Boston house prices (PyTorch)

1) Implement pytorch based housing regression using Boston dataset(not california) and model:

$$y' = M(x) = LeakyReLU(Linear(\tanh(Linear(W \cdot x + b))))$$

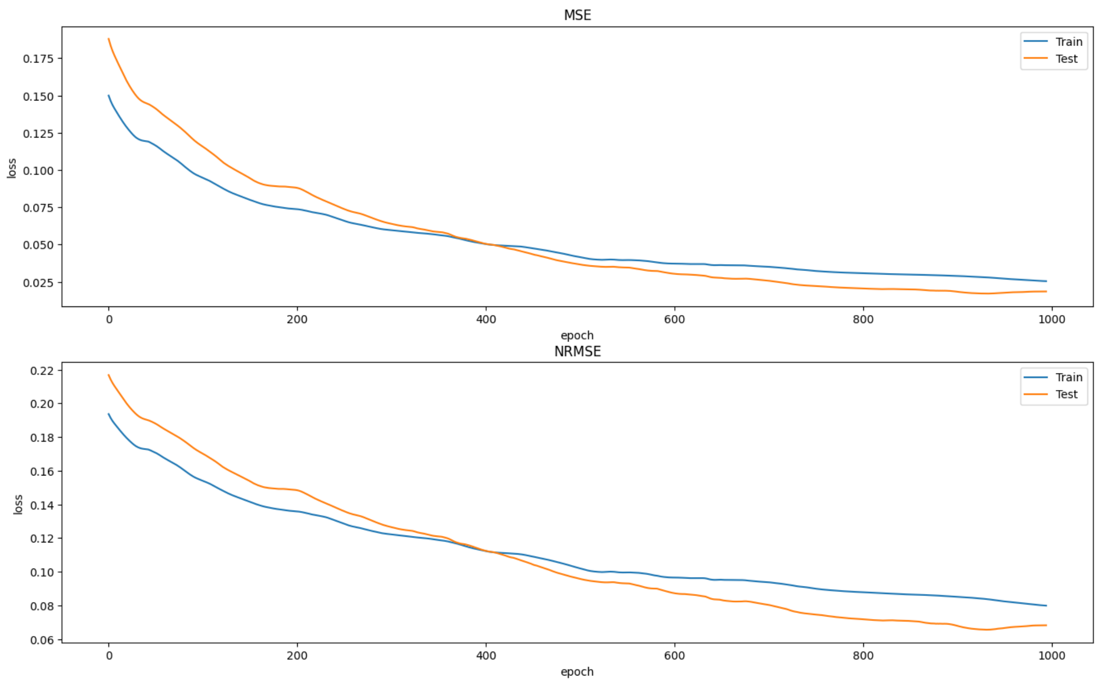
Where:

$$LeakyReLU(x) = \begin{cases} x, x > 0 \\ \alpha \cdot x, x \leq 0 \end{cases}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$Linear(x) = W \cdot x + b$$

MSE and NRMSE loss function results (x1000 epochs):



I had to store all the data as tensors in GPU memory and decrease **.item()** which does data sync (VRAM → RAM) and decreases performance. I guess something similar happens when calling **.item()** while running on CPU. So I'm calculating loss values once per 1000 epochs. Speedup is significant.

Wine classification (NumPy, PyTorch)

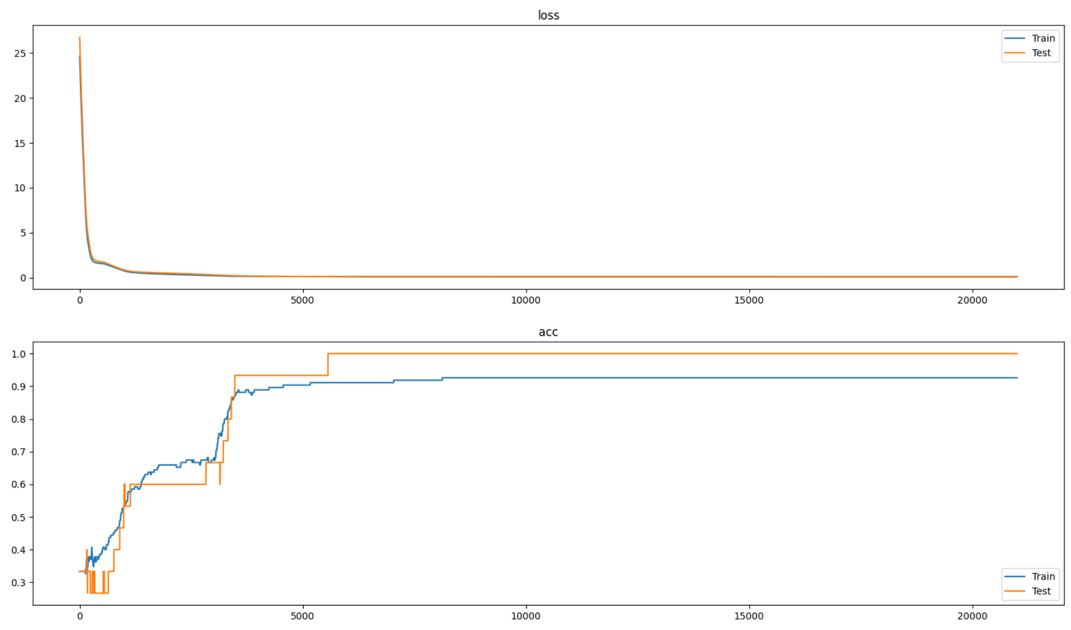
$$L(y, y') = -\frac{1}{N}\sum y \cdot \log(y')$$

$$\frac{\partial L(y, y')}{\partial y'} = -y \cdot \frac{1}{y'} = -\frac{y}{y'}$$

$$SoftMax(y = j|x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

$$\begin{bmatrix} a_0(1-a_0) & -a_0 a_1 & -a_0 a_2 \\ -a_1 a_0 & a_1(1-a_1) & -a_1 a_2 \\ -a_2 a_0 & -a_2 a_1 & a_2(1-a_2) \end{bmatrix}$$

Loss/accuracy for the Iris tutorial:



Accuracy is being calculated as correct guess count divided by total guess count:

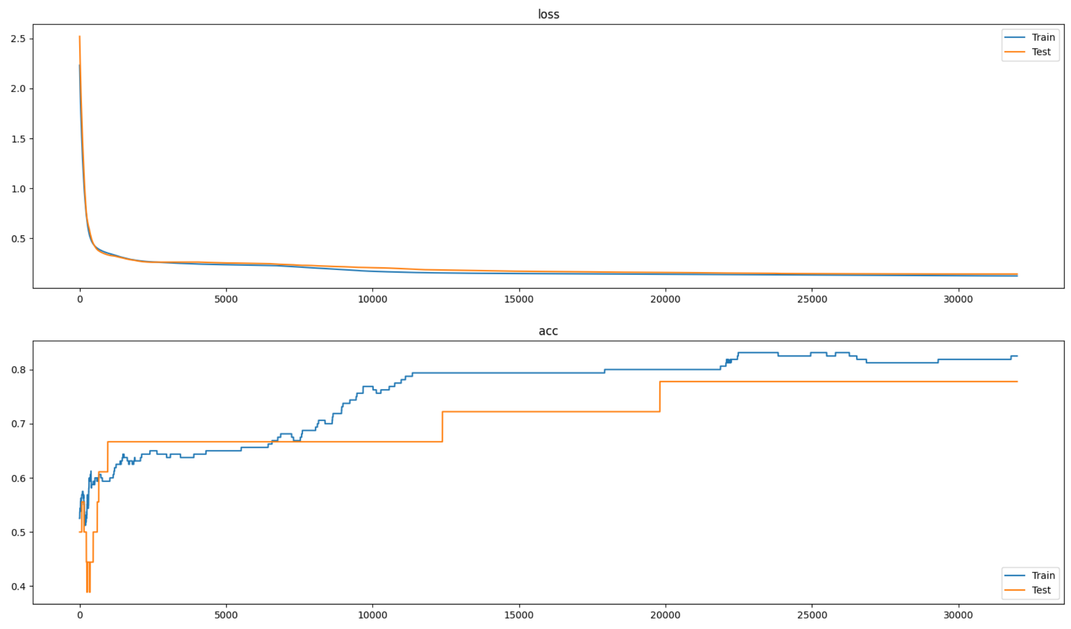
```
guess_cnt = reduce(
    lambda cnt, values: cnt + (np.argmax(values[0]) == np.argmax(values[1])),
    zip(y, y_prim.value),
    0
)
accuracy = guess_cnt / len(y_prim.value)
```

2) Implement numpy based classification using dataset – sklearn.datasets.load_wine

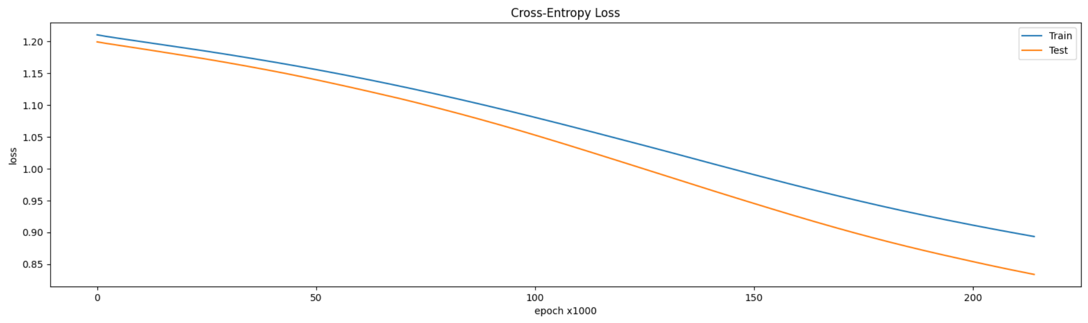
Successfully walked through Iris video tutorial, for some reason same code with different dataset and adjusted input count fails to calculate loss (results in nan).

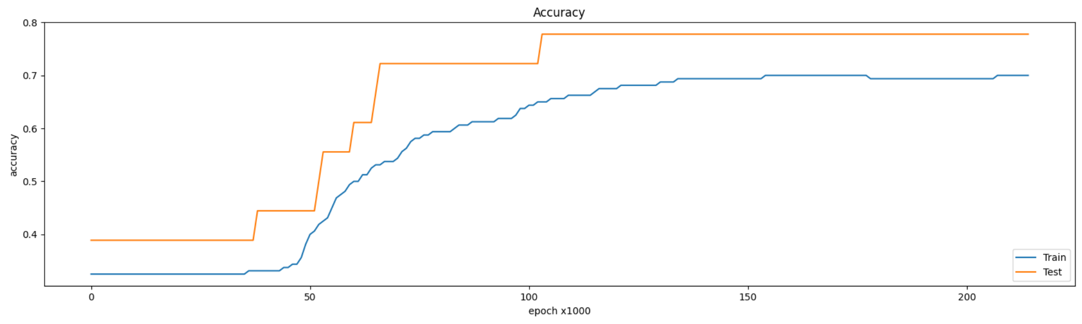
```
def forward(self, x: Variable):
    self.x = x
    np_x = np.copy(x.value)
    # numerical stability for large values
    np_x -= np.max(np_x, axis=1, keepdims=True)
    self.output = Variable(
        (np.exp(np_x + 1e-8)) / np.sum(np.exp(np_x), axis=1, keepdims=True)
    )
    return self.output
```

This is strange. But once I normalized data it started working. PyTorch works in both cases.



3) Implement pytorch based classification using dataset – sklearn.datasets.load_wine


Cross-Entropy Loss
Train
Test
loss
epoch x1000


Accuracy
Train
Test
accuracy
epoch x1000