Project : Physics informed neural networks PINNs

Outline:

- a)What is a PINN
- b)example: harmonic oscilator
  - i)training
  - ii)CPINN - Competitive PINNS
- references

a)What is a PINN?

We are trying to approximate a particular solution of second order inhomegenous ODE

$$m * \ddot{y} + c * \dot{y} + k * y = g(t),$$

with a neural network of the form

$$L_i = \sigma_i\left(\begin{bmatrix} & W_i & \end{bmatrix} * \begin{bmatrix} x_i \end{bmatrix} + \begin{bmatrix} b_i \end{bmatrix}\right)$$

$$N(t) = (L_1 \circ L_2 \circ L_3 \circ L_o)(t)$$

where σ_i is an yet undetermed activation function. and $L_o \to \mathbb{R}$ is the output neuron of the form.

$$L_o = \sigma\left(\begin{bmatrix} & w & \end{bmatrix} * \begin{bmatrix} x \end{bmatrix} + \lambda\right)$$

Therefore we define a loss function

$$\mathcal{L}(\mathbf{W}) = \frac{1}{M}\sum_{i=1}^{M} \underline{(m * \ddot{N}(t_i) + c * \dot{N}(t_i) + k * N(t_i) - g(t))^2} + (N(0) - y_0)^2 + (\dot{N}(0) - \dot{y}_0)^2$$

Where sceond and third terms are the initial value losses. To minimize the loss function we choose Stochastic Gradient descent algorithm.

$--$

b)example: harmonic oscilator

i)training

Consider the harmonic oscilator

$$\ddot{y} + r * \dot{y} + \omega^2 * y = B * cos(\Omega * t),$$

for simplicity we set all parameters to 1,

$$\ddot{y} + \dot{y} + y = cos(t),$$

and initialize the L_i ,

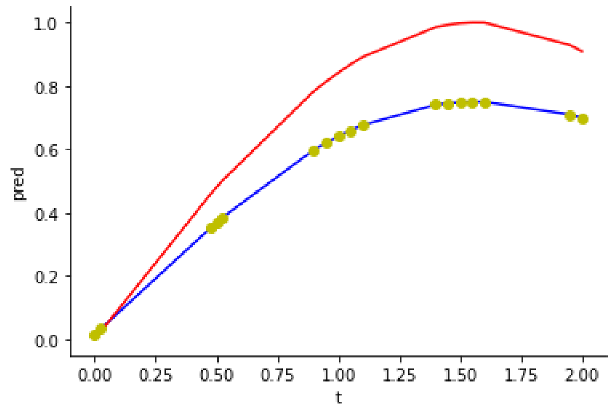
$$L_i = tanh\left(\begin{bmatrix} & 32 \times 32 & \end{bmatrix} * \begin{bmatrix} x_i \end{bmatrix} + \begin{bmatrix} 32 \end{bmatrix}\right)$$

with values in [0,1). The calculated solution of the equation is

In [5]:
```
def train(epochs = 300):
    trainable_vars = NeuralNetwork.trainable_variables()
    optimizer = tf.optimizers.SGD(learning_rate=0.01)
    for _ in range(epochs):
        with tf.GradientTape(persistent=True) as tape:
            loss = loss(train_t,NeuralNetwork,g)
        grad = tape.gradient(loss, trainable_vars)
        optimizer.apply_gradients(grad, trainable_vars)
```

That is we calculate the gradient with respect to each variable and shift the variable in the direction of the gradient accordingly,

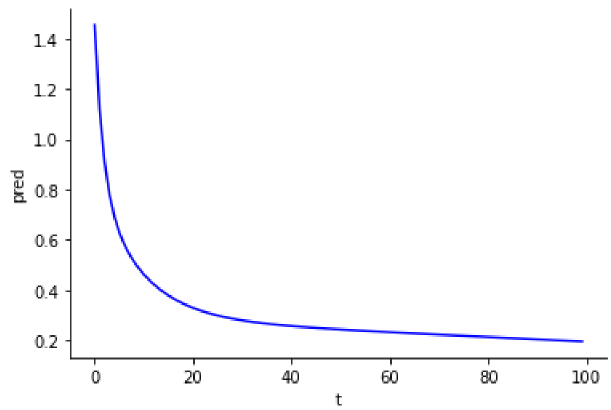
plotting the calculated solution and the predicted solution in *blue* for values in [0,2]. (the dots are training points)



Calculating the error with respect to the L_2 norm,with trapazoid integral yields

$$||NeuralNetwork(t) - sin(t)||_{L_2} = 0.20278955.$$

If we plot the loss, we can see its convergening to ≈ 0.28.



$--$

ii)CPINN - Competitive PINNS

We are trying to tackle this further by adding a discriminator neural network $T(t)$,

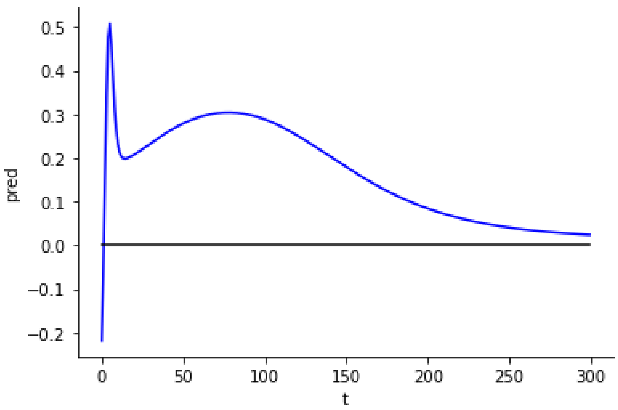
$$\mathcal{L}(N, T, t) = \frac{1}{M} \sum_{i=1}^{M} T(t) * (\ddot{N}(t_i) + \dot{N}(t_i) + N(t_i))^2$$

with the according optimization problem,

$$max_T \ min_N \ \mathcal{L}(N, T, t).$$

We try stochastic gradient descent for T and for N, that is $SGD_{learnablevariables\ L}(-\mathcal{L}(N, T, t))$ and $SGD_{learnablevariables\ NN}(\mathcal{L}(N, T, t))$ for each iteration.

Plotting the loss we see it seems to converge to the 'Nash equilibrium'. $(T, NN) \to (0, y)$ which would be a solution for the equation. This is not expected because trying to apply SGD seperatly just works for specific zero sum optimization problems.



the resulting loss plot and L_2 norm is now

$$||NeuralNetwork(t) - sin(t)||_{L_2} = 0.18570352$$

One has to consider Optimizers for zero sum problems for example Competitive Gradient Descent or Sympletic Gradient Adjustment.

references

- Prof. Seungchul Lee - *Physics-informed Neural Networks (PINN)* "https://i-systems.github.io/tutorial/KSNVE/220525/01_PINN.html#3.2.-Lab-1%3A-Simple-Example "

- Lukas Exl, Sebastian Schaffer, Norbert J. Mauser - *Vorlesungsskript Angewandtes Maschinelles Lernen*

- Benoit Liquet, Sarat Moka, and Yoni Nazarathy - *The Mathematical Engineering of Deep Learning (2021)* https://deeplearningmath.org/general-fully-connected-neural-networks

- Qi Zeng, Yash Kothari, Spencer H. Bryngelson & Florian Schäfer - *COMPETITIVE PHYSICS INFORMED NETWORKS*