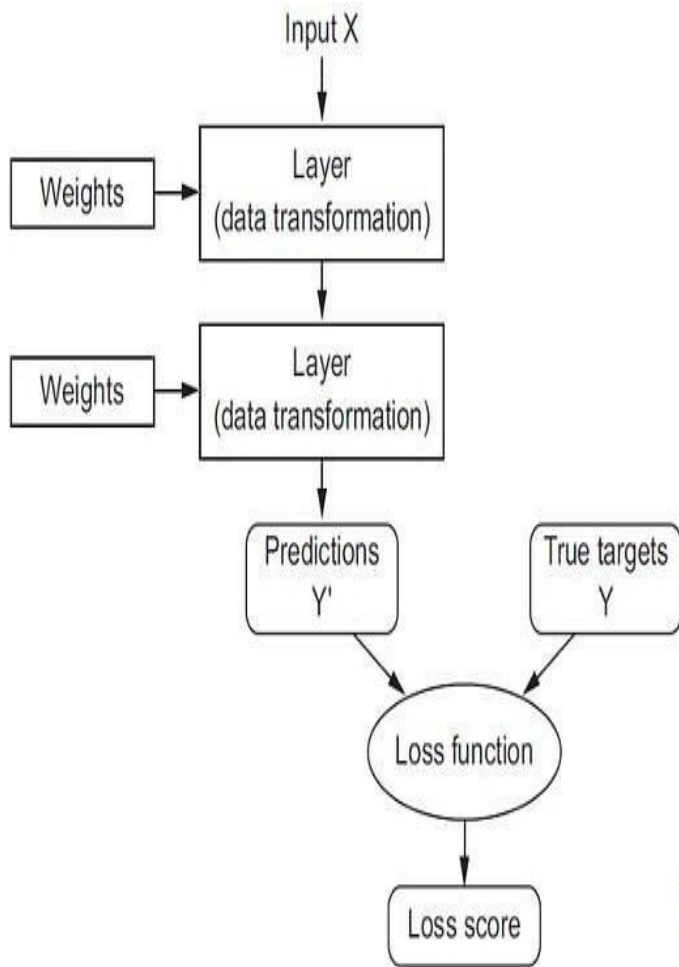


Loss functions



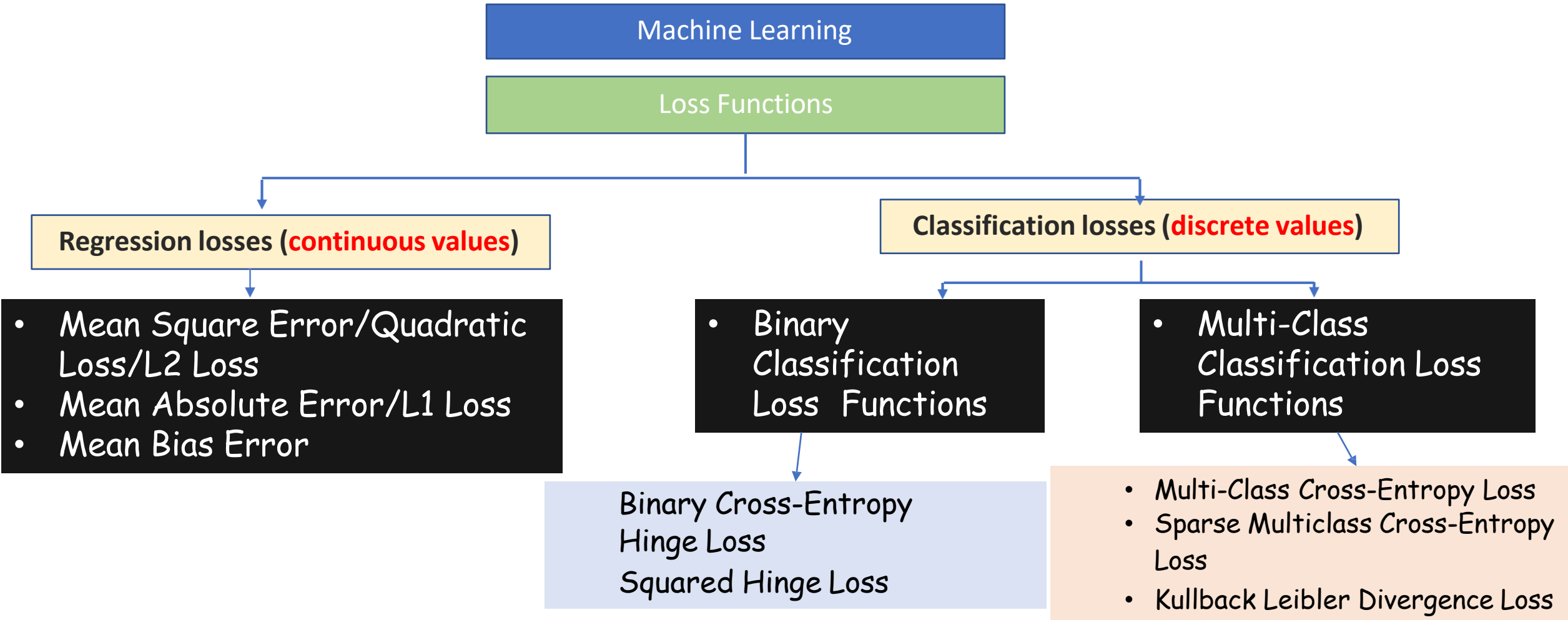
Loss Function



- A function that calculates loss for 1 data point is called the **loss function**.
- A loss function is a **measure of how good your prediction** model does in terms of being **able to predict the expected outcome**(or value).
- The loss function is the function that **computes** the **distance between the current output** of the algorithm and **the expected output**.
- An optimization problem, **define a loss function** and then optimize the algorithm to **minimize the loss function**.

Loss Function

- There's no **one-size-fits-all loss function** to algorithms in machine learning.



Regression Losses

Mean Square Error/Quadratic Loss/L2 Loss

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

- Mean square error is calculated by taking the average, specifically the mean, of errors squared from **data** as it relates to a function.
- A smaller MSE is preferred because it indicates that your data points are dispersed closely around its central moment (**mean**).
- **Sensitive to outliers**, punishes larger errors more.
- Due to **squaring**, **predictions** that are **far away** from **actual values** are **penalized heavily** in comparison to less deviated predictions

Regression Losses

Mean Absolute Error/L1 Loss

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

- Mean absolute error, on the other hand, is measured as the average of the sum of absolute differences between predictions and actual observations.
- Like MSE, this as well measures the magnitude of error without considering their direction. Unlike MSE, MAE needs more complicated tools such as linear programming to compute the gradients.
- Plus MAE is more robust to outliers since it does not make use of square.

Classification losses (discrete values)

Hinge Loss/Multi class SVM Loss

$$SVM Loss = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

- **Hinge Loss** trains **data classifiers**, especially in **Support Vector Machines (SVM)**.
- These are used for **training the classifiers**.
- If there's a negative distance from the classification boundary, it means the boundary is wrong and the classifier would be incorrectly justified.
- The difference between **Log Loss** and **Hinge Loss** is in the **estimation of probabilities**.

Scoring function as

Score vector

$$s = f(x_i, W)$$

Score for j-th class

$$s_j = f(x_i, W)_j$$

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector

$S = f(x_i, W)$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$
i.e. $s_j - s_{y_i} + 1 \leq 0$

If true, loss is 0

If false, loss is magnitude of violation

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Loss:	2.9		

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector

$S = f(x_i, W)$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$

i.e. $s_j - s_{y_i} + 1 \leq 0$

If true, loss is 0

If false, loss is magnitude of violation

$$= \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$$

$$= \max(0, 2.9) + \max(0, -3.9)$$

$$= 2.9 + 0$$

$$= 2.9$$

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Loss:	2.9	0	

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector

$S=f(x_i, W)$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$

i.e. $s_j - s_{y_i} + 1 \leq 0$

If true, loss is 0

If false, loss is magnitude of violation

$$= \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$= \max(0, -2.6) + \max(0, -1.9)$$

$$= 0 + 0$$

$$= 0$$

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Loss:	2.9	0	12.9

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector

$S=f(x_i, W)$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$

i.e. $s_j - s_{y_i} + 1 \leq 0$

If true, loss is 0

If false, loss is magnitude of violation

$$= \max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1)$$

$$= \max(0, 5.3 + 1) + \max(0, 5.6 + 1)$$

$$= 6.3 + 6.6$$

$$= 12.9$$

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Loss:	2.9	0	12.9

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector

$S = f(x_i, W)$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Want: $s_{y_i} \geq s_j + 1$

i.e. $s_j - s_{y_i} + 1 \leq 0$

If true, loss is 0

If false, loss is magnitude of violation

$$= \max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1)$$

$$= \max(0, 5.3 + 1) + \max(0, 5.6 + 1)$$

$$= 6.3 + 6.6$$

$$= 12.9$$

Linear classifier: Hinge loss

Example :

Suppose: 3 training examples, 3 classes

With some W the scores $f(x, W) = W_x$



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Loss:	2.9	0	12.9

Hinge Loss:

Given Example (x_i, y_i) where x_i is the image and y_i is (the integer) label, and using the shorthand for the scores vector $S=f(x_i, W)$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Full training loss is the overall example in the training data:

$$L = \frac{1}{N} \sum_{i=1}^N L_i$$

$$= (2.9 + 0 + 12.9) / 3$$
$$= 5.3$$

Classification losses (discrete values)

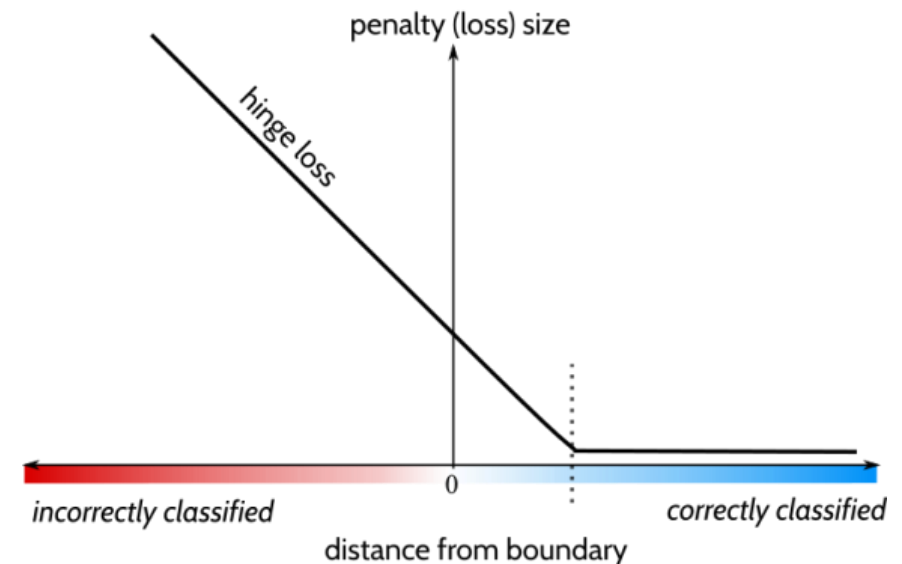
Hinge Loss/binary classification problem with two classes

- If a computed value gives a correct classification but is too close to zero (where too close is defined by a margin) there is a small hinge loss.
- And, if a computed value gives an incorrect classification there will always be a hinge loss with quite high value.

The loss function is defined as:

$$L(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

- where y is the true class label ($y = -1$ or $y = 1$) and $f(x)$ is the predicted score for the positive class.
- If $yf(x) \geq 1$, then the loss is zero, which means that the prediction is correct. If $yf(x) < 1$, then the loss is proportional to the distance from the correct prediction.



Classification losses (discrete values)

Hinge Loss/binary classification problem with two classes

Example





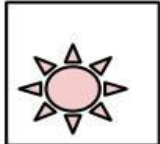

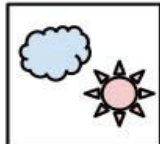
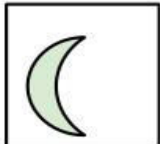
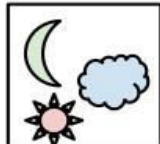
	actual	predicted	hinge loss
[0]	+1	0.97	0.03
[1]	+1	1.20	0.00
[2]	+1	0.00	1.00
[3]	+1	-0.25	1.25
[4]	-1	-0.88	0.12
[5]	-1	-1.01	0.00
[6]	-1	-0.00	1.00
[7]	-1	0.40	1.40

- The Hinge loss is that the boundary separates negative and positive instances as +1 and -1, with -1 being on the left side of the boundary and +1 being on the right.

- [0]: the actual value of this instance is +1 and the predicted value is 0.97, so the hinge loss is very small as the instance is very far away from the boundary.
- [1]: the actual value of this instance is +1 and the predicted value is 1.2, which is greater than 1, thus resulting in no hinge loss.
- [2]: the actual value of this instance is +1 and the predicted value is 0, which means that the point is on the boundary, thus incurring a cost of 1.

Cross-Entropy for Classification

- **Cross-entropy** is a commonly **used loss function** for **classification tasks**.
- **Cross-entropy** builds upon the **idea of information theory entropy** and **measures the difference** between **two probability distributions** for a given **random variable/set of events**.

Multi-Class				Multi-Label					
C = 3	Samples			Samples					
									
	Labels (t)			[0 0 1]	[1 0 0]	[0 1 0]	[1 0 1]	[0 1 0]	[1 1 1]

One-of-many classification

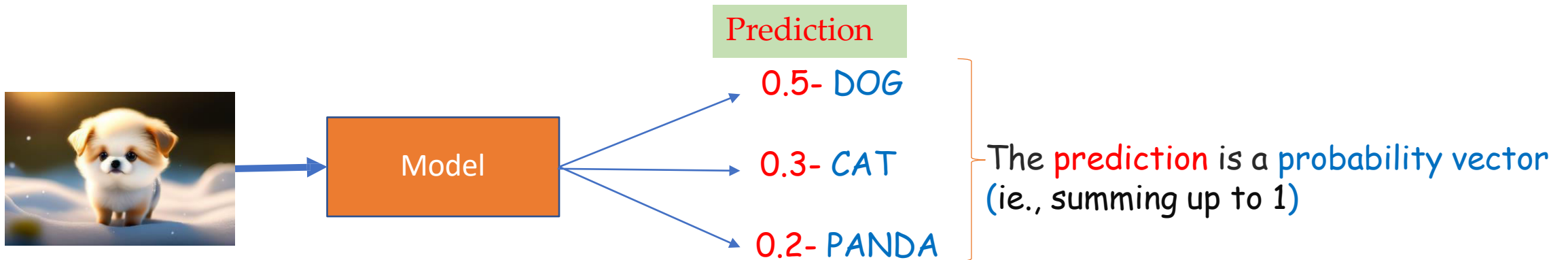
Each sample can belong to more than one class

Cross-Entropy Loss

Cross-Entropy

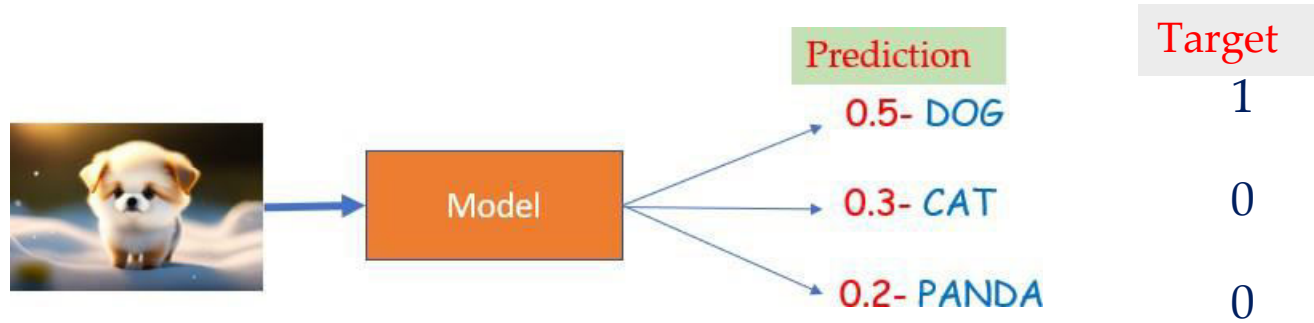
- According to Shannon, entropy is the average number of bits required to represent or transmit an event drawn from the probability distribution for the random variable. entropy indicates the amount of uncertainty of an event.

$$H(t,p) = - \sum_{s \in S} t(s) \cdot \log(p(s))$$



Cross-Entropy Loss

Cross-Entropy



Compute the cross-entropy loss for this image

I) loss for each class separately

Probability of class
in X in Target

Probability of class
in X in Prediction

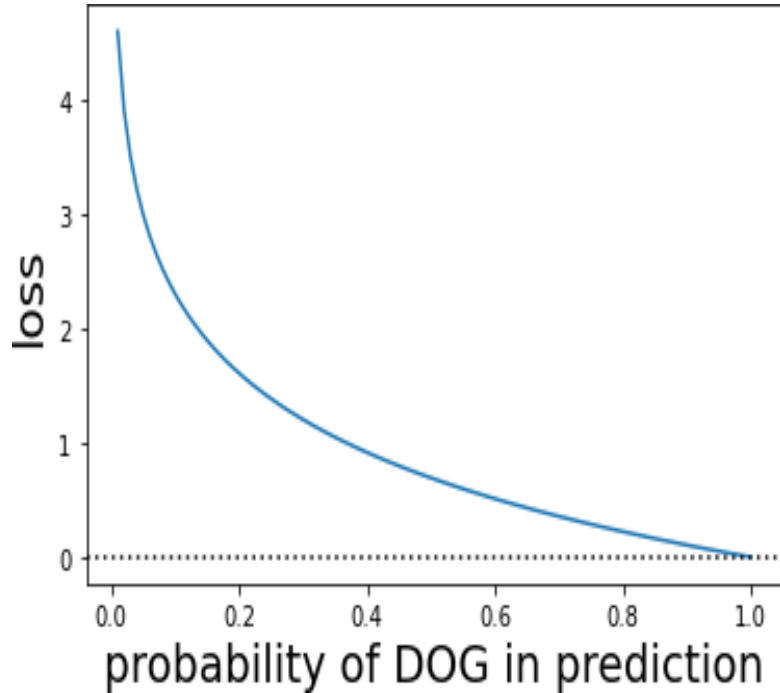
$$\text{Loss for a Class X} = -P(X) \cdot \log q(X)$$

$$1) \text{ Loss CAT} = -P(\text{CAT}) \cdot \log q(\text{CAT}) = -0. \log q(\text{CAT}) \Rightarrow 0$$

$$2) \text{ Loss PANDA} = -P(\text{PANDA}) \cdot \log q(\text{PANDA}) = -0. \log q(\text{PANDA}) \Rightarrow 0$$

$$3) \text{ Loss DOG} = -P(\text{DOG}) \cdot \log q(\text{DOG}) = -1. \log q(0.5) \Rightarrow 0.693$$

Cross-Entropy Loss



- 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.

Cross-Entropy Loss

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

$$\text{Cross-Entropy} = \text{Loss For Dog} + \text{Loss For Cat} + \text{Loss For Panda} \\ = 0.693 + 0 + 0 = 0.693$$

So, Cross-Entropy Loss is

$$\text{Cross-Entropy Loss} = -\sum P(X) \cdot \text{Log } q(X)$$

Categorical cross-entropy

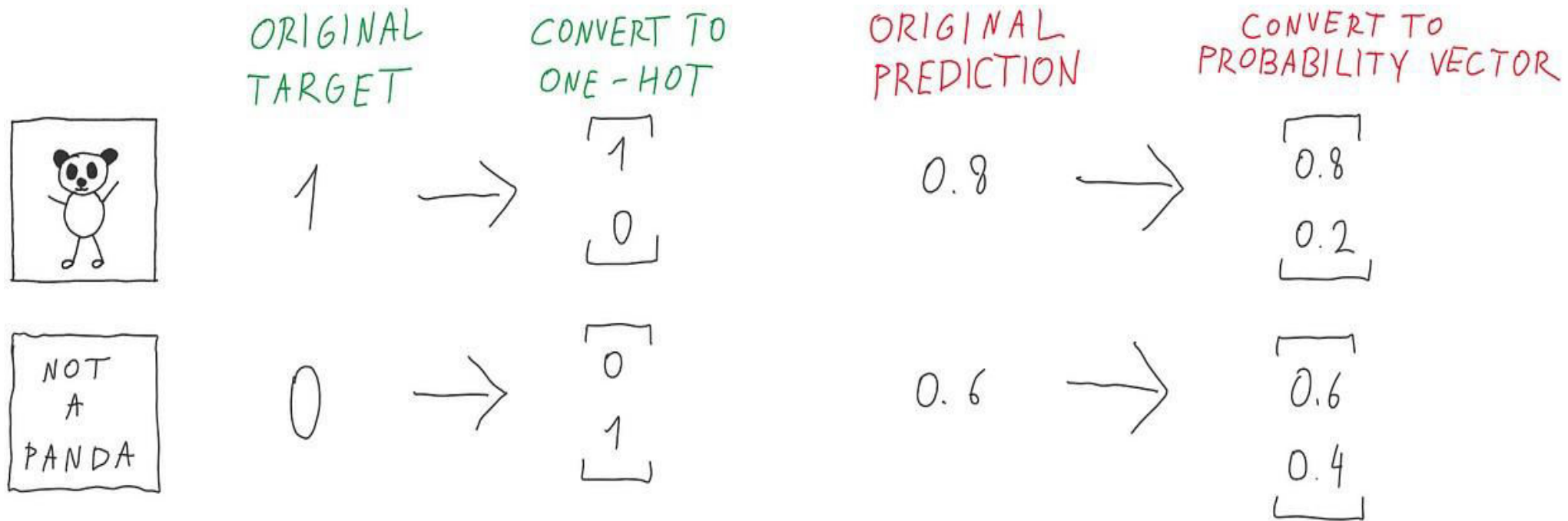
- If our target is a **one-hot vector**, we can indeed forget targets and predictions for all the other classes and compute **only the loss for the hot class**. This is the **negative natural logarithm** of **our prediction**.

$$\text{Categorical cross-entropy} = -\log q(X)$$

Cross-Entropy Loss

Binary Cross-Entropy

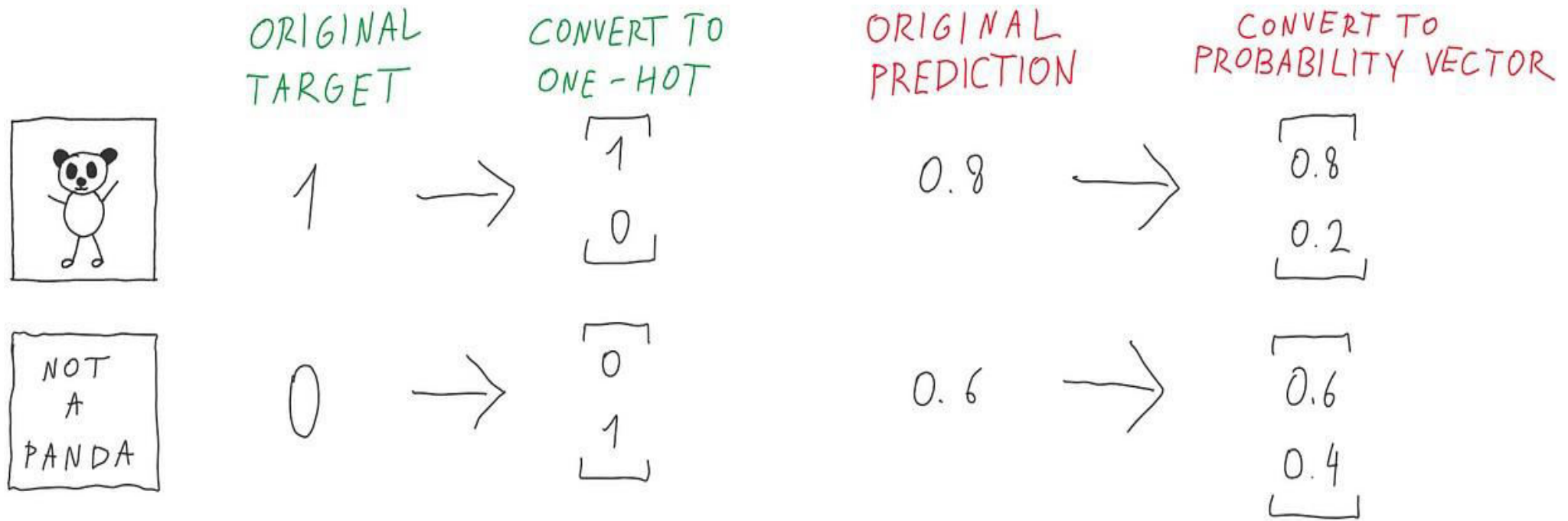
- Binary cross-entropy is another special case of cross-entropy — used if our target is either 0 or 1.



Cross-Entropy Loss

Binary Cross-Entropy

- Binary cross-entropy is another special case of cross-entropy — used if our target is either 0 or 1.



- The probability of a panda would be the same as the prediction and the probability of not-a-panda would be 1-prediction.

Cross-Entropy Loss

Binary Cross-Entropy

This will cancel out if our target is 0

• Binary cross-entropy = $-P(X) \cdot \text{Log } q(X) + (1-P(X)) \cdot \text{Log } (1-q(X))$

This will cancel out if our target is 1

Or

$$BCE = -\frac{1}{n} \sum_{i=1}^n (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

- n - the number of data points.
- y - the actual label of the data point. Also known as true label. Can only be 0 or 1.
- \hat{y} - the predicted probability of the data point. Between 0 and 1. This value is returned by model.

Cross-Entropy Loss

Binary Cross-Entropy

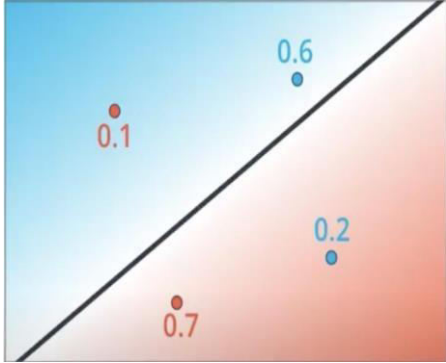
Example

actual labels of y	1	0	0	1
predicted probabilities of \hat{y}	0.8	0.2	0.6	0.9

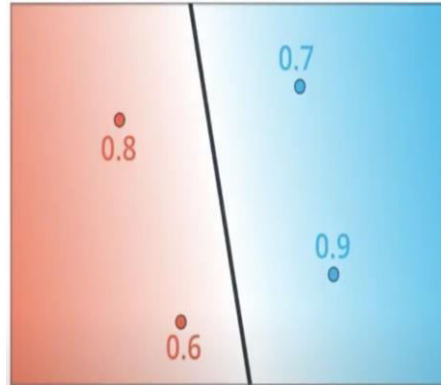
$$\begin{aligned} \text{BCE} &= - \\ & \frac{(1 * \log(0.8) + (1-1) * \log(1-0.8)) + (0 * \log(0.2) + (1-0) * \log(1-0.2)) + (0 * \log(0.6) + (1-0) * \log(1-0.6)) + (1 * \log(0.9) + (1-1) * \log(1-0.9))}{4} \\ &= - \frac{-0.09691 - 0.09691 - 0.39794 - 0.04576}{4} \\ &= - \frac{-0.63752}{4} \\ &= 0.159379 \end{aligned}$$

Example Cont'd

Model-A



Model-B



Product probability: The probability of *two(or more) independent events* that are occurring together is calculated by multiplying the events' individual probabilities.

- We want to calculate the total probability of the models by multiplying the probability of each independent student.

- **Product Probability Model A:**

- $0.1 * 0.7 * 0.6 * 0.2 = 0.0084$

- **Product Probability Model B:**

- $0.8 * 0.6 * 0.7 * 0.9 = 0.3024$

- The product probability for model B is better than that of A.

Example Cont'd

- *Product probability works better* when we *have a few items to predict*, but this *is not the case with real-life model predictions*.
- For instance, if we have a class full of *1000 students*, the product probabilities will always be *closer to 0*, regardless of how good your model is. If we also change *one probability*, *the product will change drastically and give the wrong impression* that a model performs well. So, we need to transform the products to a sum using a *logarithmic function*.

Log Model A:

- $\log(0.1) + \log(0.7) + \log(0.6) + \log(0.2)$
 $-1 + -0.154 + -0.221 + -0.698 = -2.073$

Log Model B:

- $\log(0.8) + \log(0.6) + \log(0.7) + \log(0.9)$
 $-0.09 + -0.22 + -0.15 + -0.045 = -0.505$

Example Cont'd

- The log of a number *between 0 and 1* will always be *negative*. Is the above a better way to evaluate our model performance? Not really. Instead, we'll take the *negative logarithm of predicted probabilities*.
- Negative Logs Model A:
$$-\log(0.1) + -\log(0.7) + -\log(0.6) + -\log(0.2)$$
$$1 + 0.154 + 0.221 + 0.698 = 2.073$$
- Negative Logs Model B:
$$-\log(0.8) + -\log(0.6) + -\log(0.7) + -\log(0.9)$$
$$0.09 + 0.22 + 0.15 + 0.045 = 0.505$$
- Cross-entropy loss is the *sum of the negative logarithm of the predicted probabilities* of each student.
- *Model A's cross-entropy loss is 2.073; Model B's is 0.505. Cross-Entropy gives a good measure of how effective each model is.*

Multi-class cross-entropy / categorical cross-entropy

- The probabilities of each container need to sum to 1.
- If you are dealing with a multi-class classification problem you can calculate the Log loss in the same way

$$L = -\frac{1}{m} \sum_{i=1}^m y_i \cdot \log(\hat{y}_i)$$

Fruit	Container A probabilities	Container B probabilities	Container C probabilities
Orange	0.7	0.3	0.1
Apple	0.2	0.4	0.5
Lemon	0.1	0.3	0.4

	Container A	Container B	Container C
Correct fruits in the Respective containers	Oranges	Lemon	Lemon
The predicted probability that the fruit is correct	0.7	0.3	0.4

- Product probabilities = $0.7 * 0.3 * 0.4 = 0.084$
- Cross Entropy = $-\log(0.7) + -\log(0.3) + -\log(0.4) = 1.073$

Fruits	Container A probabilities	Container B probabilities	Container C probabilities
Orange	$p1_A$	$p1_B$	$P1_C$
Apple	$p2_A$	$p2_B$	$P2_C$
Lemon	$p3_A$	$p3_B$	$P3_C$

What is the probability that it's either an *orange*, *apple*, or *lemon* in container A? We have 0.7, 0.2, and 0.1, respectively.

- The $y1$ value for container **A** is equal to 1 if it contains particular fruit; otherwise, it is 0.
- $y1_A$ - if it's an orange
- $y2_A$ - if it's an apple
- $y3_A$ - if it's a lemon.

Cross-Entropy for Container A:

$$-\log(p1_A) + -\log(p2_A) + -\log(p3_A)$$

Cross-Entropy for Container B:

$$-\log(p1_B) + -\log(p2_B) + -\log(p3_B)$$

Cross-Entropy for Container C:

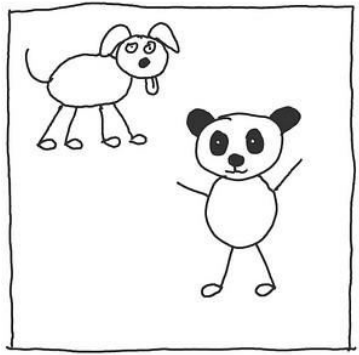
$$-\log(p1_C) + -\log(p2_C) + -\log(p3_C)$$

Total cross Entropy:

$$-\sum_{n=i}^n \sum_{j=1}^c y_{ij} \log(p_{ij})$$

Multi-label classification

- Cross-entropy can also be used as a loss function for a multi-label problem.



TARGET

$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

PREDICTION

$\begin{bmatrix} 0.6 \\ 0.7 \\ 0.4 \end{bmatrix}$

our target and prediction are **not** a probability vector

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

$\approx 0.510...$

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

Multi-label classification

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

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

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

Kullback-Leibler Divergence

- Kullback-Leibler Divergence: KL divergence is the measure of the relative difference between two probability distributions for a given random variable or set of events. KL divergence is also known as Relative Entropy.

$$D_{KL}(P\|Q) = \mathbb{E}_{x \sim P} \left[\log \frac{P(X)}{Q(X)} \right]$$

- Consider two probability distributions P, Q , on some space X . The Kullback-Leibler divergence from Q to P (written as $D_{KL}(P\|Q)$).