

- [Análise de Modelos de Rede Neural](#)
  - [Configuração do Modelo](#)
- [Modelos:](#)
  - [Taxa de aprendizagem: 10](#)
    - [Com 25 Camadas Ocultas](#)
    - [Com 50 Camadas Ocultas](#)
    - [Com 100 Camadas Ocultas](#)
  - [Taxa de aprendizagem: 1](#)
    - [Com 25 Camadas Ocultas](#)
    - [Com 50 Camadas Ocultas](#)
    - [Com 100 Camadas Ocultas](#)
  - [Taxa de aprendizagem: 0.01](#)
    - [Com 25 Camadas Ocultas](#)
    - [Com 50 Camadas Ocultas](#)
    - [Com 100 Camadas Ocultas](#)
- [Resultados das análises:](#)

# Análise de Modelos de Rede Neural

---

Este documento contém o código e as análises dos modelos de rede neural com diferentes configurações de hiperparâmetros.

## Configuração do Modelo

---

Aqui está o código Python para criar o modelo:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
#consigo tanto escolher a quantidade de camadas ocultas quanto a taxa de
aprendizado porém ela é constante
def create_model(hidden_units, learning_rate):
    model = tf.keras.Sequential([
        tf.keras.layers.Dense(784, activation='sigmoid', input_shape=(784,)),
        tf.keras.layers.Dense(hidden_units, activation='sigmoid'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])
    optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=
```

```
['accuracy'])  
    return model
```

A parte que gera todos os gradientes:

```
    return model  
def gradient(_model, _x, _y, _epochs, _batch):  
    historico_de_perdas = [] # Lista para armazenar o histórico de perdas  
    for epoch in range(_epochs):  
        for i in range(0, _x.shape[0], _batch):  
            with tf.GradientTape() as tape:  
                prediction = _model(_x[i:i + _batch])  
                loss = _model.compiled_loss(_y[i:i + _batch], prediction)  
                gradients = tape.gradient(loss, _model.trainable_variables)  
                _model.optimizer.apply_gradients(zip(gradients,  
_model.trainable_variables))  
            predictions = _model(_x)  
            loss = _model.compiled_loss(_y, predictions)  
            historico_de_perdas.append(loss.numpy()) # Armazena a perda da época atual  
            #print(f"Epoch {epoch+1} loss {loss:.4f}")  
    return historico_de_perdas
```

Eu fiz de uma forma universal onde eu conseguiria criar todos os gradientes em uma função só De acordo com o numero do Batch eu consigo definir quantas entradas eu vou calcular o gradiente

Os graficos das analises são gerador por:

```
def graphic(x, y, hidden_layer, r, epochs, batch):  
    plt.plot(gradient(create_model(hidden_layer, r), x, epochs, batch),  
label=f"hiddens layer: {hidden_layer}, r: {r}, Batch: {batch}")  
    plt.title('Convergência do Erro Empírico')  
    plt.xlabel('Época')  
    plt.ylabel('Erro Empírico')  
    plt.legend()  
    plt.savefig(f'grafico_{hidden_layer}_{r}_{batch}.png')  
    plt.close()  
    return f'grafico_{hidden_layer}_{r}_{batch}.png'
```

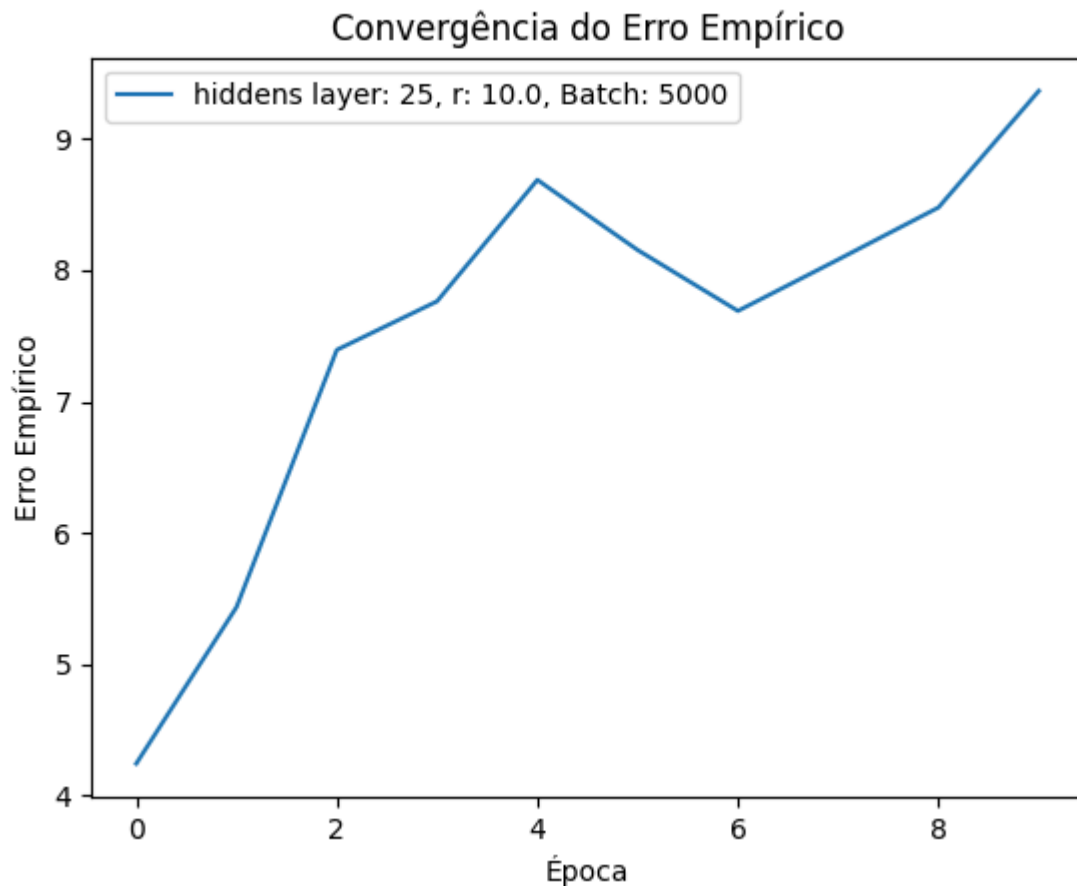
## Modelos:

(Obs: caso não tenha imagem é porque o computador não conseguiu gerar tudo)

# Taxa de aprendizagem: 10

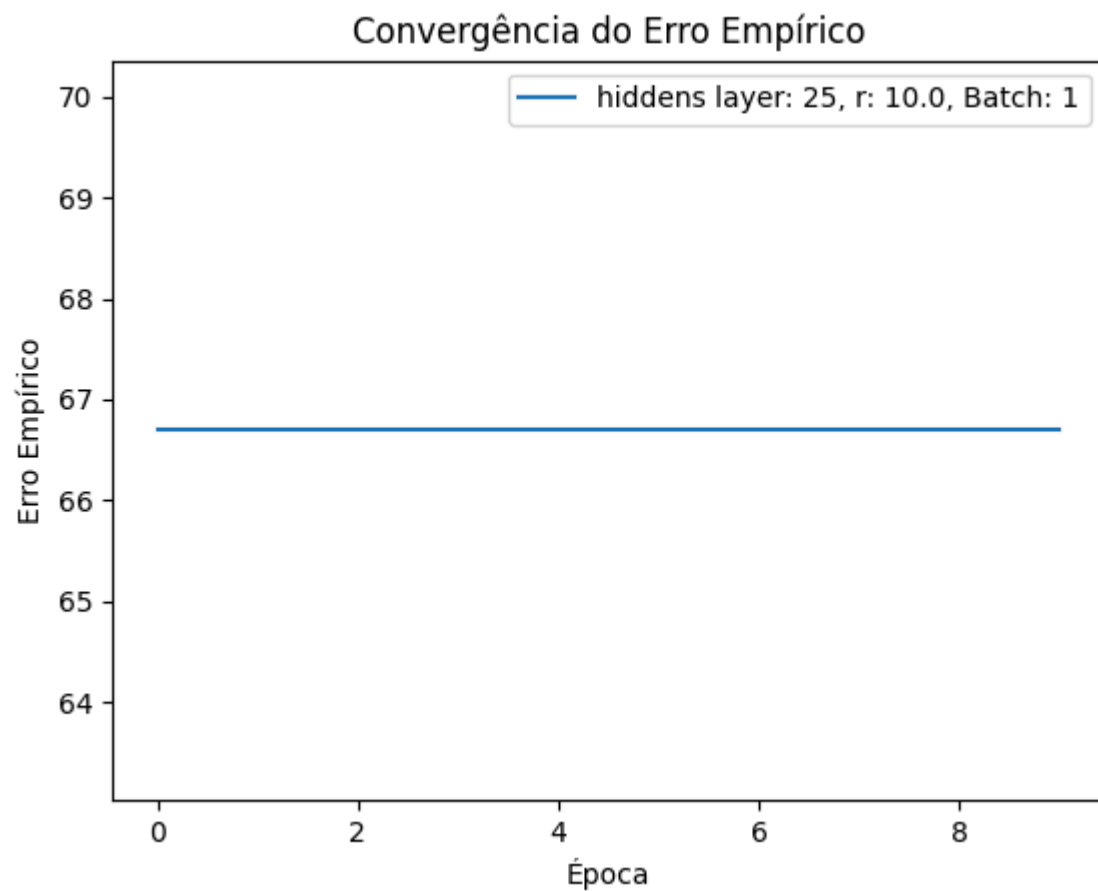
## Com 25 Camadas Ocultas

- Gradient Descent:



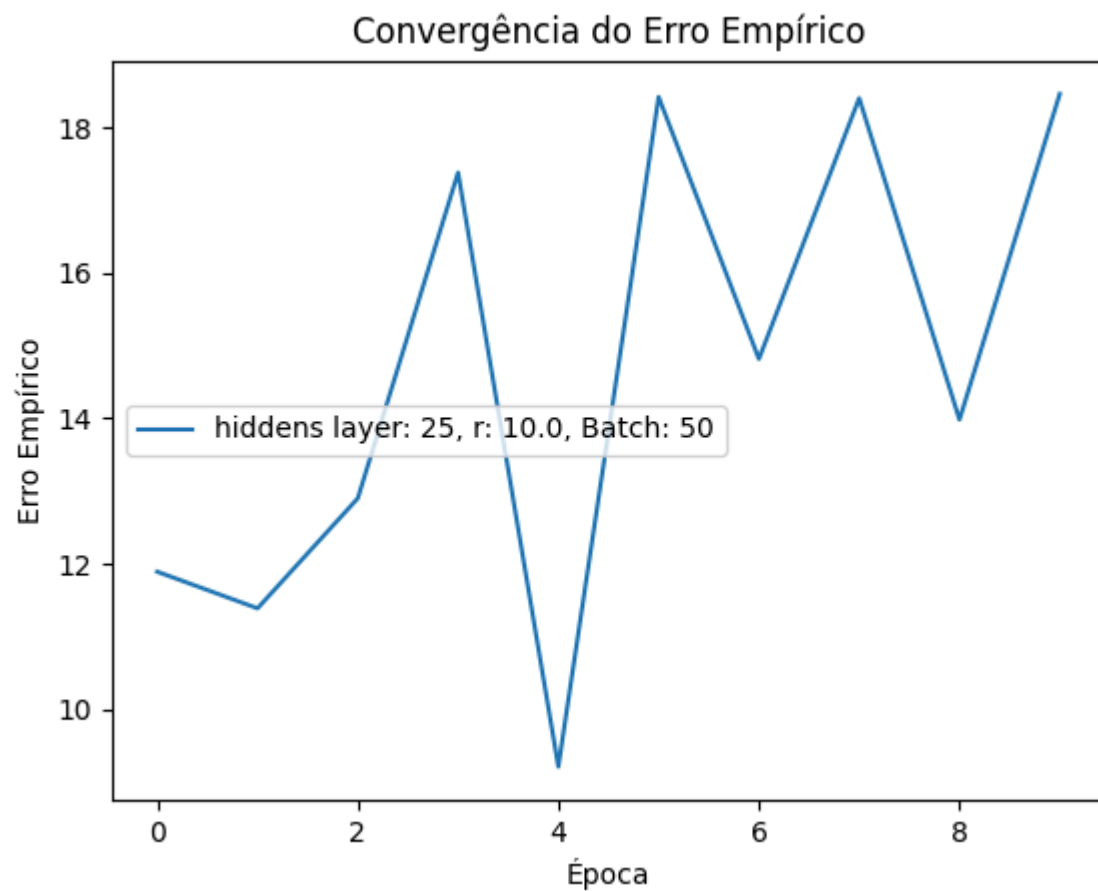
Erro empiroco subiu invez de abaixar.

- **Stochastic Gradient Descent:**



Obs: sendo bem sincero não sei o que aconteceu aqui.

- **Mini-Batch:**

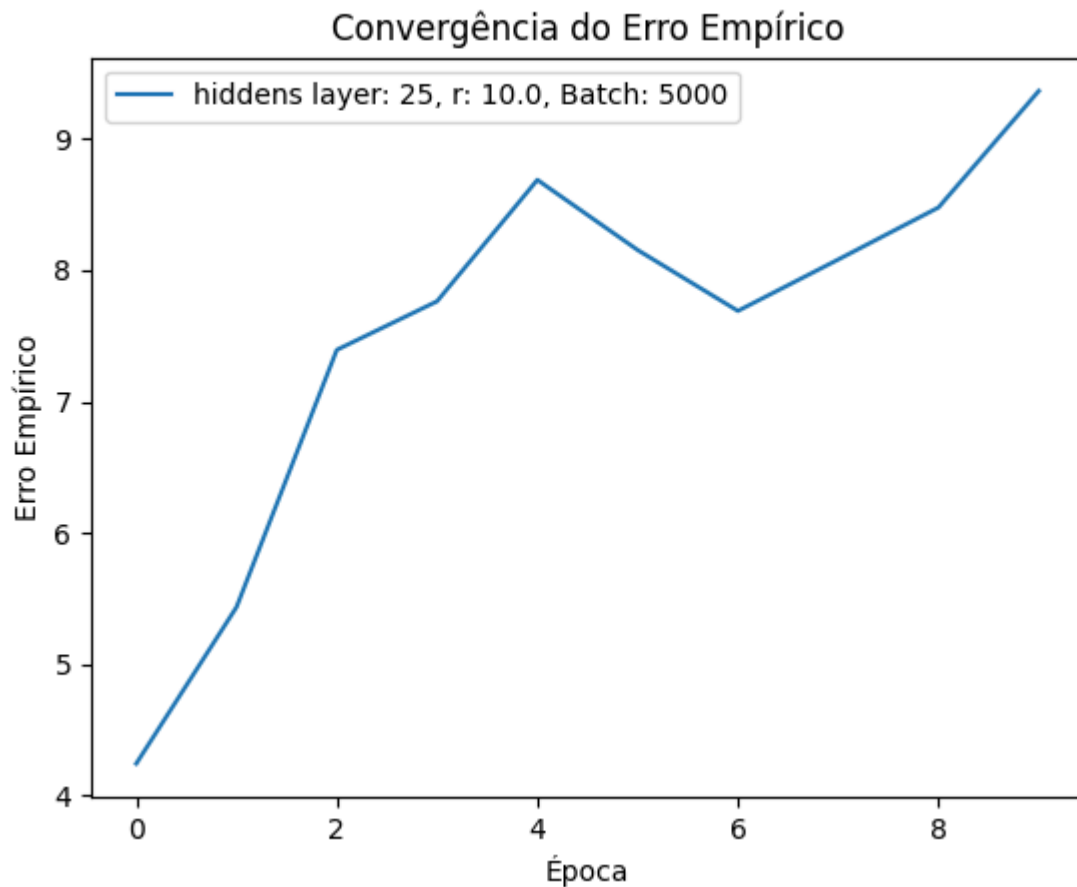


muita instabilidade.

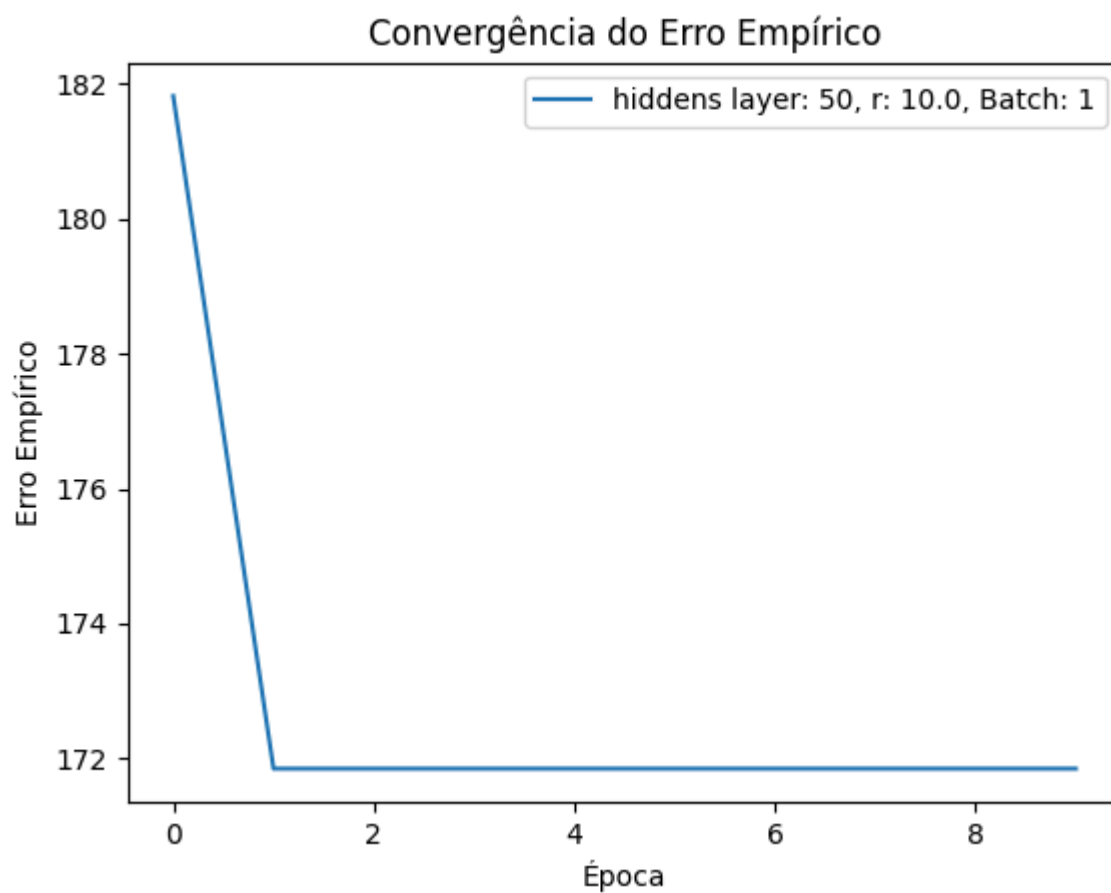
- **RESULTADO DE 25 CAMADAS E  $r = 10$ :** Resultados muito incertos mesmo com alteração dos gradientes

## Com 50 Camadas Ocultas

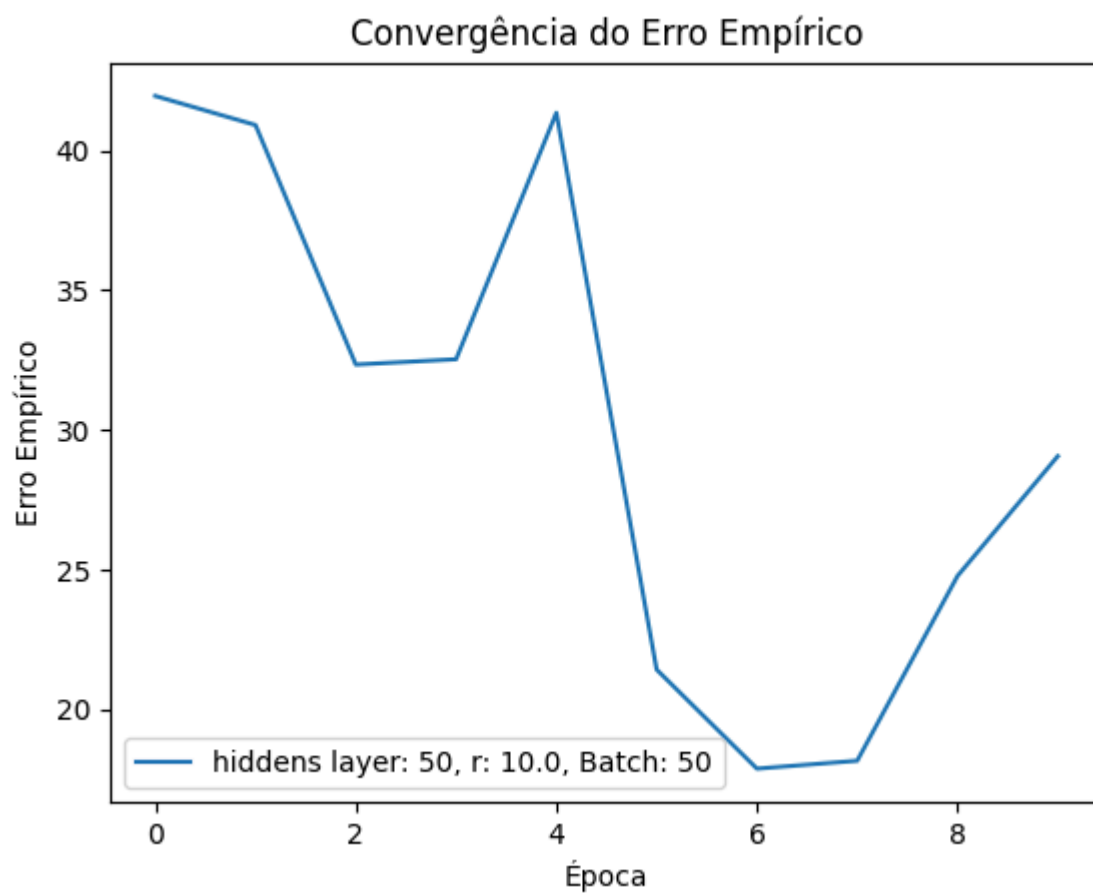
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

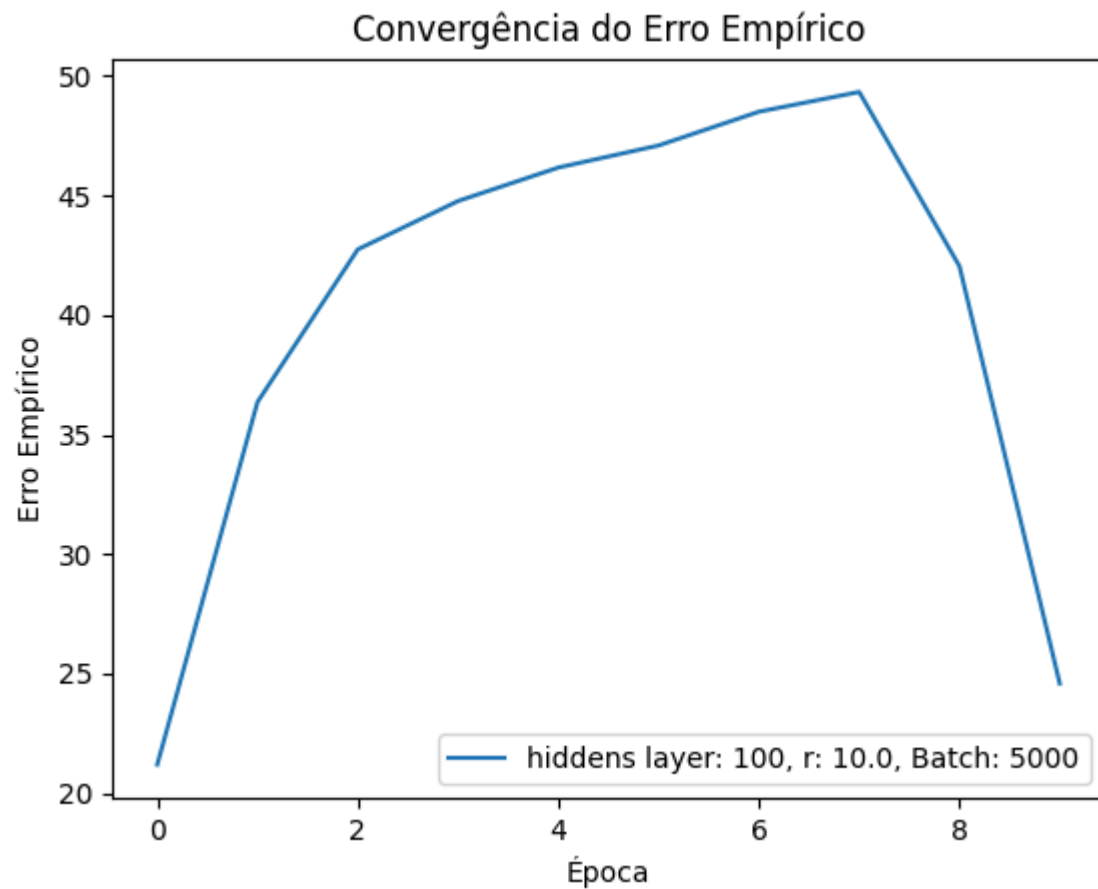


- **Mini-Batch:**




# Com 100 Camadas Ocultas

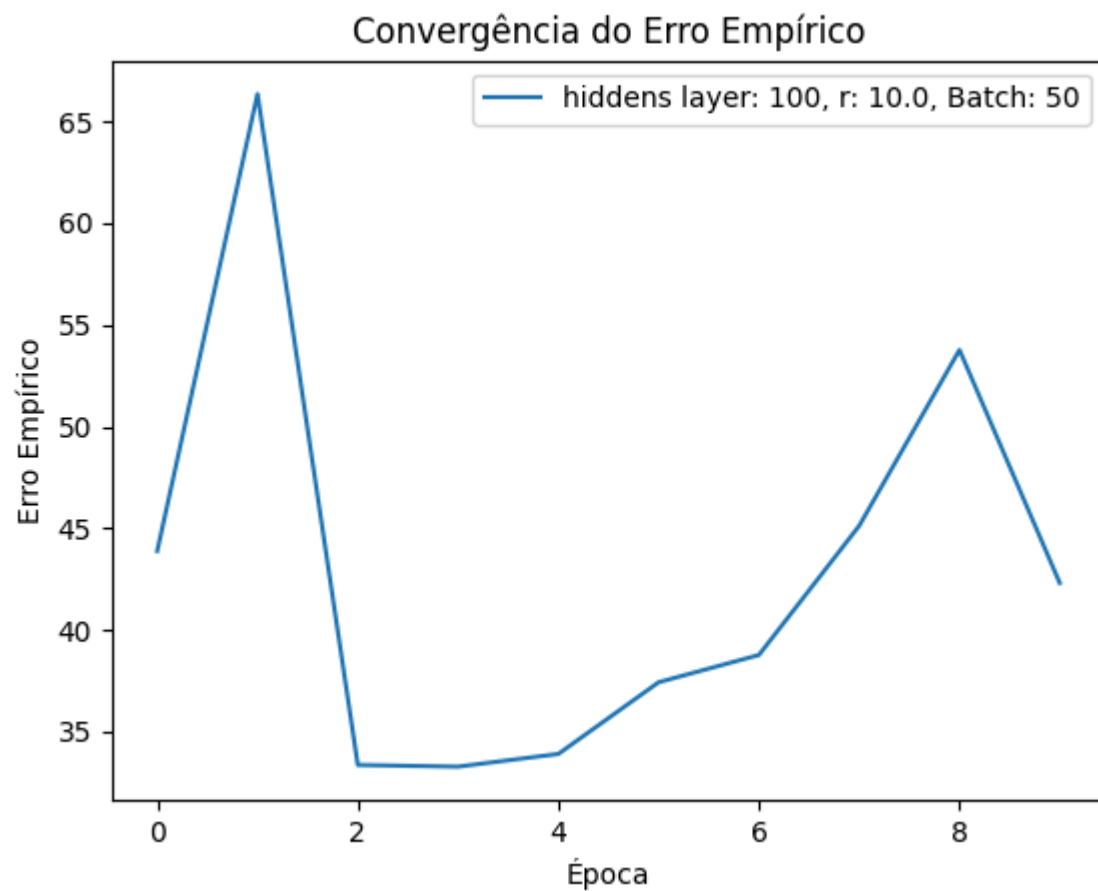
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

 Stochastic Gradient Descent - hidden\_layer: 100, r: 10, batch: 1

- Mini-Batch:



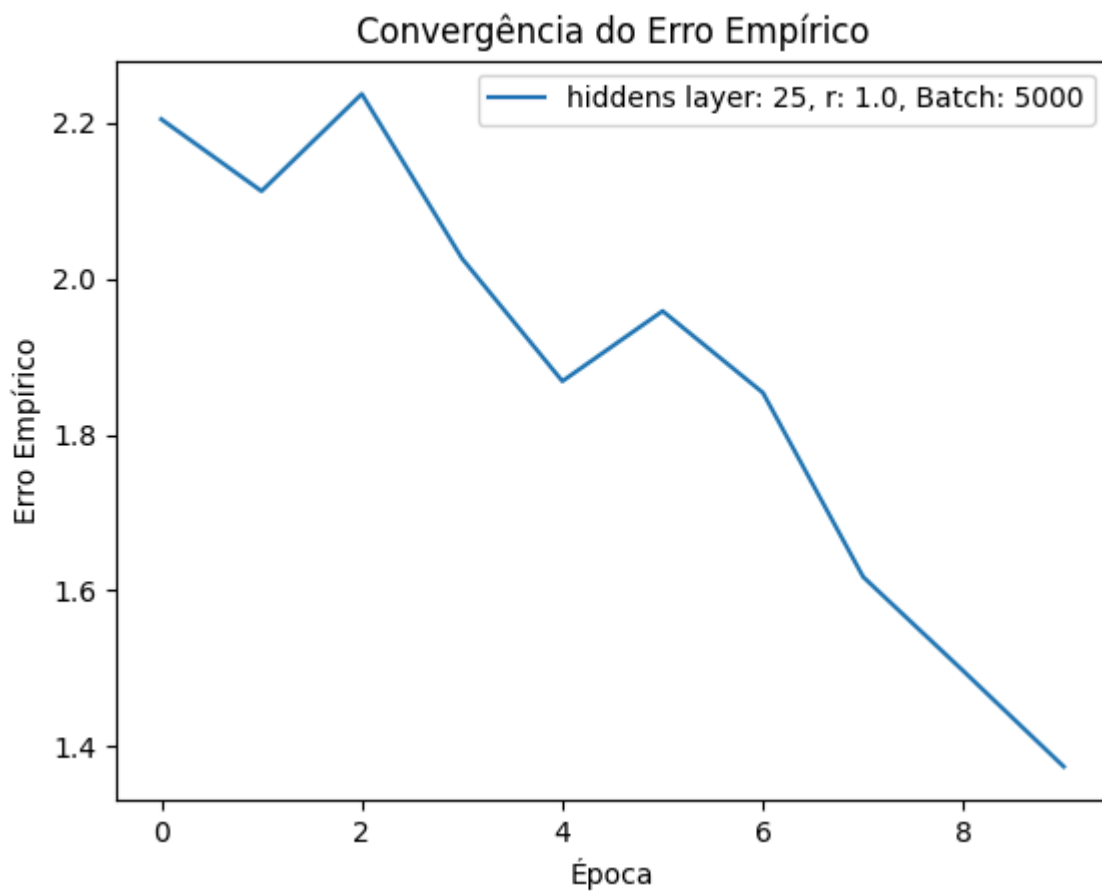
## Taxa de aprendizado: 1

---

## Com 25 Camadas Ocultas



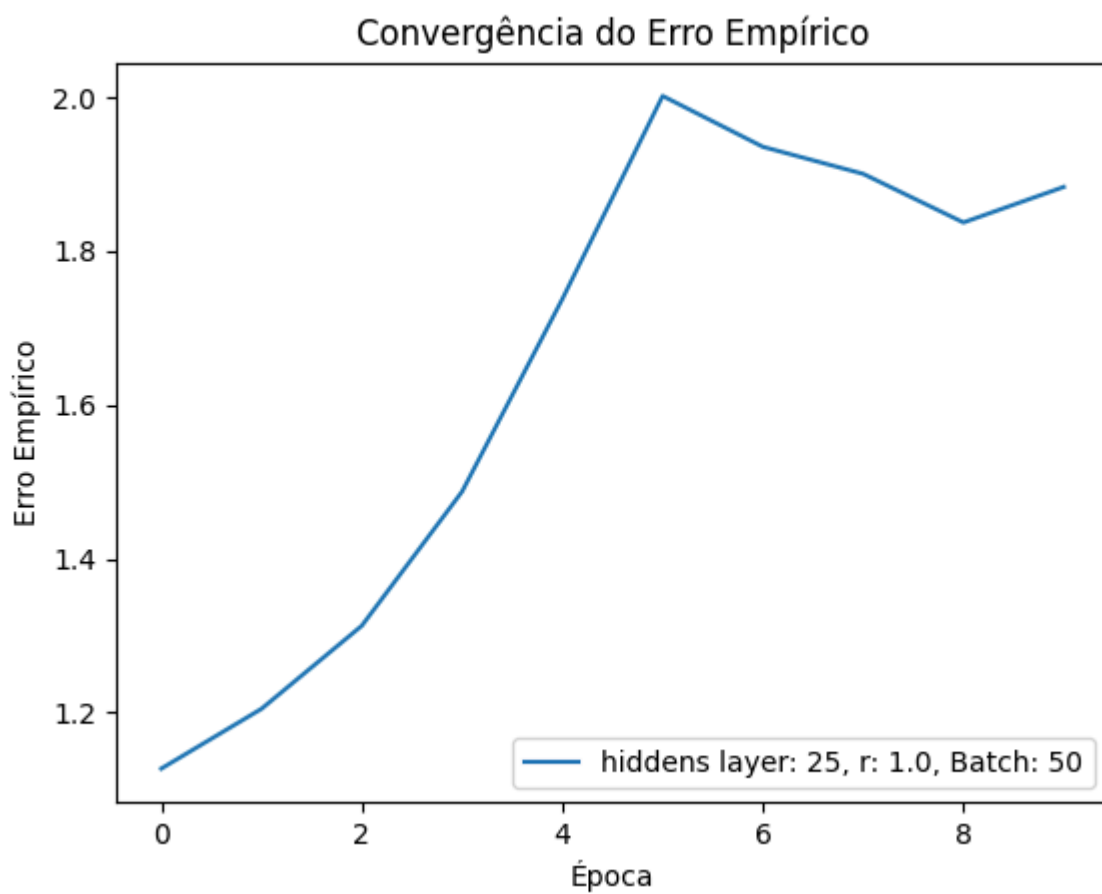
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

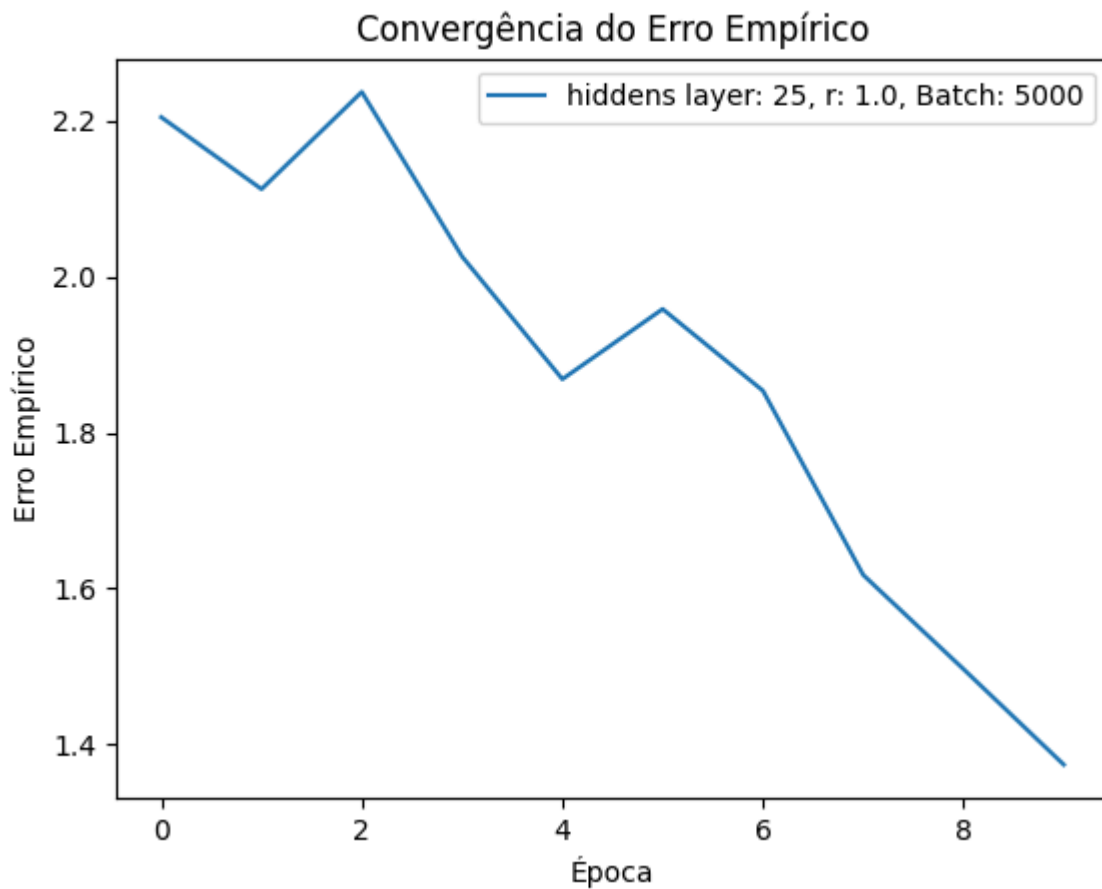
 Stochastic Gradient Descent - hidden\_layer: 25, r: 1, batch: 1

- **Mini-Batch:**




# Com 50 Camadas Ocultas

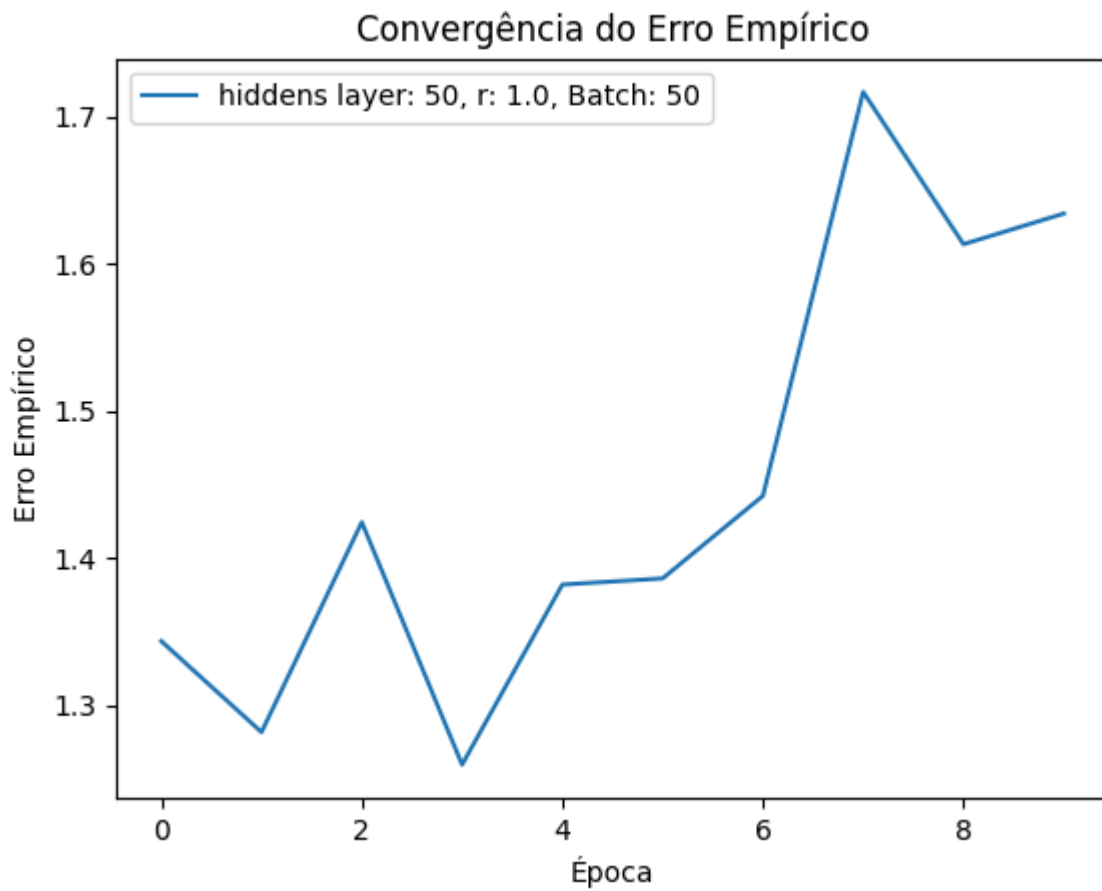
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

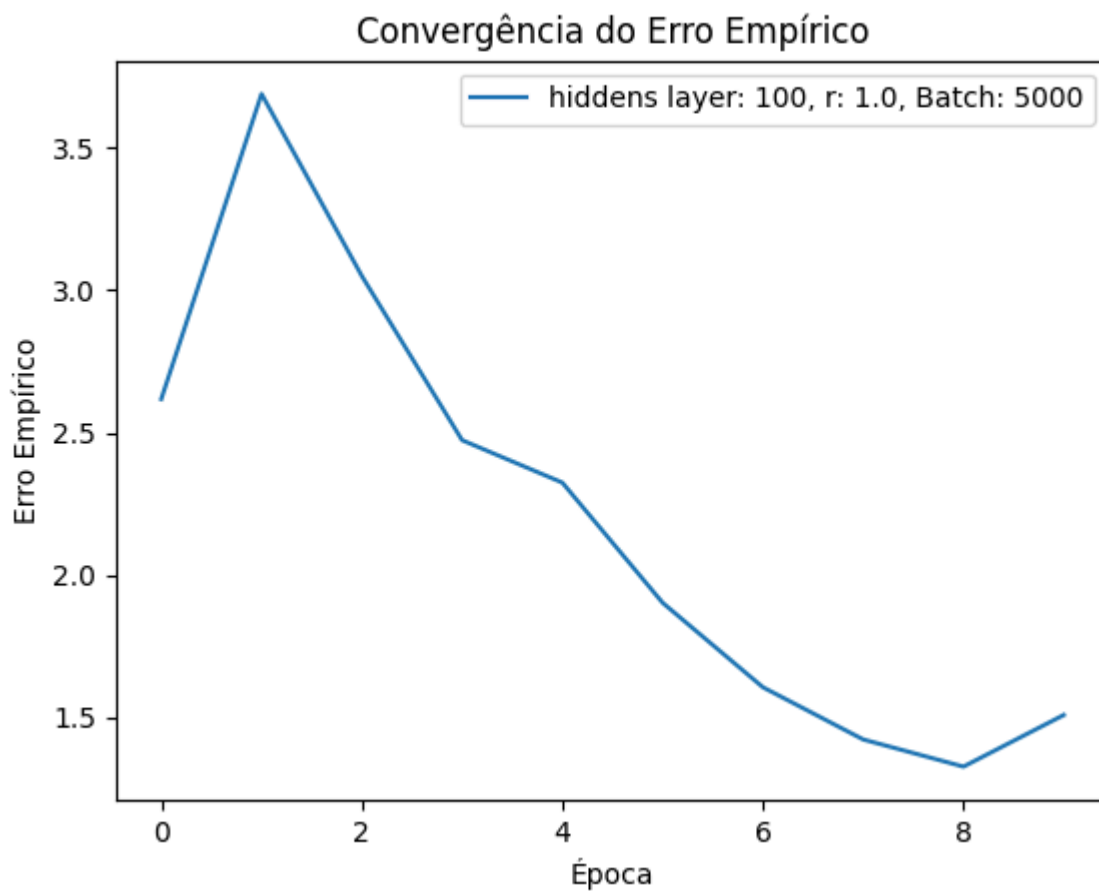
 Stochastic Gradient Descent - hidden\_layer: 50, r: 10, batch: 1

- Mini-Batch:




**Com 100 Camadas Ocultas**

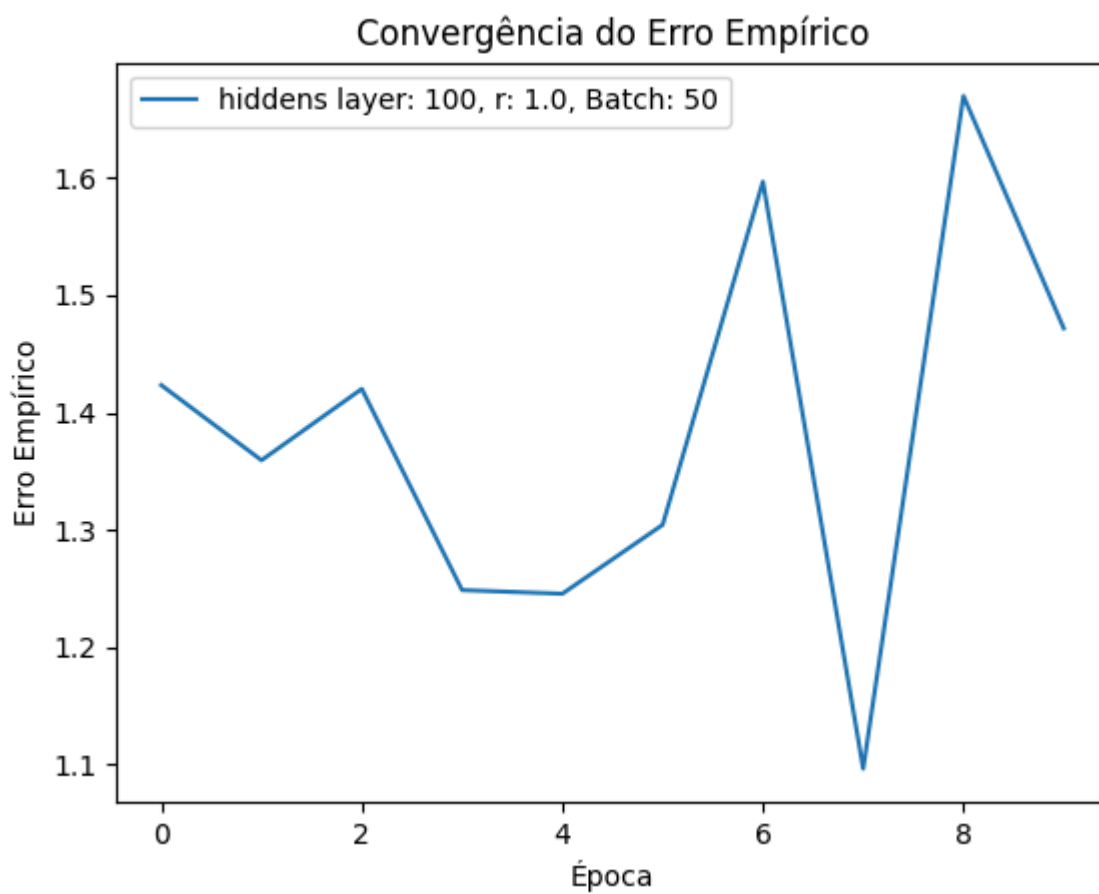
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

 Stochastic Gradient Descent - hidden\_layer: 100, r: 1, batch: 1

- **Mini-Batch:**

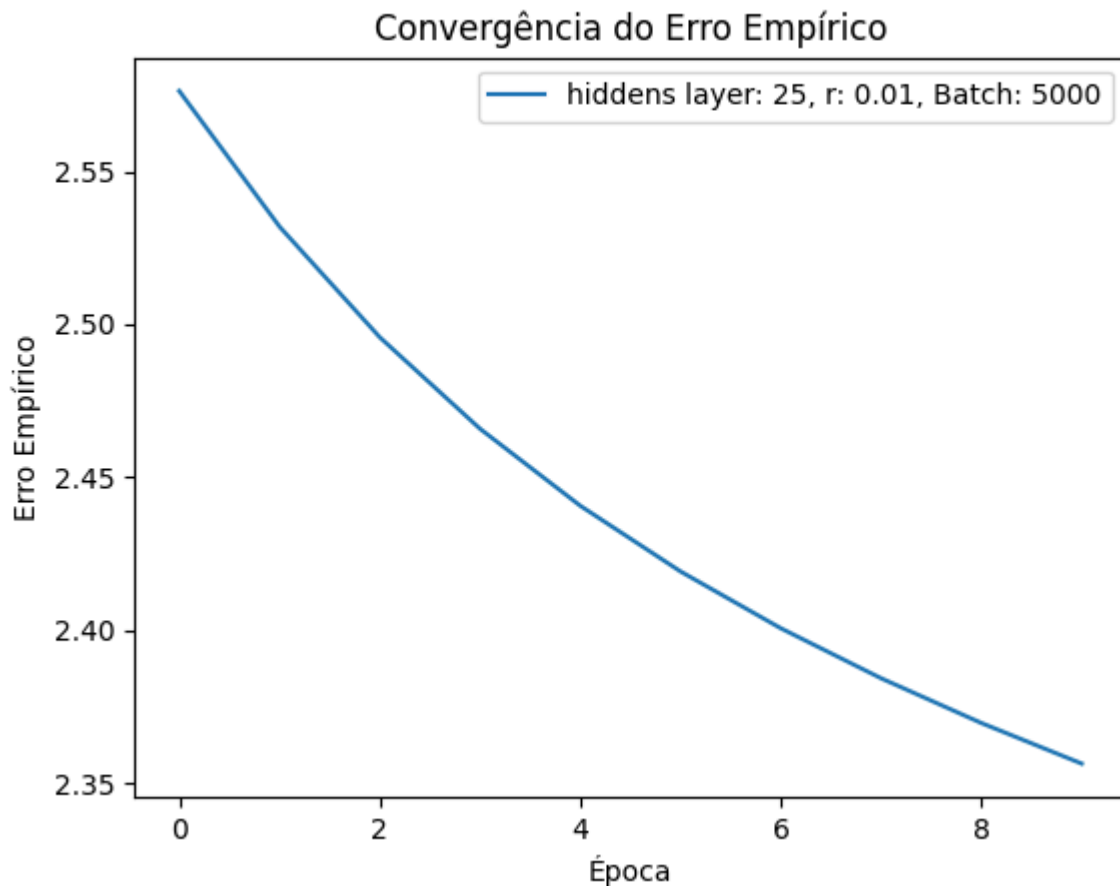


# Taxa de aprendizado: 0.01

---

## Com 25 Camadas Ocultas

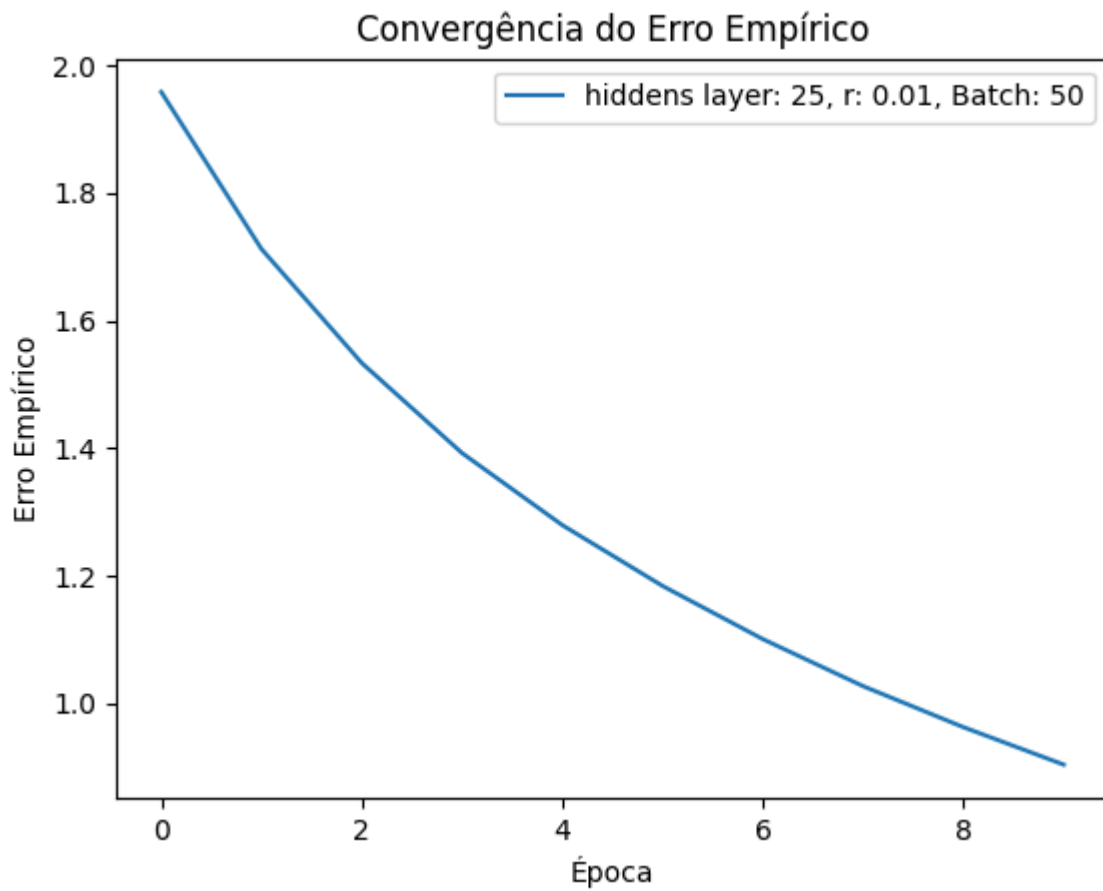
- Gradient Descent:



- Stochastic Gradient Descent:

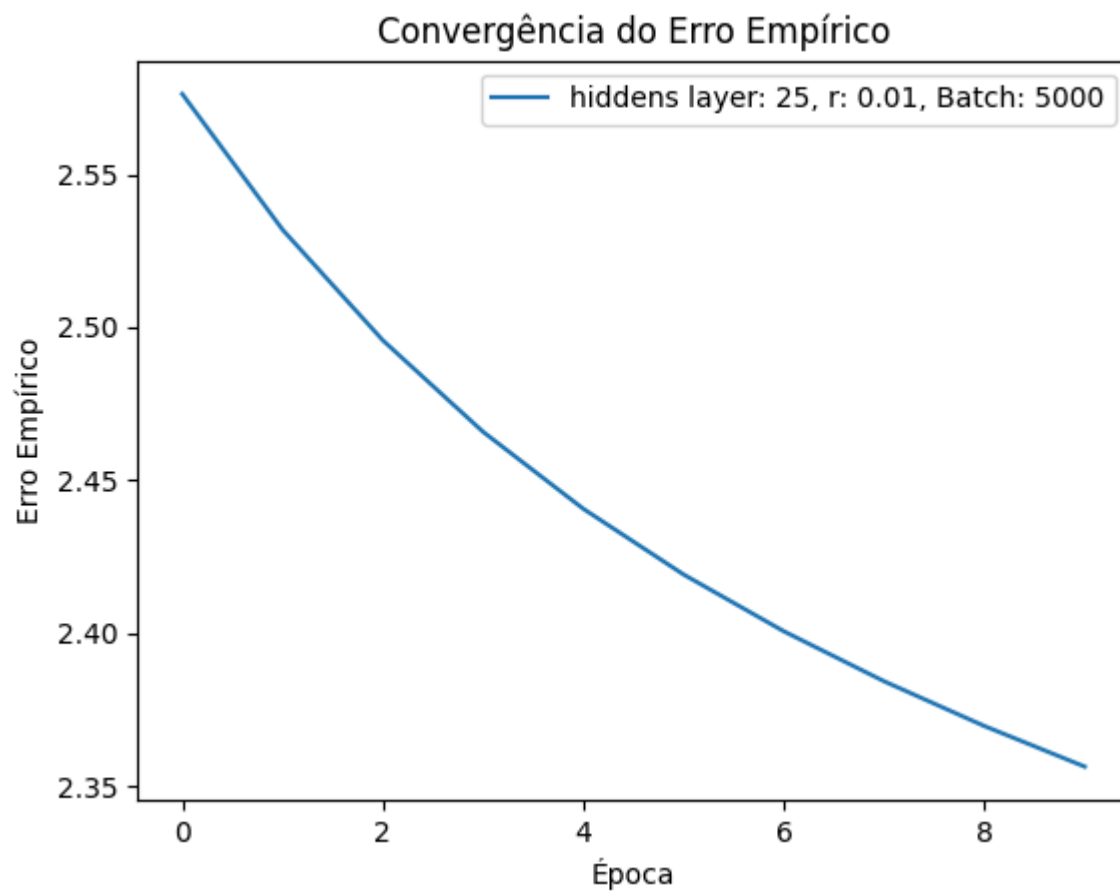
 Stochastic Gradient Descent - hidden\_layer: 25, r: 10, batch: 1

- Mini-Batch:



**Com 50 Camadas Ocultas**

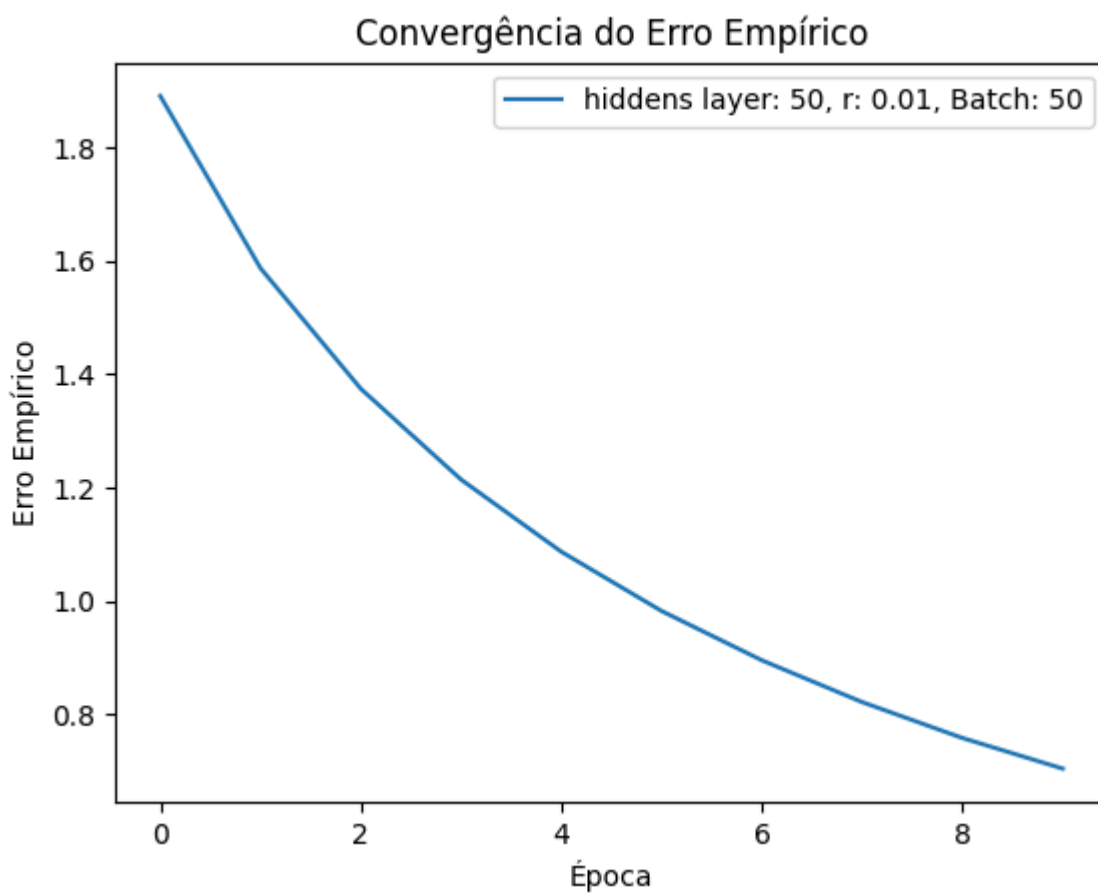
- **Gradient Descent:**



- **Stochastic Gradient Descent:**

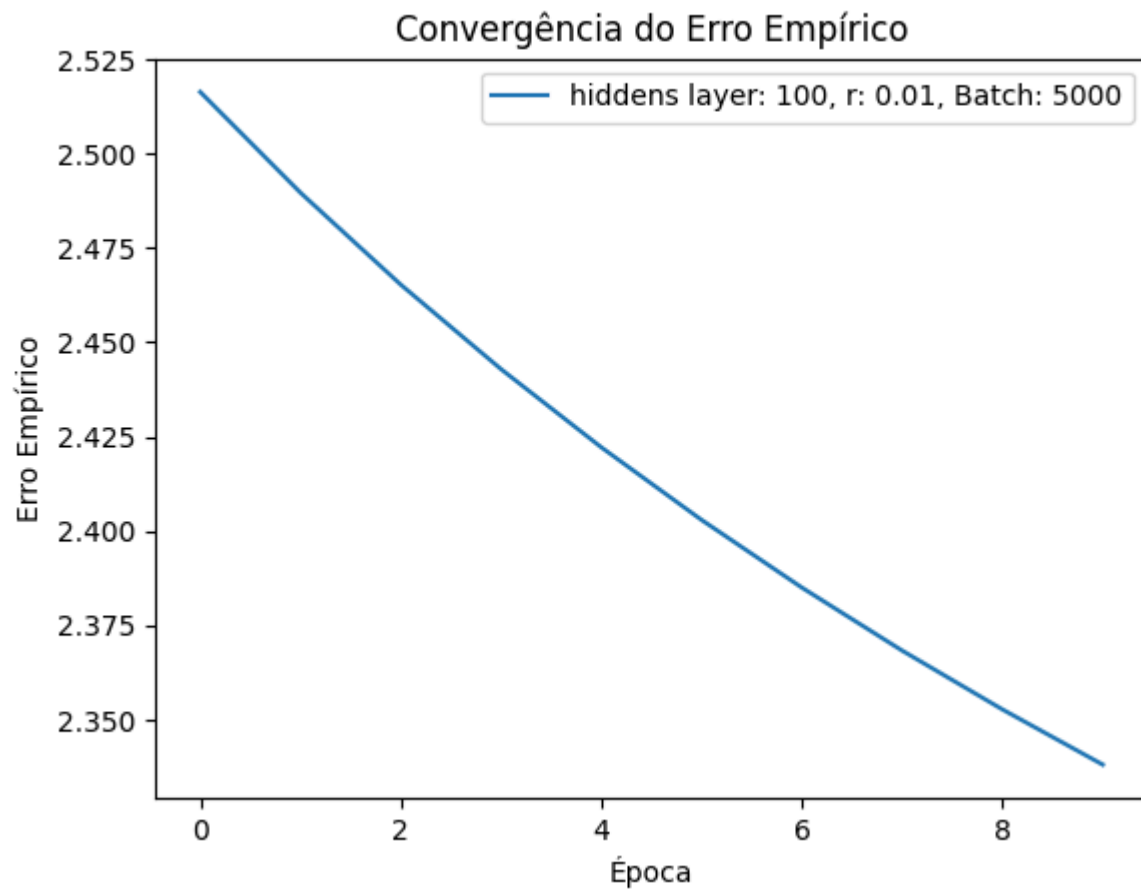
 Stochastic Gradient Descent - hidden\_layer: 50, r: 0.01, batch: 1

- **Mini-Batch:**



# Com 100 Camadas Ocultas

- **Gradient Descent:**



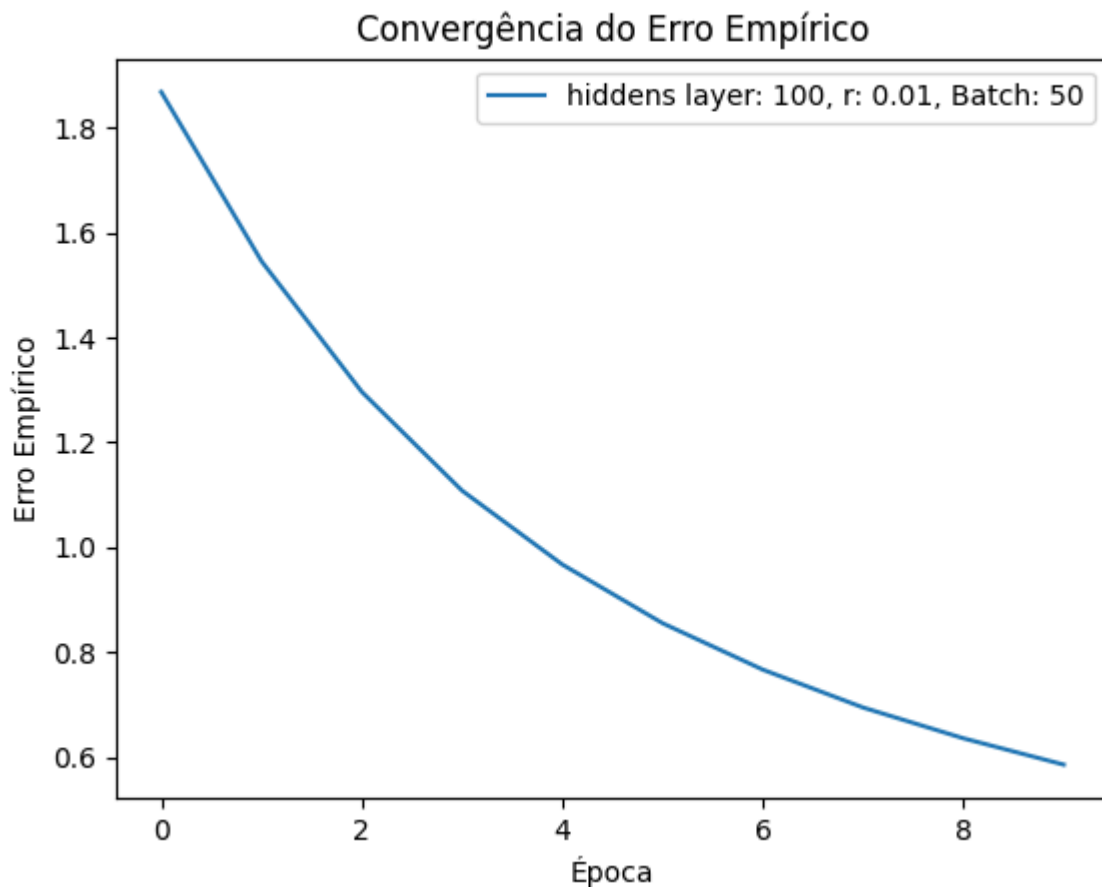
- **Stochastic Gradient Descent:**



Stochastic Gradient Descent - hidden\_layer: 100, r: 0.01, batch: 1



- **Mini-Batch:**



## Resultados das análises:

Com 25 camadas independente da taxa de aprendizado o erro foi alto e muito incerto para calculo de gradientes maiores Com 50 camadas o resultado ainda fui muito ruim para o gradiente em uma epoca só, mesmo com erro alto as coisas ficaram melhores para taxas mais baixas e gradientes com mais frequencia Para 100 camadas os graficos tenderam mais para a diminuição do erro, porém so obtive um resultado satisfatorio usando uma taxa de aprendizado baixa e gradiente mais contantes no geral gradientes mais constantes e taxa de aprendizado baixas deram resultados muito bons, o numero de camadas baixo deixou os resultados mais esquisios a taxa de aprendizado adaptativa não foi mostraado o resultado mas usando as configurações padrão do tensor os resultadaos foram incriveis mas acredito que o importante aqui não é o resultado mas sim analisar como a diferença de parametros podem e alteram os resultados.

Peço perdão pelas imagens que estão faltando mas meu computador ficou horas tentando gerar as imagens para gradientes para cada entrada mas demora muito para obtr todos. Acredito tambem que minha função pode não está tão boa para gradiente

por entradas visto os resultados esquisitos que obtive para a geração, mas é muito difícil achar o forma certa visto a demora que é para gerar