REDES NEURAIS ARTIFICIAIS

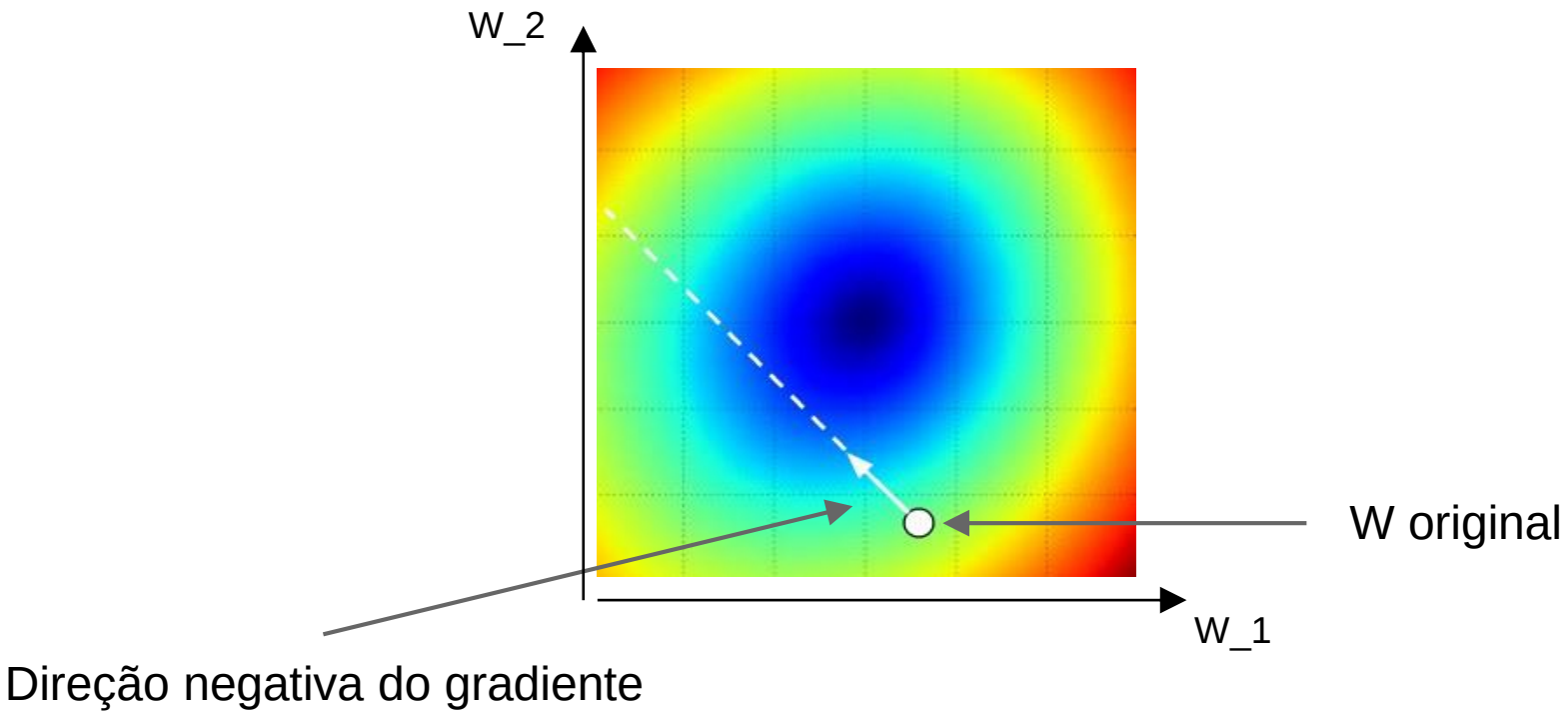
GRADIENTE DESCENDENTE ESTOCÁSTICO (SGD)

Zenilton K. G. Patrocínio Jr
zenilton@pucminas.br

Método do Gradiente (ou Descida Mais Íngreme)

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```



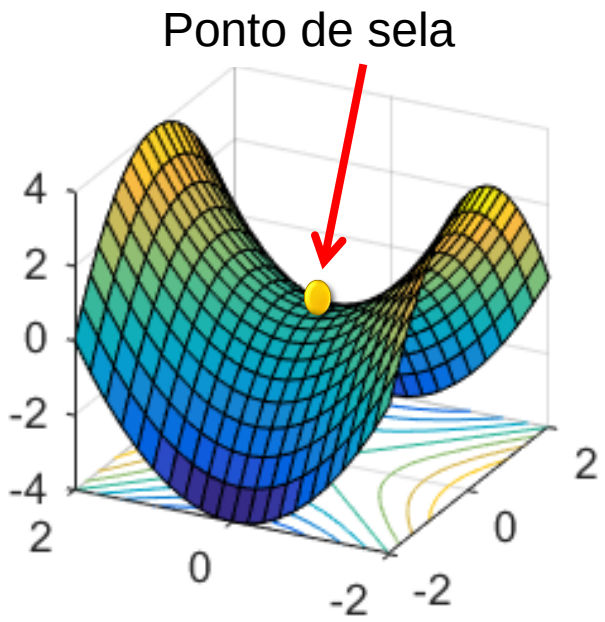
Ressalva: Pontos de Sela

Seguir a direção contrária ao gradiente deve levar a uma perda mínima

Ressalva: Pontos de Sela

Seguir a direção contrária ao gradiente deve levar a uma perda mínima

Mas pode demorar muito tempo. Uma razão é a presença de pontos de sela, em que o gradiente também desaparece

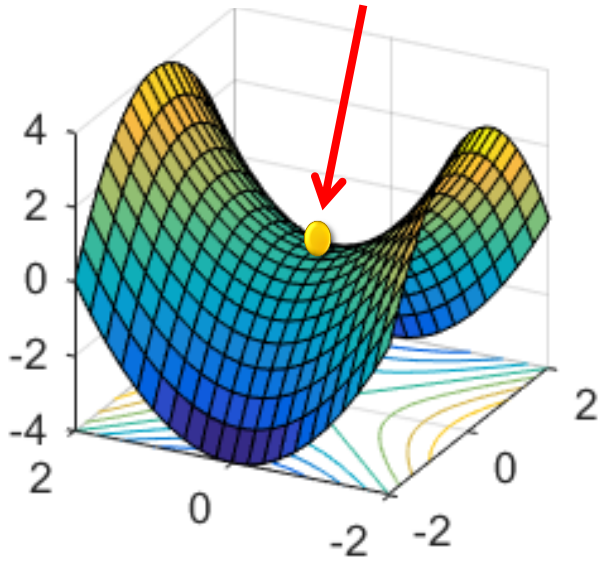


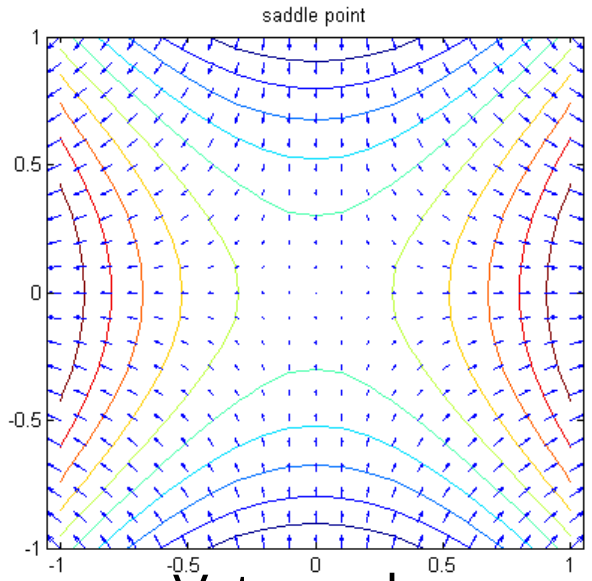
Ressalva: Pontos de Sela

Seguir a direção contrária ao gradiente deve levar a uma perda mínima

Mas pode demorar muito tempo. Uma razão é a presença de pontos de sela, em que o gradiente também desaparece

Ponto de sela



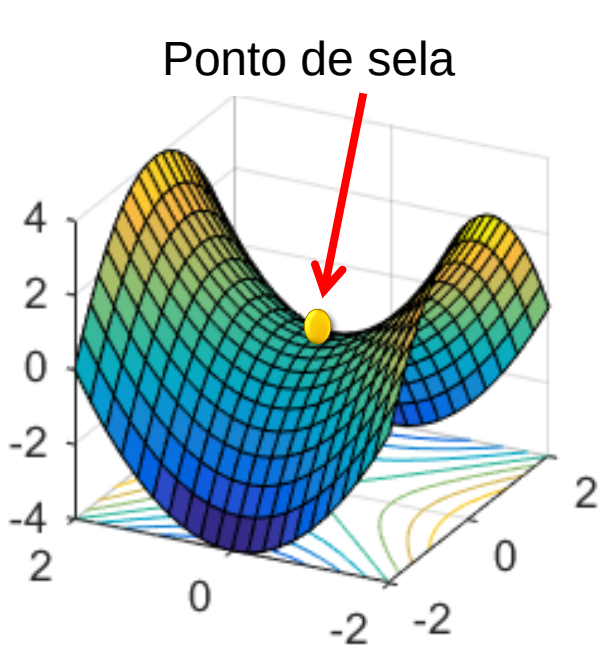


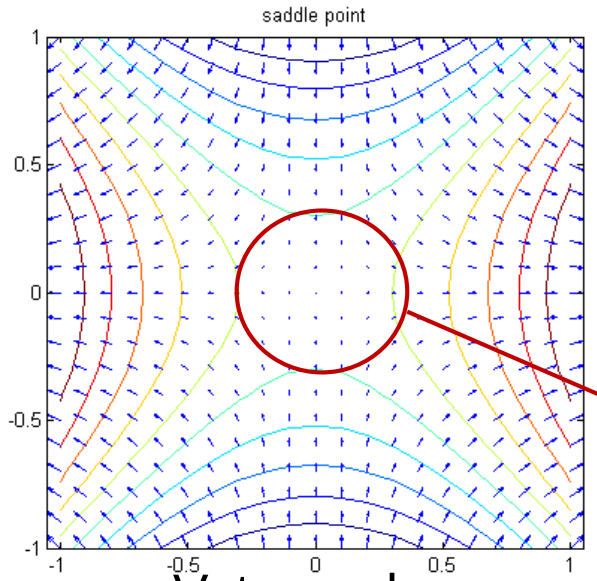
Vetores de Gradiente

Ressalva: Pontos de Sela

Seguir a direção contrária ao gradiente deve levar a uma perda mínima

Mas pode demorar muito tempo. Uma razão é a presença de pontos de sela, em que o gradiente também desaparece



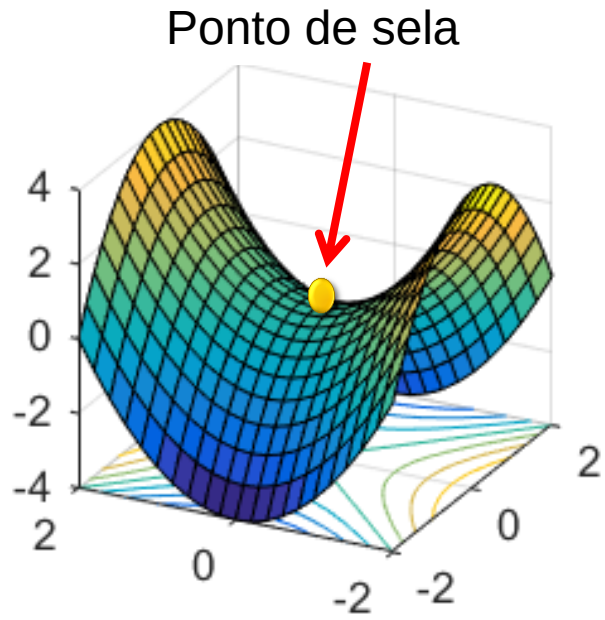


Vetores de Gradiente

Gradientes são muito pequenos aqui – ocasionando progresso muito lento

Ressalva: Pontos de Sela

Existem várias evidências de que a maioria dos extremos da função de perda (ou zeros do gradiente da função de perda $\nabla_W L$) para redes neurais profundas são de fato pontos de sela



Veja por exemplo

Dauphin et al. “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization” arXiv 1406.2572, 2014.

Ressalva : Eficiência

A perda total L é dada pela soma das perdas para todos os itens de dados

$$L = \sum_{i=1}^{N} L(x_i, y_i, W)$$

Ressalva : Eficiência

A perda total L é dada pela soma das perdas para todos os itens de dados

$$L = \sum_{i=1}^{N} L(x_i, y_i, W)$$

assim o gradiente da perda total é dado por

$$\frac{dL}{dW} = \sum_{i=1}^{N} \frac{dL}{dW}(x_i, y_i, W)$$

Ressalva : Eficiência

A perda total L é dada pela soma das perdas para todos os itens de dados

$$L = \sum_{i=1}^{N} L(x_i, y_i, W)$$

assim o gradiente da perda total é dado por

$$\frac{dL}{dW} = \sum_{i=1}^{N} \frac{dL}{dW}(x_i, y_i, W)$$

Portanto, calcular o gradiente em relação a W exige uma **passagem completa pelo conjunto de dados**

Ressalva : Eficiência

A perda total L é dada pela soma das perdas para todos os itens de dados

$$L = \sum_{i=1}^{N} L(x_i, y_i, W)$$

assim o gradiente da perda total é dado por

$$\frac{dL}{dW} = \sum_{i=1}^{N} \frac{dL}{dW}(x_i, y_i, W)$$

Portanto, calcular o gradiente em relação a W exige uma **passagem completa pelo conjunto de dados**

A obtenção do valor mínimo da função de perda pode exigir milhões de passos de gradiente e, portanto, se torna muito caro

Processamento em Lotes (*Minibatches*)

Em vez de se calcular um gradiente sobre todo o conjunto de dados, pode-se calculá-lo usando um subconjunto de tamanho fixo de amostras de dados chamado ***minibatch*** (seu tamanho é tipicamente 32, 64, 128, 256,…)

Processamento em Lotes (*Minibatches*)

Em vez de se calcular um gradiente sobre todo o conjunto de dados, pode-se calculá-lo usando um subconjunto de tamanho fixo de amostras de dados chamado ***minibatch*** (seu tamanho é tipicamente 32, 64, 128, 256,…)

O ***minibatch*** é idealmente uma amostra aleatória de tamanho m do conjunto de dados (na prática, podem ser apenas m amostras consecutivas)

Processamento em Lotes (*Minibatches*)

Em vez de se calcular um gradiente sobre todo o conjunto de dados, pode-se calculá-lo usando um subconjunto de tamanho fixo de amostras de dados chamado ***minibatch*** (seu tamanho é tipicamente 32, 64, 128, 256,…)

O ***minibatch*** é idealmente uma amostra aleatória de tamanho m do conjunto de dados (na prática, podem ser apenas m amostras consecutivas)

Então se calcula o gradiente da seguinte forma: (N o tamanho do conjunto de dados, m o tamanho do ***minibatch***)

$$g^{(t)} = \frac{1}{m} \sum_{j = i_1,\ldots,i_m \in \{1,\ldots,N\}} \nabla_W L\left(x_j, y_j, W\right)$$

Gradiente Descendente Estocástico (*SGD*)

Considerando $W^{(t)}$ como a matriz de pesos na iteração t, temos que:

$$W^{(t+1)} = W^{(t)} - \alpha \, g^{(t)}$$

Gradiente Descendente Estocástico (*SGD*)

Considerando $W^{(t)}$ como a matriz de pesos na iteração t, temos que:

$$W^{(t+1)} = W^{(t)} - \alpha \, g^{(t)}$$

Esse método é chamado de ***gradiente descendente estocástico*** (SGD). O SGD e suas variantes são usadas quase universalmente em treinamento de redes profundas

Gradiente Descendente Estocástico (*SGD*)

Considerando $W^{(t)}$ como a matriz de pesos na iteração t, temos que:

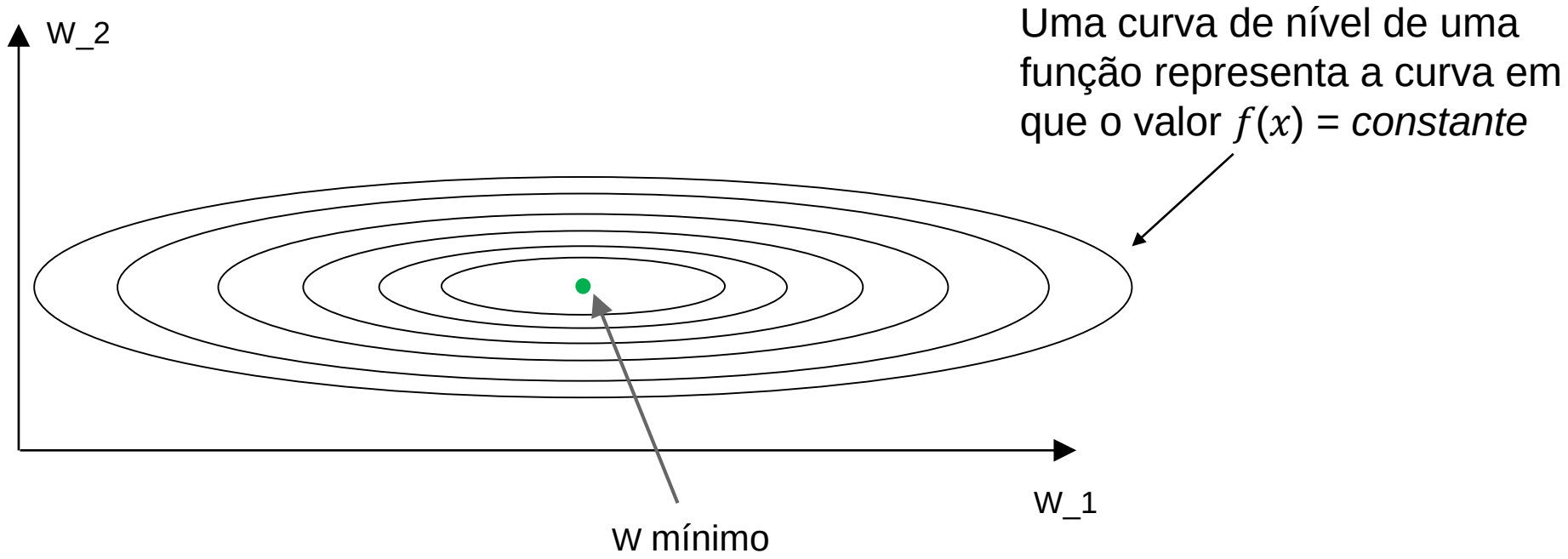
$$W^{(t+1)} = W^{(t)} - \alpha \, g^{(t)}$$

Esse método é chamado de ***gradiente descendente estocástico*** (SGD). O SGD e suas variantes são usadas quase universalmente em treinamento de redes profundas

O SGD usa $g^{(t)}$, isto é o gradiente de um ***minibatch***, no lugar do gradiente sobre o conjunto de dados completo e é dito “estocástico” pois o gradiente de um ***minibatch*** é calculado sobre uma amostra do conjunto de dados

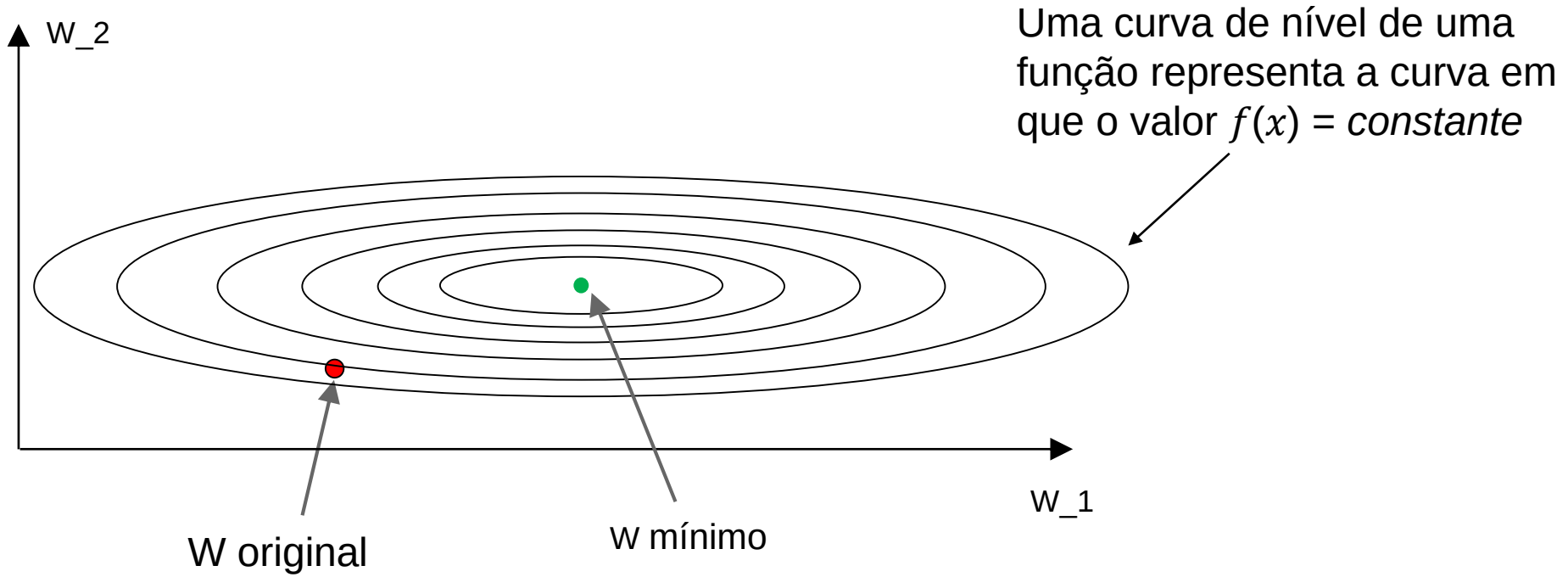
Gradiente Descendente Estocástico (*SGD*)

O gradiente da função em um ponto é sempre perpendicular a curva de nível naquele ponto



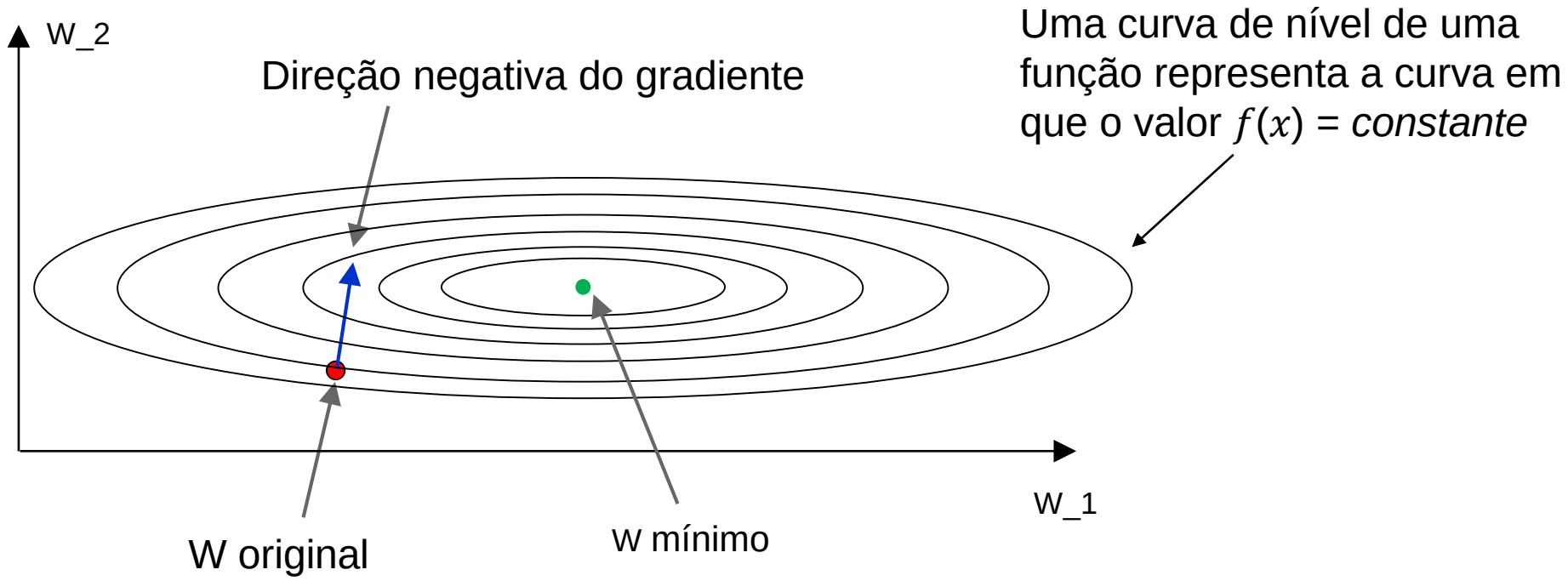
Gradiente Descendente Estocástico (*SGD*)

O gradiente da função em um ponto é sempre perpendicular a curva de nível naquele ponto

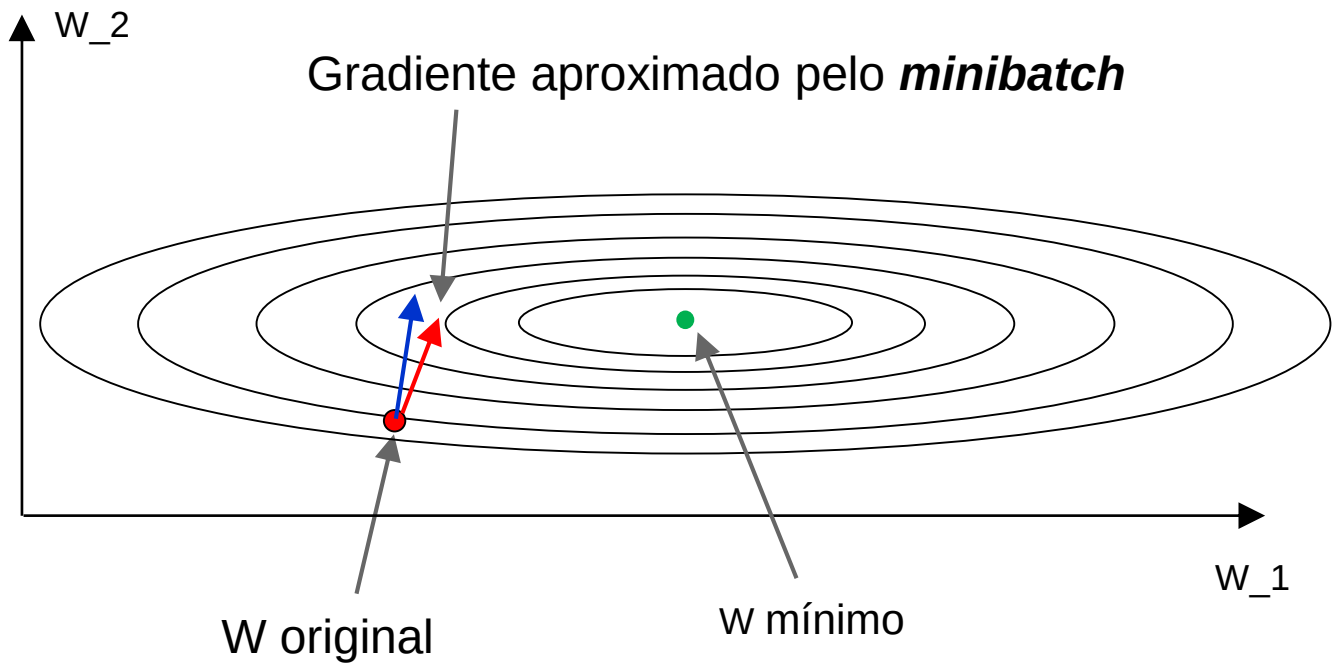


Gradiente Descendente Estocástico (*SGD*)

O gradiente da função em um ponto é sempre perpendicular a curva de nível naquele ponto

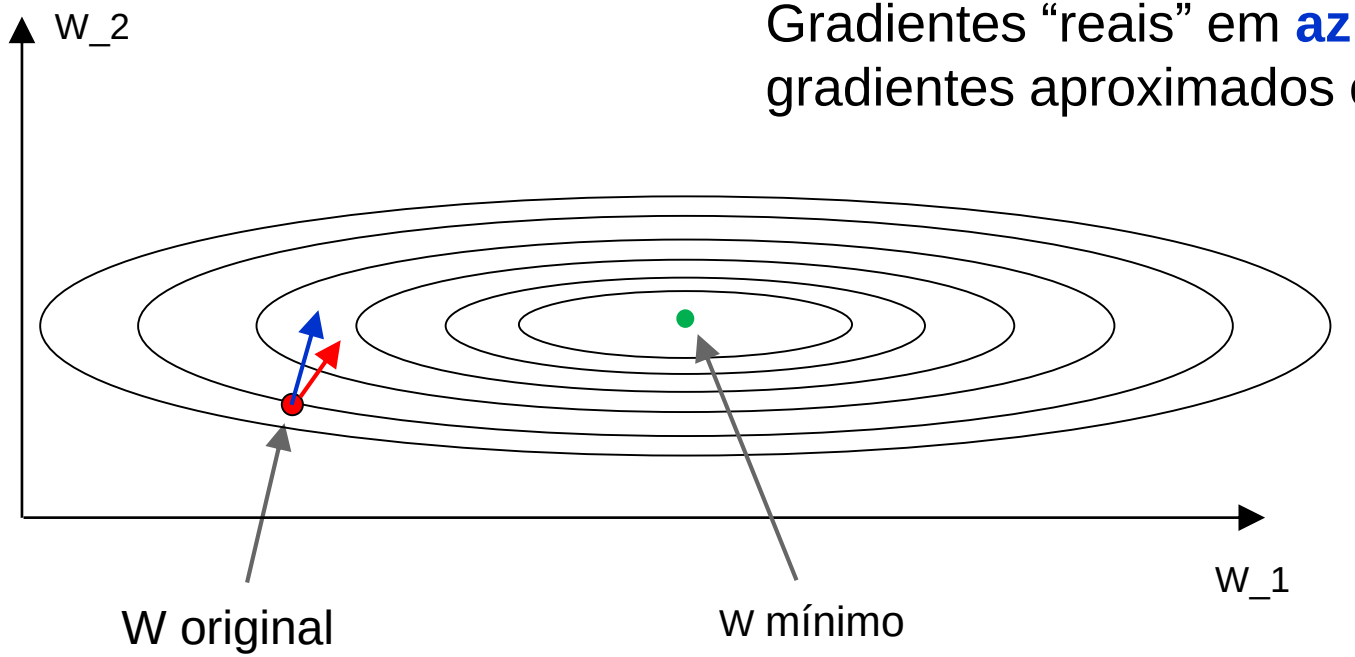


Gradiente Descendente Estocástico (*SGD*)



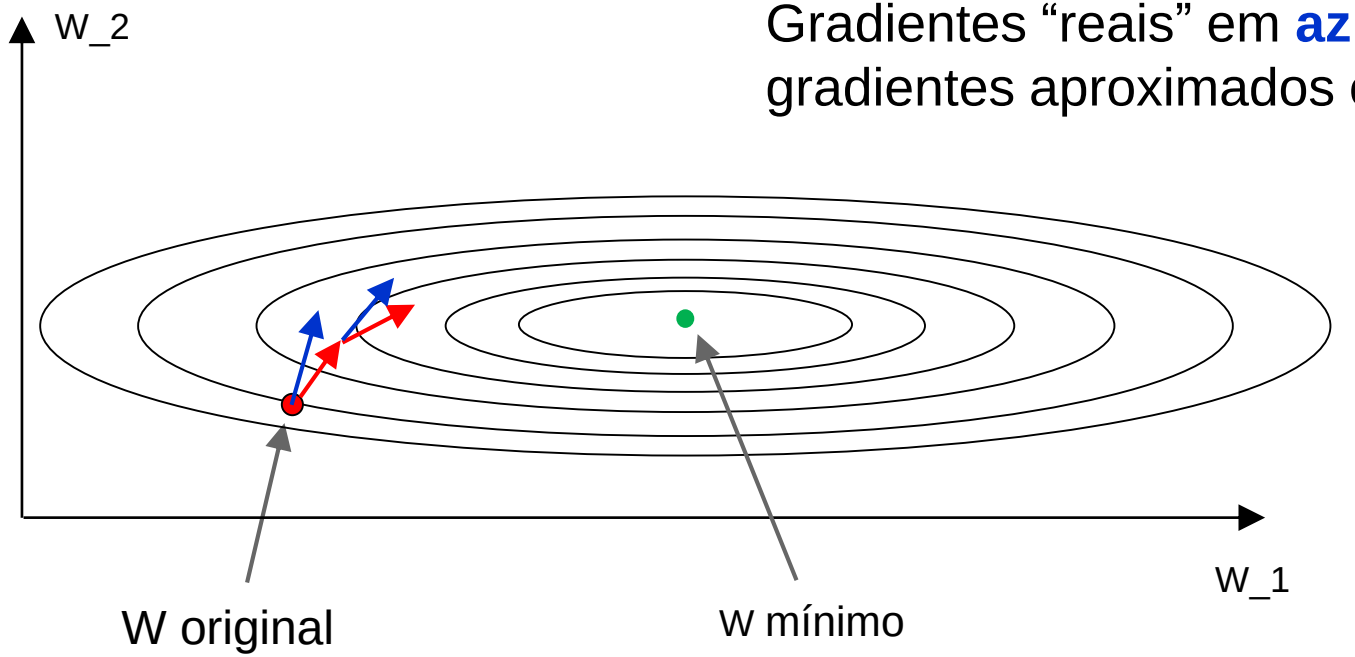
Gradiente Descendente Estocástico (*SGD*)

Gradientes apresentam “ruídos” mas ainda são bons para progredir na média em direção ao mínimo



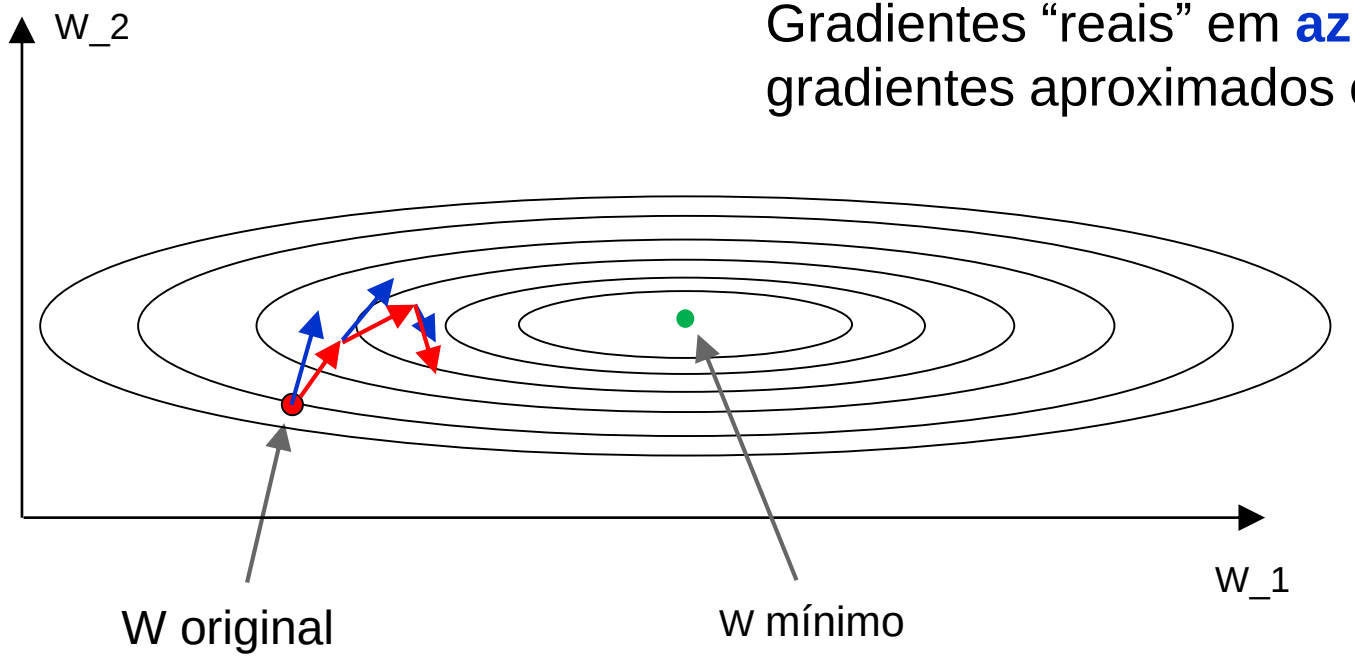
Gradiente Descendente Estocástico (*SGD*)

Gradientes apresentam “ruídos” mas ainda são bons para progredir na média em direção ao mínimo



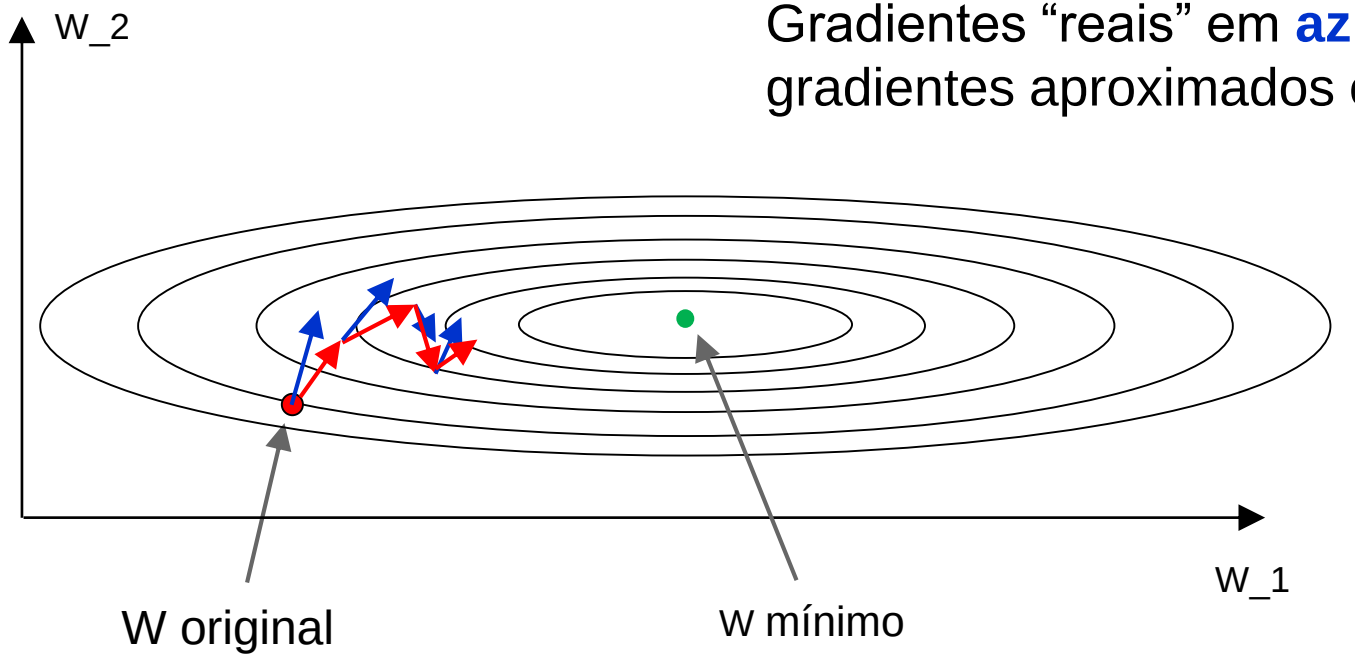
Gradiente Descendente Estocástico (*SGD*)

Gradientes apresentam “ruídos” mas ainda são bons para progredir na média em direção ao mínimo



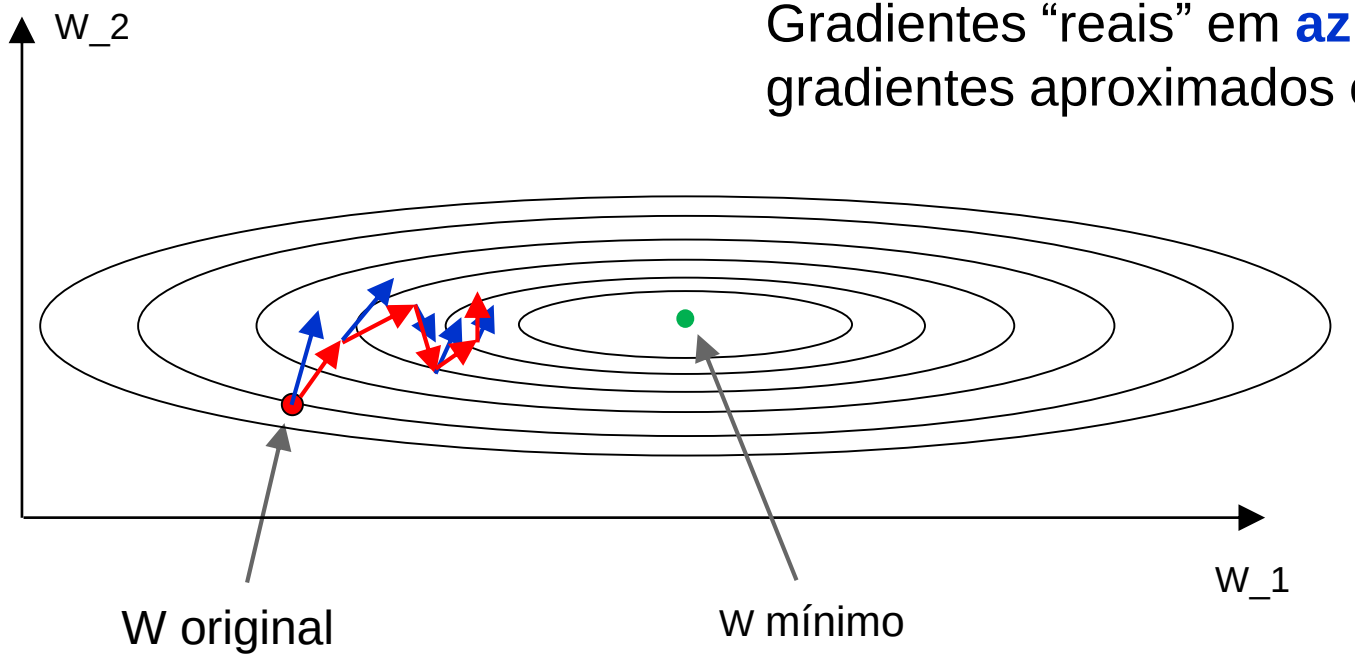
Gradiente Descendente Estocástico (*SGD*)

Gradientes apresentam “ruídos” mas ainda são bons para progredir na média em direção ao mínimo



Gradiente Descendente Estocástico (*SGD*)

Gradientes apresentam “ruídos” mas ainda são bons para progredir na média em direção ao mínimo



Gradiente Descendente Estocástico (*SGD*)

Ideia principal: usar apenas uma pequena amostra do conjunto de treinamento para calcular o gradiente em cada passo

Gradiente Descendente Estocástico (*SGD*)

Ideia principal: usar apenas uma pequena amostra do conjunto de treinamento para calcular o gradiente em cada passo

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Gradiente Descendente Estocástico (*SGD*)

Ideia principal: usar apenas uma pequena amostra do conjunto de treinamento para calcular o gradiente em cada passo

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Tamanhos comuns de ***minibatches*** são amostras de 32, 64, 128, 256, ...

Por exemplo, na AlexNet utilizou-se lotes com 256 amostras