INF721

2023/2

UFV

Aprendizado em Redes Neurais Profundas

A10: Otimização

Logística

Avisos

- Teste T2: Multilayer Perceptron na próxima aula!

Última aula

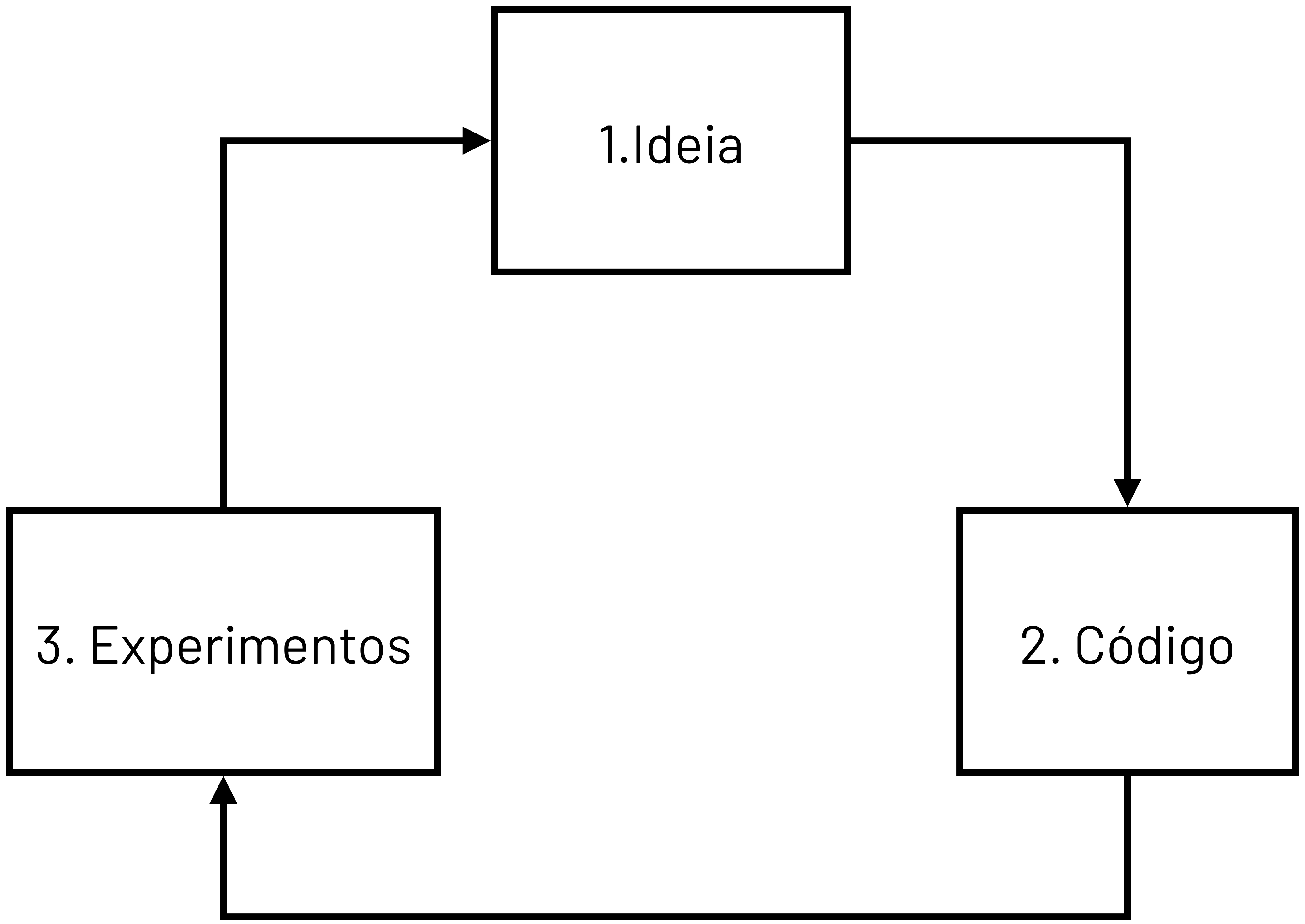
- Regularização L1
- Regularização L2
- Dropout

Plano de Aula

- Gradiente Descendente Mini-batch
- Gradiente Descendente com Momento
  - Média Móvel Exponencial
- Root Mean Squared Propagation (RMSProp)
- Adaptive Moment Estimation (Adam)

A Prática de Deep Learning

- Processo iterativo de avaliação de modelos:
  1. Ideia de modelo;
  2. Implementar e treinar o modelo;
  3. Testar o modelo.
- Funciona muito bem com altos volumes de dados (*big data*)
- O tempo de treinamento é um fator crucial para criar modelos neurais de sucesso:
  - Vetorização
  - GPUs



Vetorização

- Vetorização nos permite treinar RNAs de maneira eficiente
- Em conjuntos de dados pequenos, podemos processar uma época do gradiente descendente em tempo constante.
- Em conjuntos de dados muito grandes, isso não é possível:
  - A matriz de entrada X e consequente os pesos da RNA ((W1, b1), (W2, b2), ... , (WL, bL)) não cabem em memória "de uma vez"

Gradiente Descendente Mini-batch

Divididir o conjunto de treinamento em subconjuntos chamados mini-batches

$$X^{\{1\}} \qquad X^{\{2\}} \qquad X^{\{n//b\}}$$

$$X \quad \underbrace{x^{(1)}\, x^{(2)} \cdots x^{(32)}}\, \underbrace{x^{(33)} \cdots x^{(64)}} \cdots \underbrace{x^{(n-32)} \cdots x^{(n-1)}\, x^{(n)}}$$

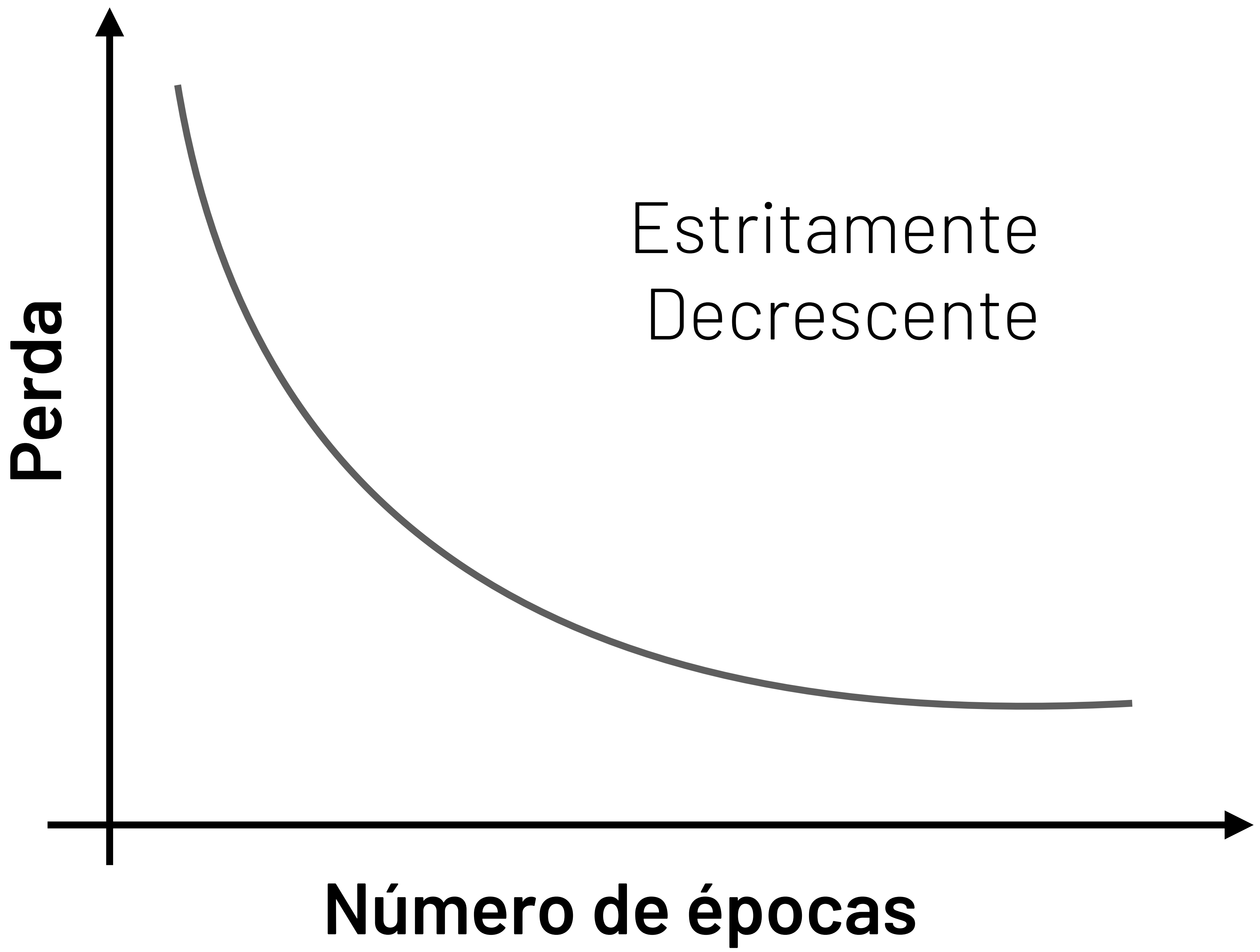
$$Y \quad \underbrace{y^{(1)}\, y^{(2)} \cdots y^{(32)}}_{Y^{\{1\}}}\, \underbrace{y^{(33)} \cdots y^{(64)}}_{Y^{\{2\}}} \cdots \underbrace{y^{(n-32)} \cdots y^{(n-1)}\, y^{(n)}}_{Y^{\{n//b\}}}$$

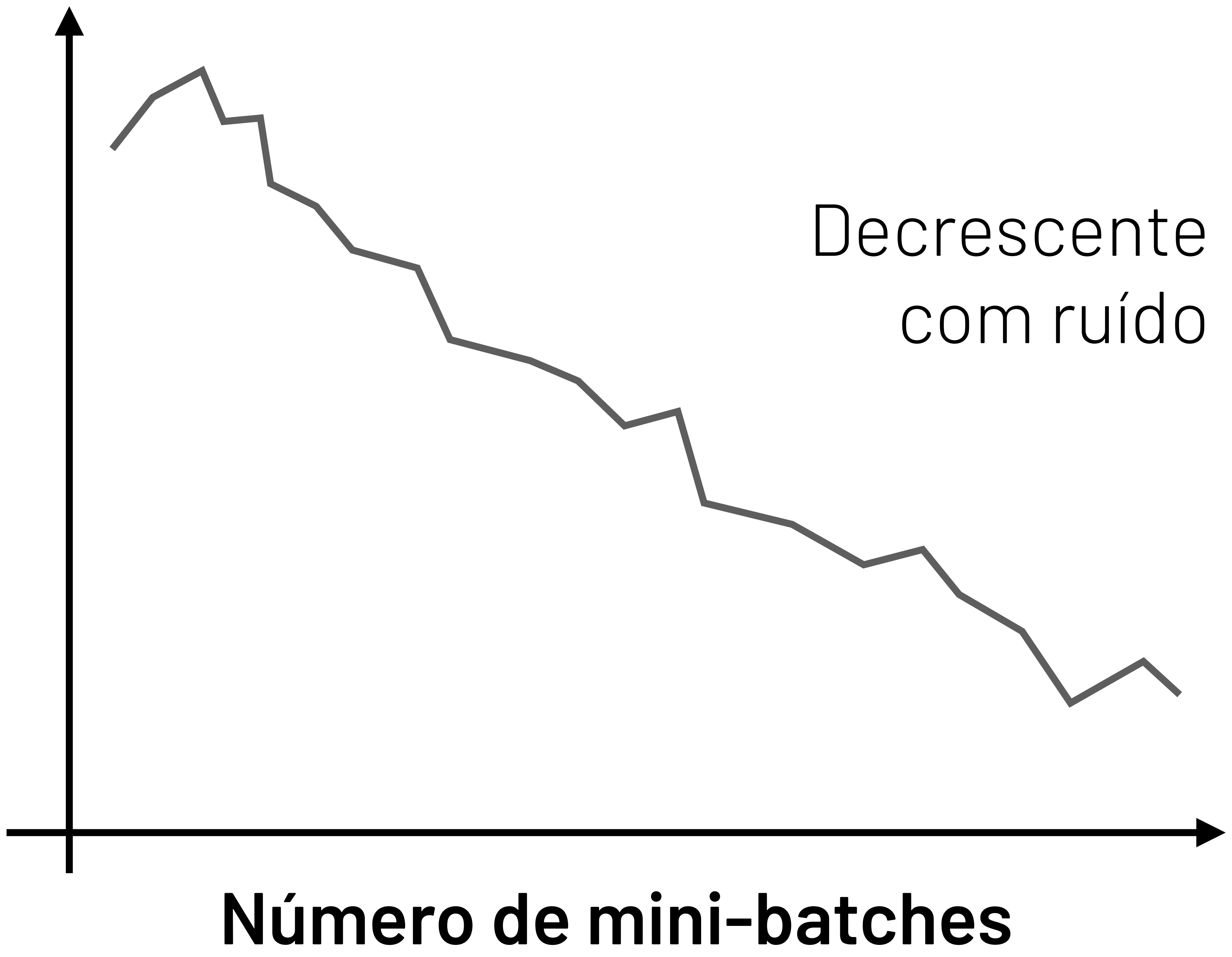
Gradiente Descendente Mini-batch

```python
n_batches = n//batch_size

# Para cada época
for e in range(n_epochs):
  # Para cada minibatch X_t
  for t in range(n_batches):
    # Propagação da entrada X_t
    Yh_t = forward_pass(X_t)
    # Cálculo de perda de Yh_t
    l_t = 1/1000 * np.sum(L(Yh_t, Y_t))
    # Retropropagação de l_t
    dW_t, db_t = backward_pass(X_t, Y_t)

    # Atualização de pesos
    W[l] = W[l] - lr * dW_t
    b[l] = b[l] - lr * db_t
```

- Calcular o erro e atualizar os pesos para cada batch $X^{\{t\}}$, ao invés do conjunto de treinamento inteiro X.
- Múltiplas atualizações de pesos por época
- Gradiente Descendente Batch (b = n)
- Gradiente Descendente Estocástico (b = 1)
- Gradiente Descendente Mini-batch (1 > b < n)

Comportamento de treinamento





Tempo de treinamento

Gradiente Descendente Batch

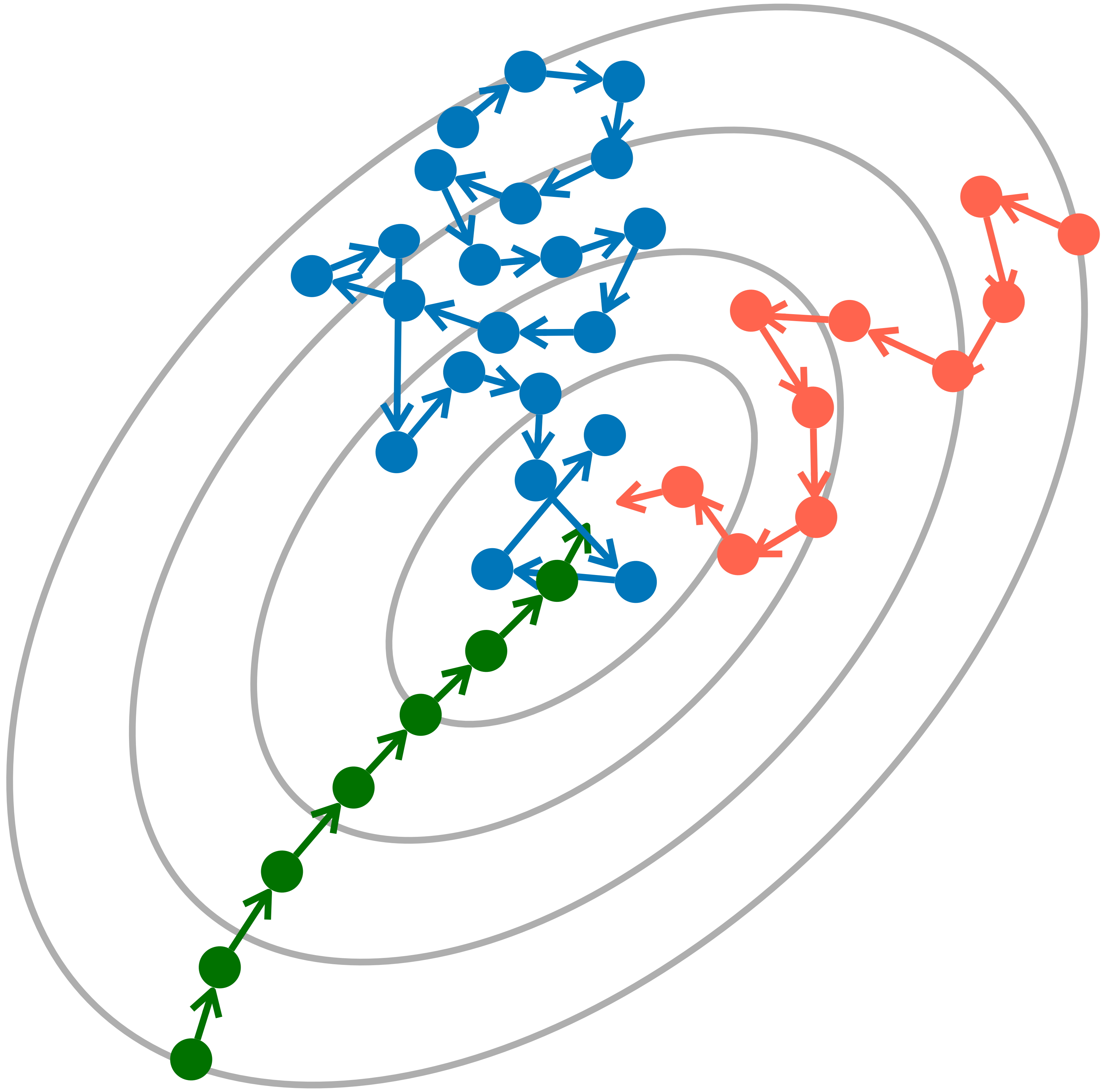
- Uma atualização de pesos por época
- Gradiente exato, porém atualização muito lenta

Gradiente Descendente Estocástico

- n atualizações de pesos por época
- Atualização muito rápida, porém gradiente com ruído
- Não utiliza vetorização

Gradiente Descendente Mini-batch (mais usado!)

- Uma atualização de pesos para cada mibi-bath X^t
- Atualização rápida com boas aproximações do gradiente



UFV

Escolhendo o tamanho do mini batch

Conjunto de treinamento pequeno

- Gradiente Descendente Batch

Conjunto de treinamento grande

- Gradiente Descendente Mini-batch
- Tamanho de mini-batch (híper-parâmetro):
  - Potência de dois
  - Cabe em memória da CPU/GPU
  - 64, 128, 256, 512, 1024, ...

Gradiente Descendente com Momento

UFV

Média Móvel

Médias móveis são métricas de média para séries temporais:

Média móvel simples: $v_t = \frac{1}{T} \cdot \sum_{t=1}^{T} x_t$

Média móvel ponderada: $v_t = \frac{1}{\sum_{t=1}^{T} w_t} \cdot \sum_{t=1}^{T} x_t \cdot w_t$

Média móvel exponencial: $v_t = \beta v_{t-1} + (1 - \beta)\theta_t$

Média Móvel Exponencial

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t \qquad \beta = 0.9$$

$$v_1 = 0.9v_0 + 0.1\theta_1 \qquad \theta_1 = 16$$

$$v_2 = 0.9v_1 + 0.1\theta_2 \qquad \theta_2 = 24$$

$$v_3 = 0.9v_2 + 0.1\theta_3 \qquad \theta_3 = 28$$

$$\ldots \qquad \ldots$$

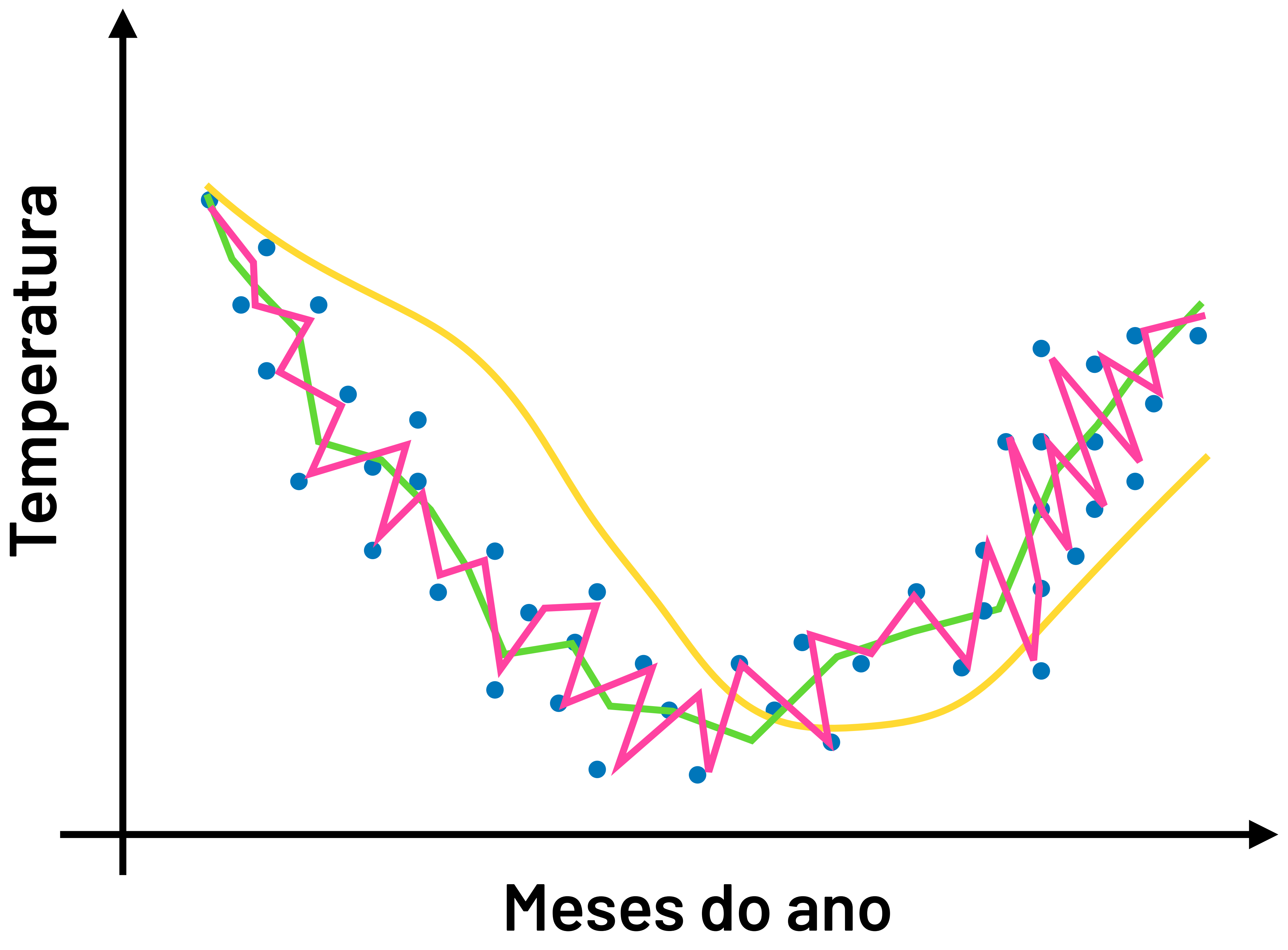
v_t é aproxidamente a média dos últimos $\dfrac{1}{1-\beta}$ dias!

$$\beta = 0.9 = \frac{1}{1+0.9} \approx 10 \text{ dias}$$

$$\beta = 0.98 = \frac{1}{1+0.98} \approx 50 \text{ dias}$$

$$\beta = 0.5 = \frac{1}{1+0.5} \approx 2 \text{ dias}$$

Quanto maior o valor de β, mais lentamente a média se adapta aos novos valores de θ_i



UFV

Média Móvel Exponencial

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

$$\begin{aligned} v_{100} &= 0.9v_{99} + 0.1\theta_{100} \\ v_{99} &= 0.9v_{98} + 0.1\theta_{99} \\ v_{98} &= 0.9v_{97} + 0.1\theta_{98} \end{aligned}$$

$$\begin{aligned} v_{100} &= 0.1\theta_{100} + 0.9v_{99} \\ &= 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9v_{98}) \\ &= 0.1\theta_{100} + 0.9(0.1\theta_{99} + 0.9(0.1\theta_{98} + 0.9v_{97})) \\ &= 0.1\theta_{100} + 0.1(0.9)\cdot\theta_{99} + 0.1(0.9)^2\cdot\theta_{98} + 0.1(0.9)^3\cdot\theta_{97} + \ldots \end{aligned}$$

A **média móvel exponencial** é uma soma ponderada por pesos que decrescem exponencialmente!

Correção de Viés (Bias Correction)

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t \longrightarrow v_t = \frac{\beta v_{t-1} + (1-\beta)\theta_t}{1-\beta^t}$$

Quanto maior o valor de β, mais lentamente a média se adapta aos novos valores de θ_i

$$v_0 = 0$$
$$v_1 = \cancel{0.98 v_0} + 0.02\theta_1$$
$$v_2 = 0.98 v_1 + 0.02\theta_2$$
$$= 0.98 \cdot 0.02\theta_1 + 0.02\theta_2$$
$$= 0.00196\theta_1 + 0.02\theta_2$$

$$v_2 = \frac{0.00196\theta_1 + 0.02\theta_2}{1 - 0.98^2}$$

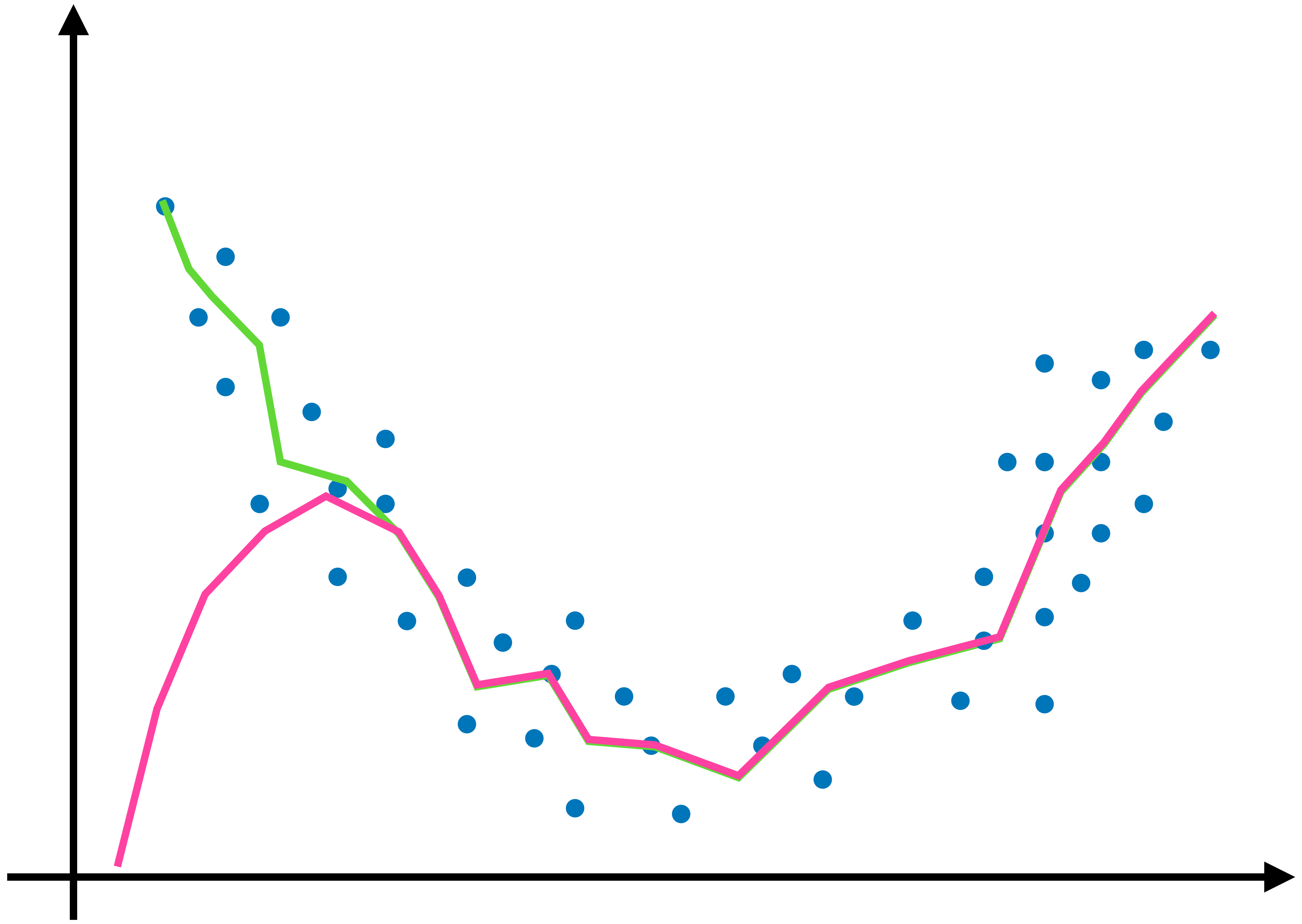
$$v_2 = \frac{0.00196\theta_1 + 0.02\theta_2}{0.0396}$$

Média ponderada!

$\theta_1 = 16 \quad v_1 = 0.32$

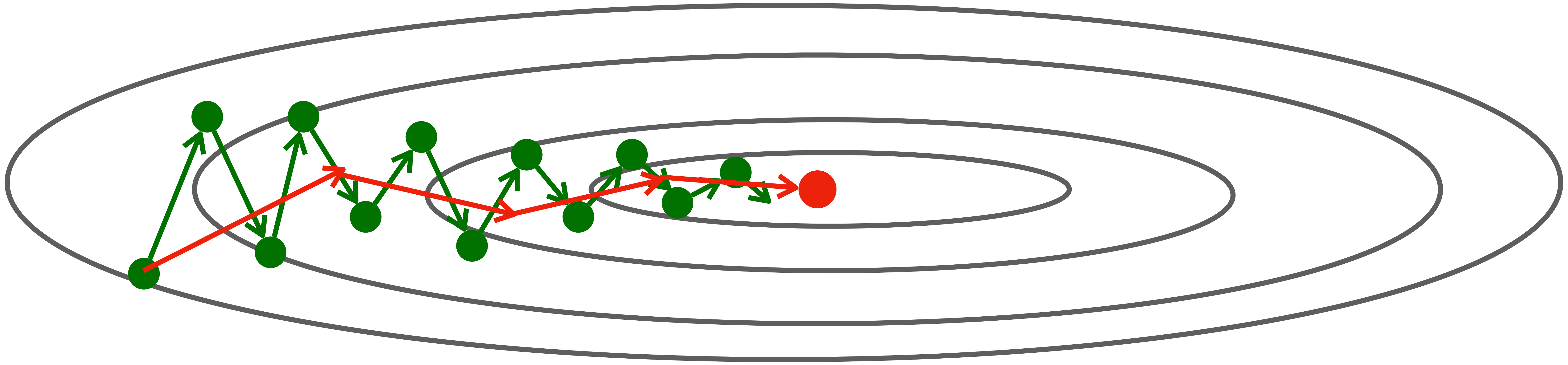
$\theta_2 = 24 \quad v_2 = 0.51136$

Inicialmente, os valores das médias são estimativas muito ruins!

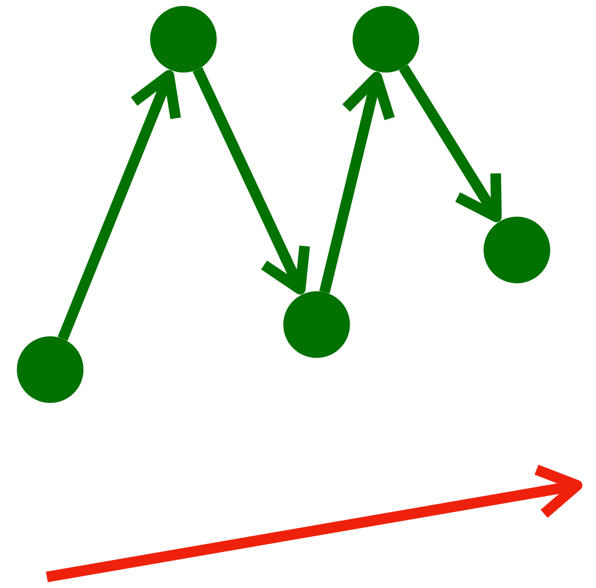


UFV

Gradiente Descendente com Momento



Média zero na direção vertical!



Gradiente Descendente Batch

Taxa de aprendizado relativamente baixa para evitar divergência

Ideal

↕ Taxa de aprendizado baixa no eixo vertical

↔ Taxa de aprendizado alta no eixo horizontal

Gradiente Descendente com Momento

$$dw, db = backward(X^t)$$

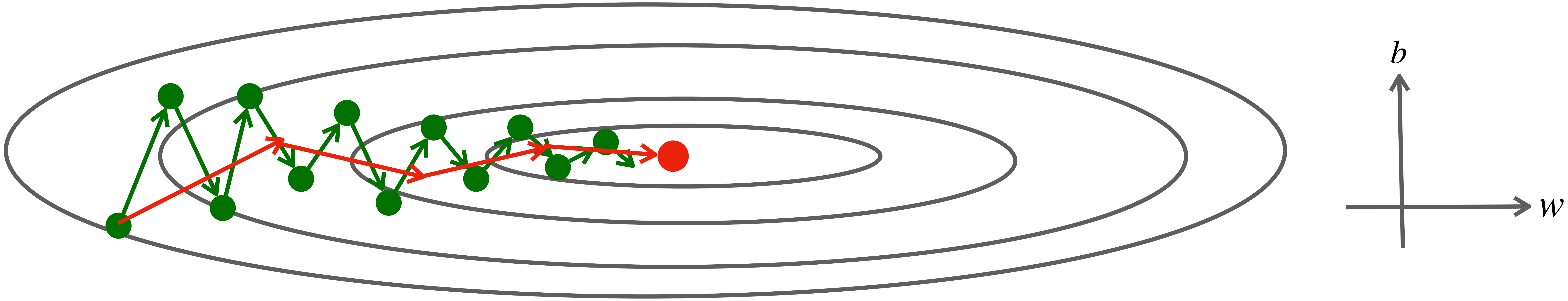
$$Vdw = \beta \cdot Vdw + (1 - \beta)dw$$

$$Vdb = \beta \cdot Vdb + (1 - \beta)db$$

$$W^{[l]} = W^{[l]} - \alpha Vdw$$

$$b^{[l]} = b^{[l]} - \alpha Vdb$$

Root Mean Squared Propagation (RMSProp)



Gradiente Descendente Batch
Taxa de aprendizado relativamente baixa para evitar divergência

Ideal

↕ Taxa de aprendizado baixa no eixo vertical

↔ Taxa de aprendizado alta no eixo horizontal

RMSProp

$$dw, db = backward(X^t)$$

$$Sdw = \beta_2 \cdot Sdw + (1 - \beta_2)\underline{dw^2}$$ Valores esperados pequenos

$$Sdb = \beta_2 \cdot Sdb + (1 - \beta_2)\underline{db^2}$$ Valores esperados grandes

$$w = w - \alpha \frac{dw}{\sqrt{\underline{Sdw}}}$$ Divisão por um número pequeno

$$b = b - \alpha \frac{db}{\underline{\sqrt{Sdb}}}$$ Divisão por um número grande

UFV

Adaptive Moment Estimation (Adam)

O Adam combina o RMSProp e momento

$$dw, db = backward(X^t)$$

$$Vdw = \beta_1 \cdot Vdw + (1-\beta_1)dw, \quad Vdb = \beta_1 \cdot Vdb + (1-\beta_1)db$$ Momento

$$Sdw = \beta_2 \cdot Sdw + (1-\beta_2)dw^2, \quad Sdb = \beta_2 \cdot Sdb + (1-\beta_2)db^2$$ RMSProp

$$Vdw = \frac{Vdw}{1-\beta_1^t}, \quad Vdb = \frac{Vdb}{1-\beta_1^t}$$

$$Sdw = \frac{Sdw}{1-\beta_2^t}, \quad Sdb = \frac{Sdb}{1-\beta_2^t}$$

$$w = w - \alpha \frac{Vdw}{\sqrt{Sdw}}$$

$$b = b - \alpha \frac{Vdb}{\sqrt{Sdb}}$$

Recomendações de valores para os híper-parâmetros:

$$\beta_1 = 0.9$$

$$\beta_2 = 0.999$$

UFV

Próxima aula

A11: Pytorch Autograd

Aula prática sobre o framework Pytorch, com enfoque no seu processo de derivação automática.