



SS3 (HML x DLA x MLG)

[1. Materials](#)

[2. MLPs](#)

[2.1. Cross Entropy](#)

[2.2. Idea](#)

1. Materials

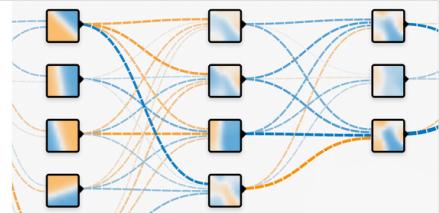
Tensorflow - Neural Network Playground

It's a technique for building a computer program that learns from data. It is based very loosely on how we think the human brain works. First, a collection of software "neurons" are created and connected together, allowing them to send messages to each other. [🔗 <https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=4,2&seed=0.78406&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>](https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle®Dataset=reg-plane&learningRate=0.03®ularizationRate=0&noise=0&networkShape=4,2&seed=0.78406&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false)

[Primer on TensorFlow and Keras: The past \(TF1\) & the present \(TF2\)](#)

[Zero#false&hideText#false style known as define-then-run. This is opposed to define-by-run which is for example Python execution style. But what does that mean? Define then run means that, just because you called/defined something it's not executed. You](#)

[⚠️ <https://stackoverflow.com/questions/59112527/primer-on-tensorflow-and-keras-the-past-tf1-the-present-tf2>](#)



Keras vs Tensorflow vs Pytorch [Updated] | Deep Learning Frameworks | Simplilearn

Deep learning is a subset of Artificial Intelligence (AI), a field growing in popularity over the last several decades. Like any new concept, some questions and details need ironing out before employing it in real-world applications. But before we explore the PyTorch vs TensorFlow vs

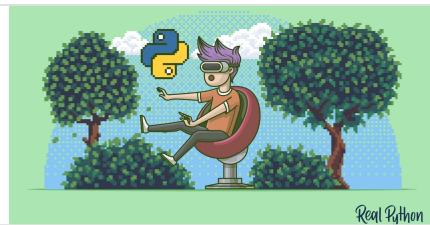
[💡 <https://www.simplilearn.com/keras-vs-tensorflow-vs-pytorch-article>](#)



Python Virtual Environments: A Primer - Real Python

Updated 2018-01-12: Clarified pyenv vs. venv usage on Python 3.6+ Updated 2016-06-11: Added section on changing Python versions with virtualenv Python, like most other modern programming languages, has its own unique way of downloading,

 <https://realpython.com/python-virtual-environments-a-primer/>



2. MLPs

2.1. Cross Entropy

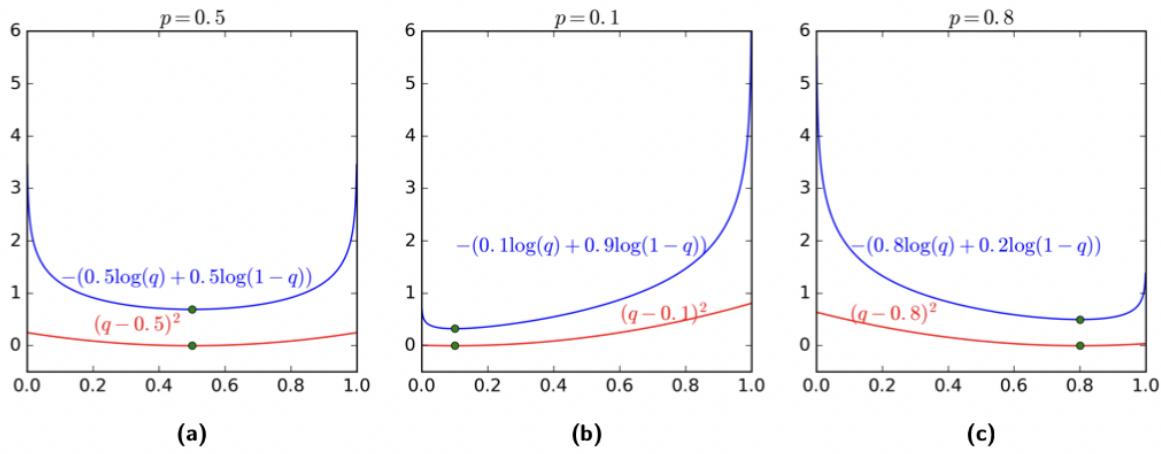
Hàm mất mát của softmax regression được xây dựng dựa trên bài toán tối thiểu sự khác nhau giữa *đầu ra dự đoán a* và *đầu ra thực sự y* (ở dạng one-hot). Khi cả hai là các vector thể hiện xác suất, khoảng cách giữa chúng thường được đo bằng một đại lượng được gọi là *cross entropy*. Một đặc điểm nổi bật của đại lượng này là nếu cố định một vector xác suất, giá trị của nó *đạt giá trị nhỏ nhất khi hai vector xác suất bằng nhau, và rất lớn khi hai vector đó lệch nhau nhiều*.

Cross entropy giữa hai vector phân phối \mathbf{p} và \mathbf{q} rời rạc được định nghĩa bởi

$$H(\mathbf{p}, \mathbf{q}) = - \sum_{i=1}^C p_i \log q_i \quad (15.4)$$

Hình 15.4 thể hiện rõ ưu điểm của hàm cross entropy so với hàm bình phương khoảng cách Euclid. Đây là ví dụ trong trường hợp $C = 2$ và p_1 lần lượt nhận các giá trị 0.5, 0.1 và 0.8. Chú ý rằng $p_2 = 1 - p_1$. Có hai nhận xét quan trọng sau đây:

1. Giá trị nhỏ nhất của cả hai hàm số đạt được khi $q = p$ tại hoành độ các điểm màu lục.



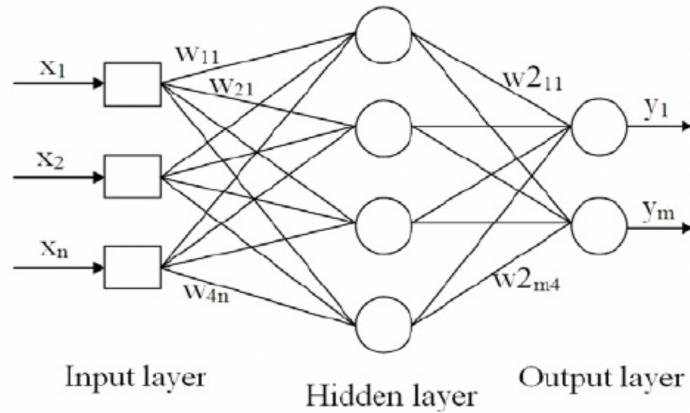
Hình 15.4: So sánh giữa hàm cross entropy và hàm bình phương khoảng cách. Các điểm màu lục thể hiện các giá trị nhỏ nhất của mỗi hàm.

- Quan trọng hơn, hàm cross entropy nhận giá trị rất cao (tức mất mát rất cao) khi q ở xa p . Trong khi đó, sự chênh lệch giữa các mất mát ở gần hay xa nghiệm của hàm bình phương khoảng cách $(q-p)^2$ là ít đáng kể hơn. Về mặt tối ưu, hàm cross entropy sẽ cho nghiệm gần với p hơn vì những nghiệm ở xa bị *trừng phạt* rất nặng.

Hai tính chất trên đây khiến cho cross entropy được sử dụng rộng rãi khi tính khoảng cách giữa hai phân phối xác suất. Tiếp theo, chúng ta sẽ chứng minh nhận định sau.

<https://www.youtube.com/watch?v=ErfnhcEV1O8>

2.2. Idea



$$\hat{\mathbf{y}} = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix} \quad D(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_j y_j \ln \hat{y}_j \quad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\Rightarrow \text{cross-entropy}(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{k=1}^K y_k \cdot \log_k(\hat{y}_k)$$

□ Mô hình hóa:

```

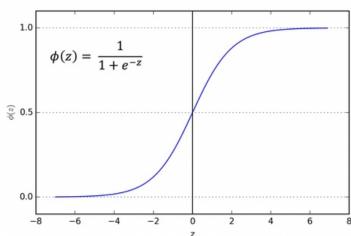
hidden = x.dot(W1) + b1 # X: [1, K] ; W1: [K, M] ; b1: [M,]
hidden = activation(hidden) # [1, M]
logits = hidden.dot(W2) + b2 # [1, num_classes]
outputs = softmax(logits) # [1, num_classes]
loss = cross-entropy(outputs, one-hot(real-label))
predicted-label = argmax(outputs)i # [1,]
    
```

□ Hàm softmax:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

□ Hàm activation:

> sigmoid



> ReLU

