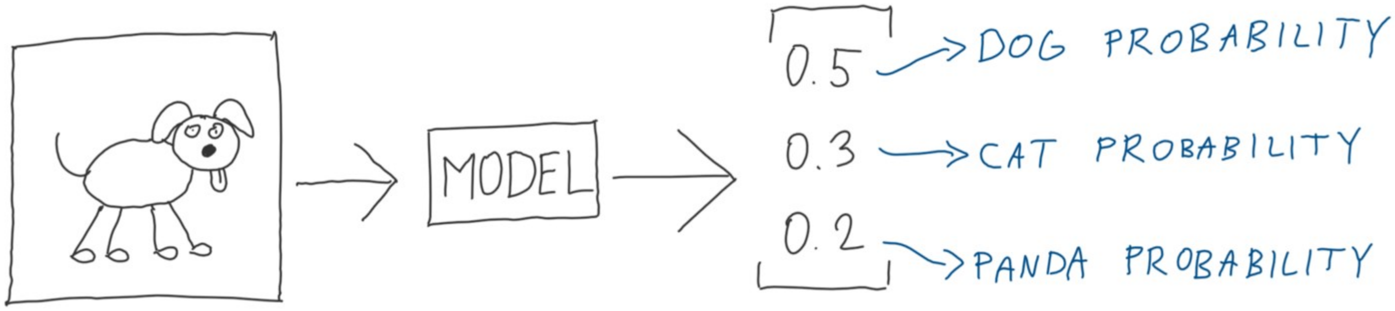
Regression Algorithm	**Classification Algorithm**
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.
The regression Algorithm can be further divided into Linear and Non-linear Regression.	The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.

Cross-entropy is a commonly used loss function for classification tasks. Let's see why and where to use it. We'll start with a typical multi-class classification task.

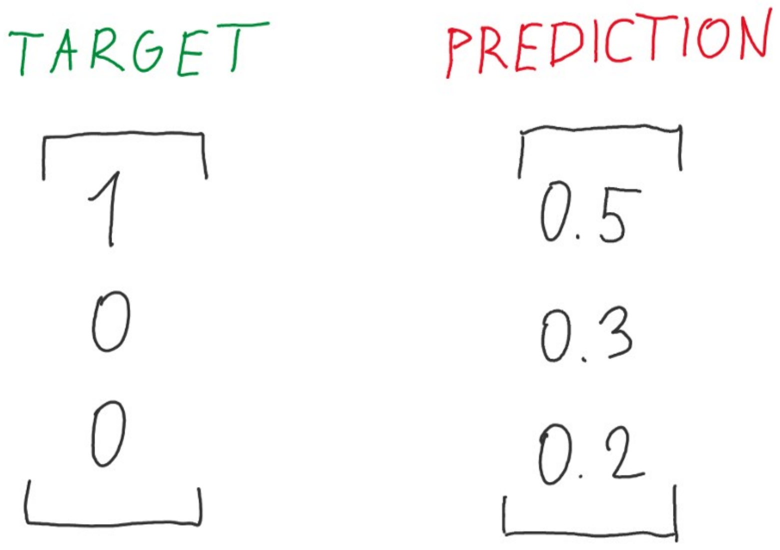
Multi-class classification

Which class is on the image — dog, cat, or panda? It can only be one of them. Let's have an image of a dog.



The prediction is a **probability vector**, meaning it represents predicted probabilities of all classes, summing up to 1.

In a neural network, you typically achieve this prediction by having the last layer activated by a softmax function, but anything goes — it just must be a probability vector.



Let’s compute the cross-entropy loss for this image.

Loss is a measure of performance of a model. The lower, the better. When learning, the model aims to get the lowest loss possible.

The target represents probabilities for all classes — dog, cat, and panda.

The target for multi-class classification is a one-hot vector, meaning it has 1 on a single position and 0’s everywhere else.

For the dog class, we want the probability to be 1. For other classes, we want it to be 0.

We will start by calculating the **loss for each class** separately and then summing them. The loss for each separate class is computed like this:

$$\text{Loss for class } X = -\underbrace{p(X)}_{\substack{\text{probability}\\\text{of class } X\\\text{in TARGET}}} \cdot \log \underbrace{q(X)}_{\substack{\text{probability}\\\text{of class } X\\\text{in PREDICTION}}}$$

Don’t worry too much about the formula, we’ll cover that in a second. Just notice that if the class probability is 0 in the target, the loss for it is also 0.

$$\begin{aligned}\text{Loss for CAT} &= -p(\text{CAT}) \cdot \log q(\text{CAT}) \\ &= -0 \cdot \log q(\text{CAT}) \\ &= 0\end{aligned}$$

$$\begin{aligned}\text{Loss for PANDA} &= -p(\text{PANDA}) \cdot \log q(\text{PANDA}) \\ &= -0 \cdot \log q(\text{PANDA}) \\ &= 0\end{aligned}$$

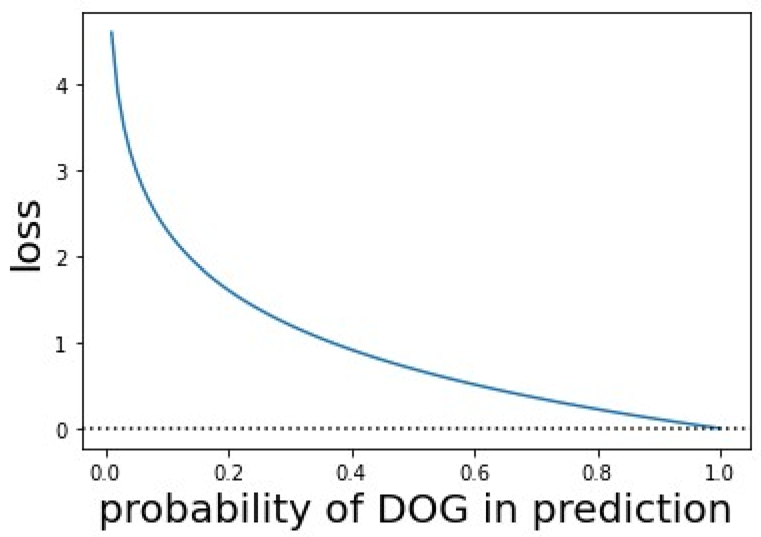
$$\begin{array}{cccl} & \text{TARGET} & \text{PREDICTION} & \\ \text{DOG} & 1 & 0.5 & \rightarrow \text{LOSS IS ?} \\ \text{CAT} & 0 & 0.3 & \rightarrow \text{LOSS IS } 0 \\ \text{PANDA} & 0 & 0.2 & \rightarrow \text{LOSS IS } 0 \end{array}$$

And lastly — the loss for the dog class:

$$\begin{aligned} \text{Loss for DOG} &= -p(\text{DOG}) \cdot \log q(\text{DOG}) \\ &= -1 \cdot \log 0.5 \\ &= 0.693\ldots \end{aligned}$$

What does that number mean?

Let’s see how would the loss behave if the predicted probability was different:



- The loss is 0 when the prediction is 1 (the same as the target).
- The loss is infinity if the prediction is 0 (the complete opposite of our target).
- We will never predict something less than 0 or more than 1, so we don't have to worry about that.

What if we predict something in the middle?

The loss gets steeper, the further away from the target we get.
You can think of it as a similar concept to square error — the further away we are from the target, the faster the error grows.

Why is the loss 0 for the cat and panda classes?

It looks like we are rewarding the model with low loss, even if it predicts a high probability for a class that is not present in the image.

We don't mind if the model predicts that there is a cat with an 80% probability if there is none because then it has only 20% left to predict the correct class. There, the loss will be that much bigger. In other words — we don't care on which classes the model wastes the predicted probabilities, only how correctly it identifies the only present class.

The total loss for this image is the sum of losses for each class.

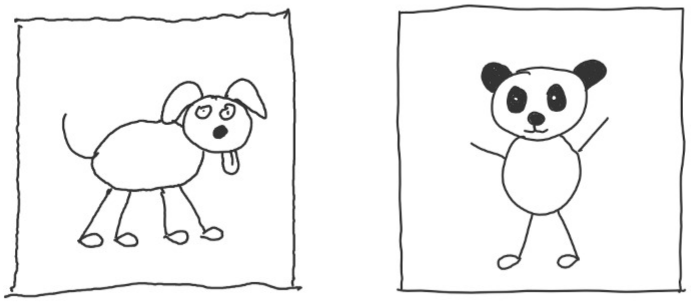
$$\begin{aligned}\text{Cross-entropy} &= \text{Loss for DOG} + \text{Loss for CAT} + \text{Loss for PANDA}\\ &= 0.693 \quad + 0 \quad + 0\\ &= 0.693\end{aligned}$$

It can be formulated as a sum over all classes.

$$\text{Cross-entropy} = -\sum_{x} p(x) \cdot \log q(x)$$

This is the **cross-entropy** formula that can be used as a loss function for any two probability vectors. That is our loss for 1 image — the image of a dog we showed at the beginning. If we wanted the loss for our batch or the whole dataset, we would just sum up the losses of the individual images.

Suppose we have 2 different models giving the following predictions:



MODEL A:	100% DOG	10% PANDA
MODEL B:	90% DOG	20% PANDA

In the eyes of cross-entropy, model B is better — it has a lower cross-entropy loss. If you can see why — well done! Have a panda.

Training models by punishing big mistakes a lot more than small mistakes turned out to be a good idea in machine learning.

Why sum up over all the classes if the loss for most of them is 0?
If our target is a one-hot vector, we can indeed forget targets and predictions for all the other classes and compute only the loss for the hot class. This is the negative natural logarithm of our prediction.

$$\text{Categorical Cross-entropy} = -\log q(x)$$

x is the class that is 1 in our TARGET

This is called **categorical cross-entropy** — a special case of cross-entropy, where our target is a one-hot vector.

The thing is — the cross-entropy loss works even for distributions that are not one-hot vectors.
The loss would work even for this task:

$$\begin{array}{cc} \text{TARGET} & \text{PREDICTION} \\ \begin{bmatrix} 0.5 \\ 0.1 \\ 0.4 \end{bmatrix} & \begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix} \end{array}$$

With the cross-entropy, we would still be able to compute the loss, and it would be minimal if all the classes would be correct, and still have the property of punishing bigger mistakes much more.

In our one-hot target example, the entropy was conveniently 0, so the minimal loss was 0. If your target is a probability vector that is **not** one-hot, entropy (minimal loss) will be bigger than 0, but you can still use the cross-entropy loss just fine.

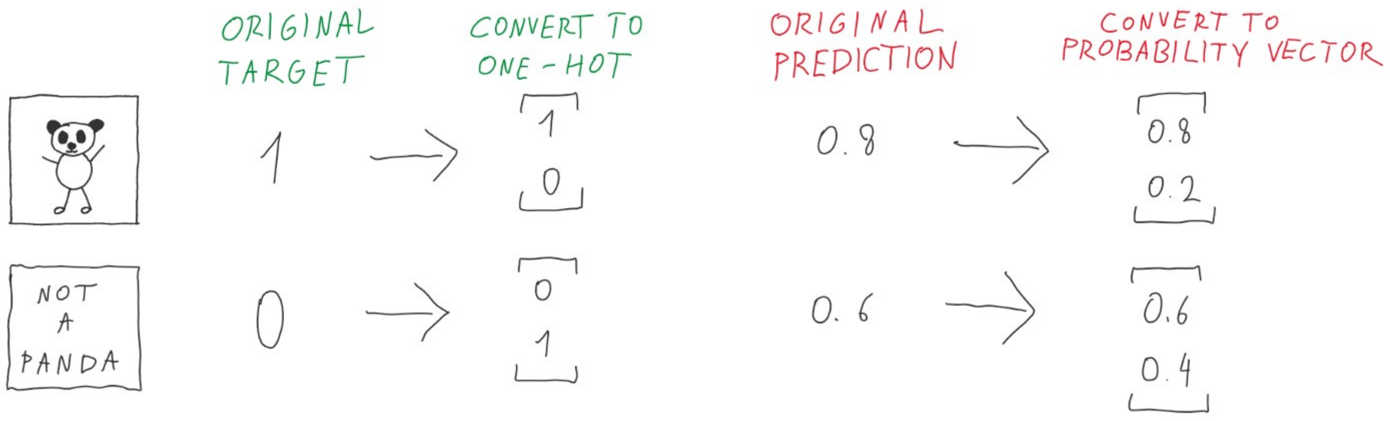
If you're more curious about what entropy is, I recommend watching this video.

Binary classification

Binary cross-entropy is another special case of cross-entropy — used if our target is either 0 or 1. In a neural network, you typically achieve this prediction by sigmoid activation.

The target is **not** a probability vector. We can still use cross-entropy with a little trick.

We want to predict whether the image contains a panda or not.



This is the same as if we would convert the target to a one-hot vector and our prediction to a probability vector — the probability of panda would be the same as the prediction and probability of not-a-panda would be 1-prediction. In other words, if we predict 0.6, that means we are saying that it's 60% a panda and 40% not-a-panda.

This loss can be computed with the cross-entropy function since we are now comparing just two probability vectors or even with categorical cross-entropy since our target is a one-hot vector. It can also be computed without the conversion with a **binary cross-entropy**.

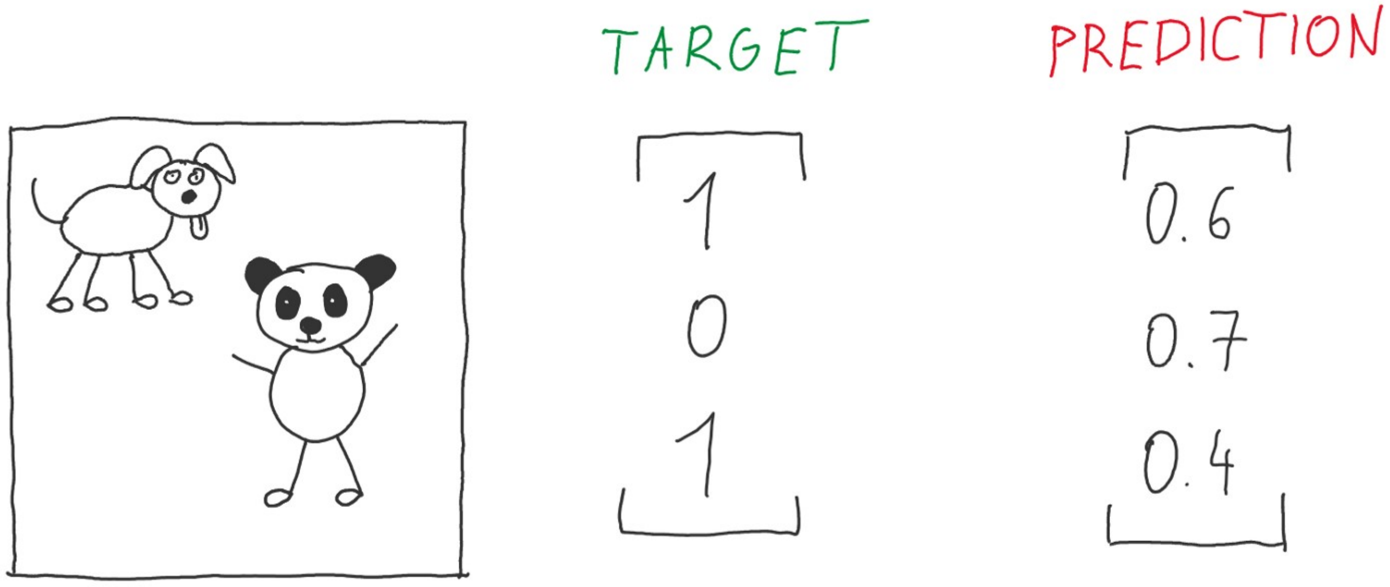
$$\text{Binary Cross-entropy} = -\Big(\underbrace{p(x)\cdot\log q(x)}_{\substack{\text{This cancels out}\\ \text{if the target is } 0}} + \underbrace{(1-p(x))\cdot\log(1-q(x))}_{\substack{\text{This cancels out}\\ \text{if the target is } 1}}\Big)$$

We are just applying the natural logarithm to the difference between our prediction and our target.

And that’s all there is to binary cross-entropy.

Multi-label classification

Cross-entropy can also be used as a loss function for a multi-label problem with this simple trick:



Notice our target and prediction are **not** a probability vector. It's possible that there are all classes in the image, as well as none of them. In a neural network, you typically achieve this by sigmoid activation.

We can look at this problem as multiple binary classification subtasks. Let's say we want to only predict if there is a dog or not.

We know that our target is 1 and we have predicted 0.6

We will compute the binary cross-entropy for this subtask:

$$\begin{aligned}\text{Binary Cross-entropy}_{DOG} &= -\left(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\right)\\ &= -\left(1\cdot\log 0.6 + (1-1)\cdot\log(1-0.6)\right)\\ &= -\left(\log 0.6 + 0\right)\\ &= 0.510\ldots\end{aligned}$$

And do the same for the other classes.

For a cat, our target is 0, so the other part of binary cross-entropy cancels out:

$$\begin{aligned}\text{Binary Cross-entropy}_{CAT} &= -\left(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\right)\\ &= -\left(0\cdot\log 0.7 + (1-0)\cdot\log(1-0.7)\right)\\ &= -\left(0 + \log 0.3\right)\\ &= 1.20\ldots\end{aligned}$$

$$\begin{aligned}\text{Binary Cross-entropy}_{PANDA} &= -\left(p(x)\cdot\log q(x) + (1-p(x))\cdot\log(1-q(x))\right)\\ &= -\left(1\cdot\log 0.4 + (1-1)\cdot\log(1-0.4)\right)\\ &= -\left(\log 0.4 + 0\right)\\ &= 0.916\ldots\end{aligned}$$

And sum up the losses for each subtask:

$$\begin{aligned}\text{Total Loss} &= \text{Binary Cross-entropy}_{DOG}\\ &+\text{Binary Cross-entropy}_{CAT}\\ &+\text{Binary Cross-entropy}_{PANDA}\\ &= 2.631\ldots\end{aligned}$$

That’s all there is to the cross-entropy loss for multi-label classification.

Conclusion/TL;DR

Cross-entropy — the general formula, used for calculating loss among two probability vectors. The more we are away from our target, the more the error grows — similar idea to square error.

$$\text{Cross-entropy} = -\sum_{x} p(x)\cdot\log q(x)$$

Multi-class classification — we use multi-class cross-entropy — a specific case of cross-entropy where the target is a one-hot encoded vector. It can be computed with the cross-entropy formula but can be simplified.

$$\text{Categorical Cross-entropy} = -\log q(\underbrace{x}_{\text{x is the class that is 1 in our TARGET}})$$

Binary classification — we use binary cross-entropy — a specific case of cross-entropy where our target is 0 or 1. It can be computed with the cross-entropy formula if we convert the target to a one-hot vector like [0,1] or [1,0] and the predictions respectively. We can compute it even without this conversion, with the simplified formula.

$$\text{Binary Cross-entropy} = -\Big(\underbrace{p(x)\cdot\log q(x)}_{\text{This cancels out if the target is 0}} + \underbrace{(1-p(x))\cdot\log(1-q(x))}_{\text{This cancels out if the target is 1}}\Big)$$

Multi-label classification — Our target can represent multiple (or even zero) classes at once. We compute the binary cross-entropy for each class separately and then sum them up for the complete loss.

$$\text{Total Loss} = \sum_{x} \text{Binary Cross-entropy}_x$$