Aprendizado Profundo 1

Descida de Gradiente

Professor: Lucas Silveira Kupssinskü

Agenda

- Otimização de funções vs Aprendizado
- Força bruta é uma opção?
- Por que a descida de gradiente funciona
- Problemas Convexos
- Problemas Não-Convexos
- Variações mais usadas
  - *SGD*
  - *Momentum*
  - *AdaGrad*
  - *RMSProp*
  - *Adam*

Otimização de Funções vs Aprendizado

- "*Aprender*" no contexto de redes neurais é um pouco diferente de otimização de funções no sentido mais clássico.
  - Otimização de funções
    - Específico para o problema
    - Tem garantias matemáticas de encontrar soluções
  - Aprendizado
    - "Genérico"
    - Pouca ou nenhuma garantia de encontrar soluções

Otimização de Funções vs Aprendizado

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim p_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

Queremos encontrar um extremo de uma função J parametrizada por $\boldsymbol{\theta}$ em relação a uma distribuição geradora dos nossos dados.

O que tem de errado/inviável nessa abordagem?

Otimização de Funções vs Aprendizado

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim p_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

Queremos encontrar um extremo de uma função J parametrizada por $\boldsymbol{\theta}$ em relação a uma distribuição geradora dos nossos dados.

O que tem de errado/inviável nessa abordagem?

- Não conhecemos p_{data}, temos acesso apenas a distribuição empírica $\hat{p}_{data}$ (nosso dataset)

Otimização de Funções vs Aprendizado

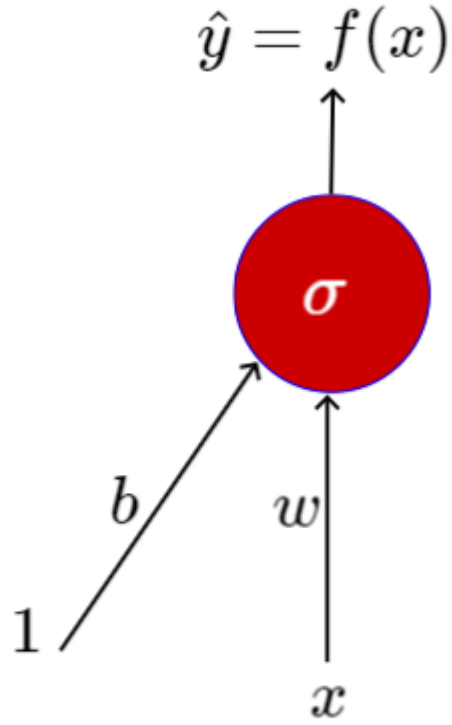
$$J(\boldsymbol{\theta}) = \mathbb{E}_{(x,y)\sim p_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

- $J(\theta)$ é conhecido como risco
- Caso p_{data} fosse conhecido, teríamos um problema de otimização clássico, com garantias formais
- Como não conhecemos p_{data}, estamos trabalhando com a minimização do risco *empírico*
  - Ou seja, um problema de aprendizado de máquina ☺

Força Bruta

- Possível?
  - Vamos tentar
  - $\underset{w,b}{\operatorname{argmin}}\, \mathcal{L}(w,b)$

$$\mathcal{L}(w,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$



$$f(x) = \frac{1}{1+e^{-(wx+b)}}$$

Força Bruta

- Possível?
  - Vamos tentar
  - $\underset{w,b}{\operatorname{argmin}} \mathcal{L}(w,b)$

Inviável!

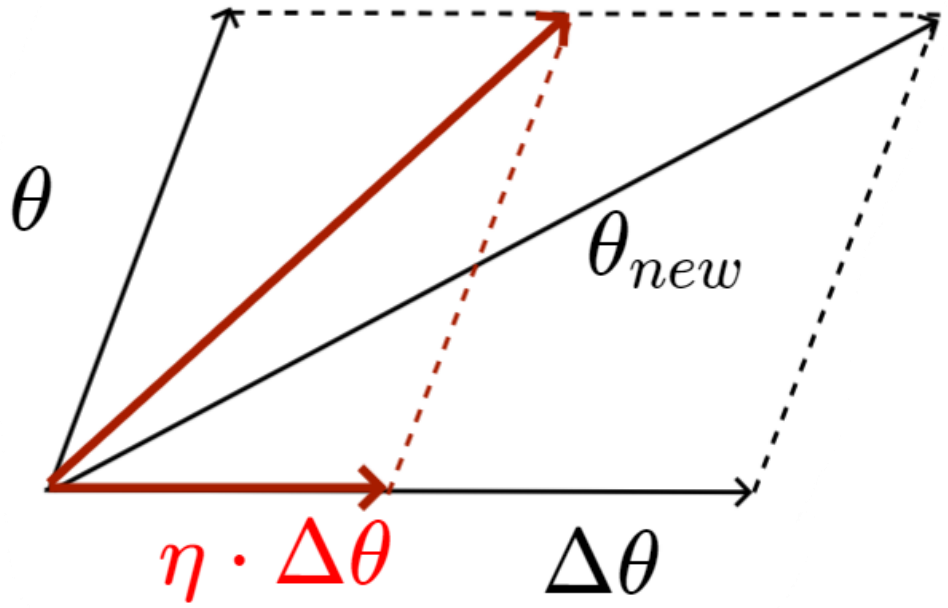
$$\mathcal{L}(w,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$

$$f(x) = \frac{1}{1+e^{-(wx+b)}}$$

Qual será a estratégia de aprendizado?

- Queremos aplicar uma atualização em θ $(\Delta\theta)$ que reduza $J(\theta)$

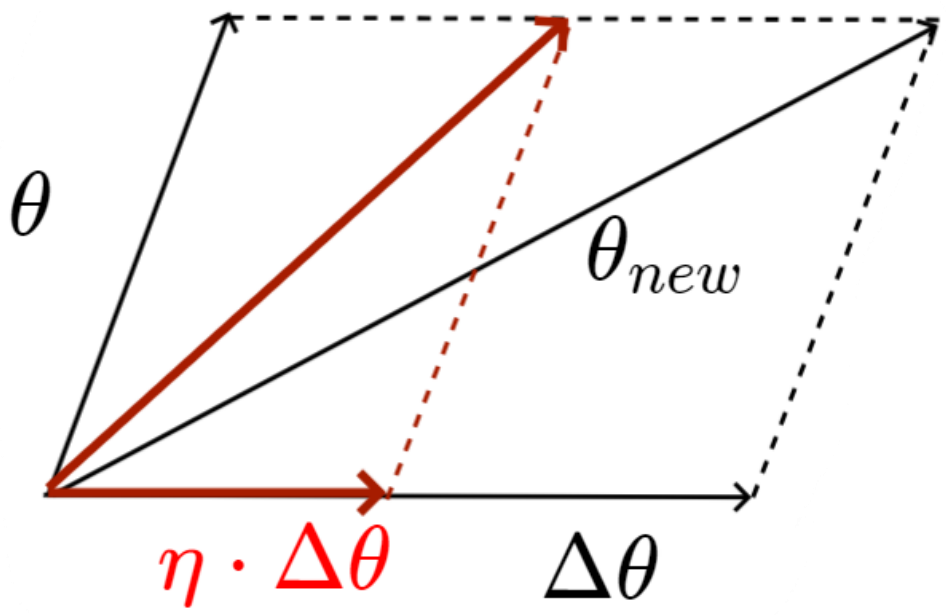
$$\theta + \eta\Delta\theta$$



Qual será a estratégia de aprendizado?

- Queremos aplicar uma atualização em θ $(\Delta\theta)$ que reduza $J(\theta)$

$$\theta + \eta\Delta\theta$$

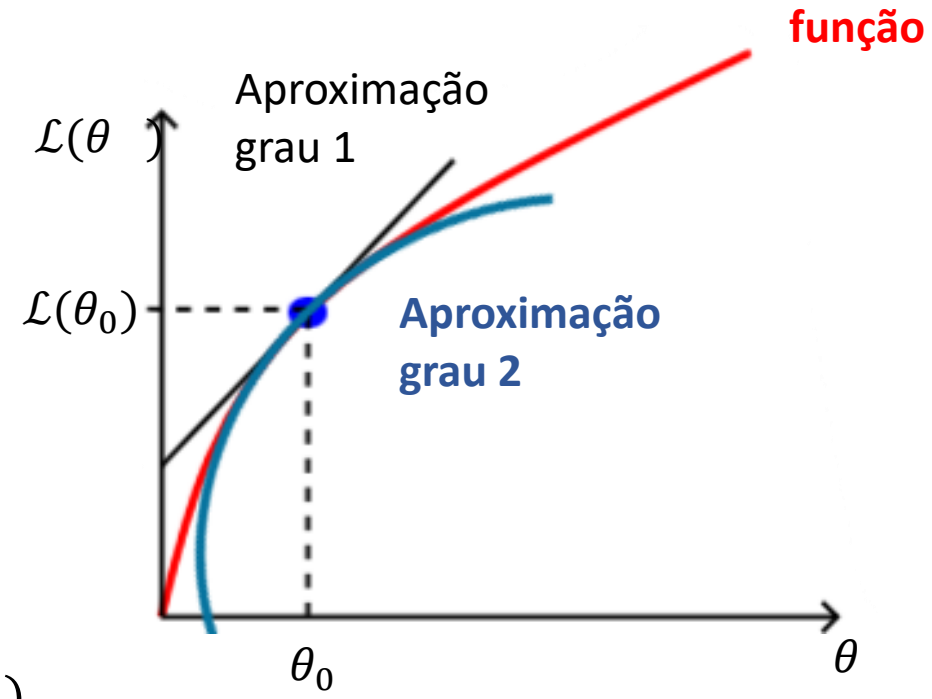


Qual variação de θ aplicar?

Vamos aplicar uma expansão em Série de Taylor ☺

Refresher

- Série de Taylor nos fornece uma forma de aproximar uma função continua diferenciável $\mathcal{L}(\theta)$ usando polinômios de grau n
  - + graus = melhor aproximação



$$\mathcal{L}(\theta) = \mathcal{L}(\theta_0) + \frac{\mathcal{L}`(\theta_0)}{1!}(\theta - \theta_0) + \frac{\mathcal{L}``(\theta_0)}{2!}(\theta - \theta_0)^2 + \frac{\mathcal{L}```(\theta_0)}{3!}(\theta - \theta_0)^3 + \cdots$$

Qual será a estratégia de aprendizado?

Vamos aplicar Taylor na Loss (considere $\Delta\theta = u$ para facilitar a notação)

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta) + \frac{\eta^2}{2!} * u^T \nabla_\theta^2 \mathcal{L}(\theta) + \frac{\eta^3}{3!} * u^T \nabla_\theta^3 \mathcal{L}(\theta) + \cdots$$

Qual será a estratégia de aprendizado?

Vamos aplicar Taylor na Loss (considere $\Delta\theta = u$ para facilitar a notação)

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta) + \frac{\eta^2}{2!} * u^T \nabla_\theta^2 \mathcal{L}(\theta) + \frac{\eta^3}{3!} * u^T \nabla_\theta^3 \mathcal{L}(\theta) + \cdots$$

Como o termo η tipicamente é pequeno, $\eta^2, \eta^3, \ldots \approx 0$

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta)$$

Qual será a estratégia de aprendizado?

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta)$$

Note que:

- Queremos que a mudança ηu nos parâmetros θ diminua a Loss

Logo

$$\mathcal{L}(\theta + \eta u) - \mathcal{L}(\theta) < 0$$

Qual será a estratégia de aprendizado?

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta)$$

Note que:

- Queremos que a mudança ηu nos parâmetros θ diminua a Loss

Logo

$$\mathcal{L}(\theta + \eta u) - \mathcal{L}(\theta) < 0$$

Qual será a estratégia de aprendizado?

$$\mathcal{L}(\theta + \eta u) = \mathcal{L}(\theta) + \eta * u^T \nabla_\theta \mathcal{L}(\theta)$$

Note que:

- Queremos que a mudança ηu nos parâmetros θ diminua a Loss

Logo

$$u^T \nabla_\theta \mathcal{L}(\theta) < 0$$

Produto Escalar

Qual será a estratégia de aprendizado?

$$u^T \nabla_\theta \mathcal{L}(\theta) < 0$$ Produto Escalar

Quando um produto escalar é menor que zero?

Qual será a estratégia de aprendizado?

$$u^T \nabla_\theta \mathcal{L}(\theta) < 0$$ Produto Escalar

Seja β o ângulo entre u^T e $\nabla_\theta \mathcal{L}(\theta)$

Sabemos que:

- $u^T \cdot \nabla_\theta \mathcal{L}(\theta) = \|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta)$

Qual será a estratégia de aprendizado?

$$u^T \nabla_\theta \mathcal{L}(\theta) < 0$$ Produto Escalar

Seja β o ângulo entre u^T e $\nabla_\theta \mathcal{L}(\theta)$

Sabemos que:

- $u^T \cdot \nabla_\theta \mathcal{L}(\theta) = \|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta)$

$$\|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta) < 0$$

Qual será a estratégia de aprendizado?

$$u^T \nabla_\theta \mathcal{L}(\theta) < 0$$ Produto Escalar

Seja β o ângulo entre u^T e $\nabla_\theta \mathcal{L}(\theta)$

Sabemos que:

- $u^T \cdot \nabla_\theta \mathcal{L}(\theta) = \|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta)$

$\|u^T\|$ → Sempre positivo

$*$

$\|\nabla_\theta \mathcal{L}(\theta)\|$ → Sempre positivo

$*$

$\cos(\beta) < 0$ → Negativo para $\beta \in (90º, 180º)$

Qual será a estratégia de aprendizado?

$$\|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta) < 0$$

""""Mais negativo para $\beta = 180^{o}$""""

Descida de Gradiente

Essa é a descida de gradiente!

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathrm{J}$$

$$\|u^T\| * \|\nabla_\theta \mathcal{L}(\theta)\| * \cos(\beta) < 0$$

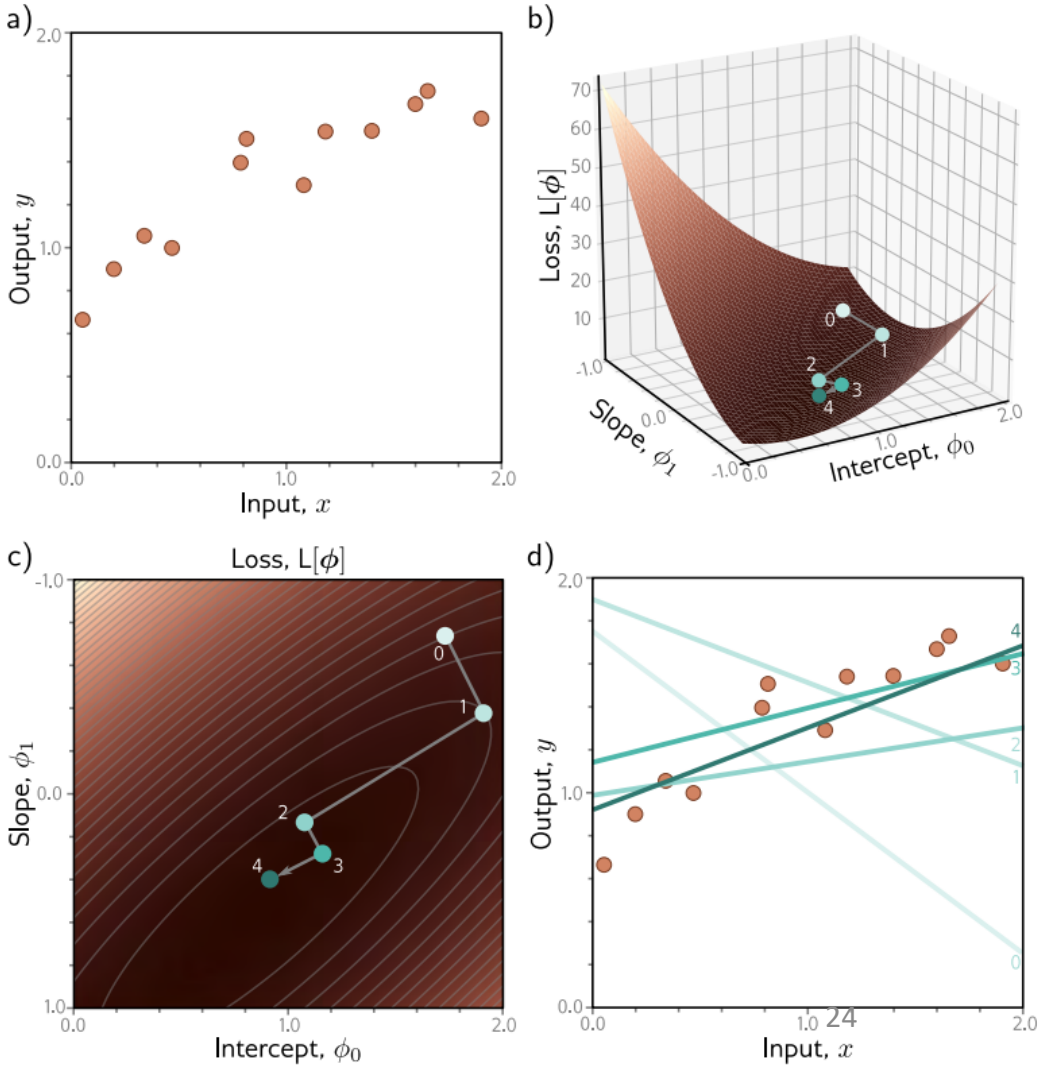
""""Mais negativo para $\beta = 180º$""""

Intuição

Situação 1 – Função Convexa

$$\hat{y} = \theta_0 + \theta_1 x$$
$$J(\theta) = (y - \hat{y})^T (y - \hat{y})$$

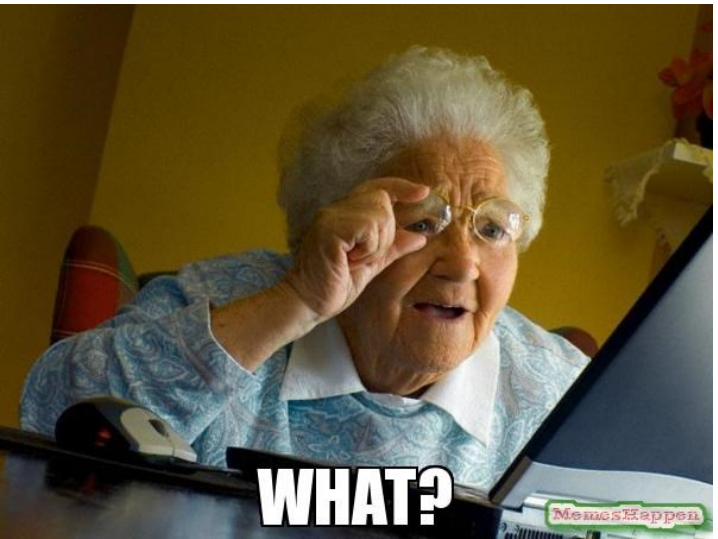
- Qualquer inicialização (eventualmente) vai nos levar ao mínimo global caso a taxa de aprendizado seja suficientemente pequena!



Situação 2 – Função Não Convexa

$$\hat{y} = \sin(\theta_0 + 0.06 * \theta_1 * x) * \exp\left(-\frac{(\theta_0 + 0.06 * \theta_1 * x)^2}{32}\right)$$

$$J(\theta) = (y - \hat{y})^T (y - \hat{y})$$

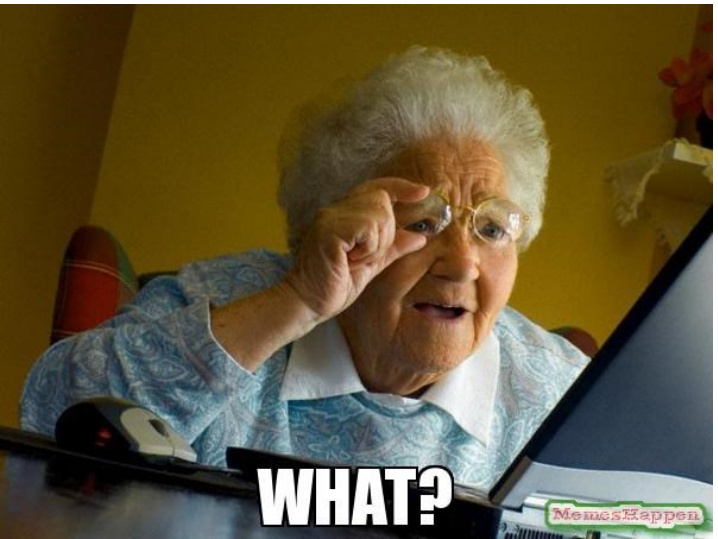


Situação 2 – Função Não Convexa

$$\hat{y} = \sin(\theta_0 + 0.06 * \theta_1 * x) * \exp\left(-\frac{(\theta_0 + 0.06 * \theta_1 * x)^2}{32}\right)$$

$$J(\theta) = (y - \hat{y})^T (y - \hat{y})$$

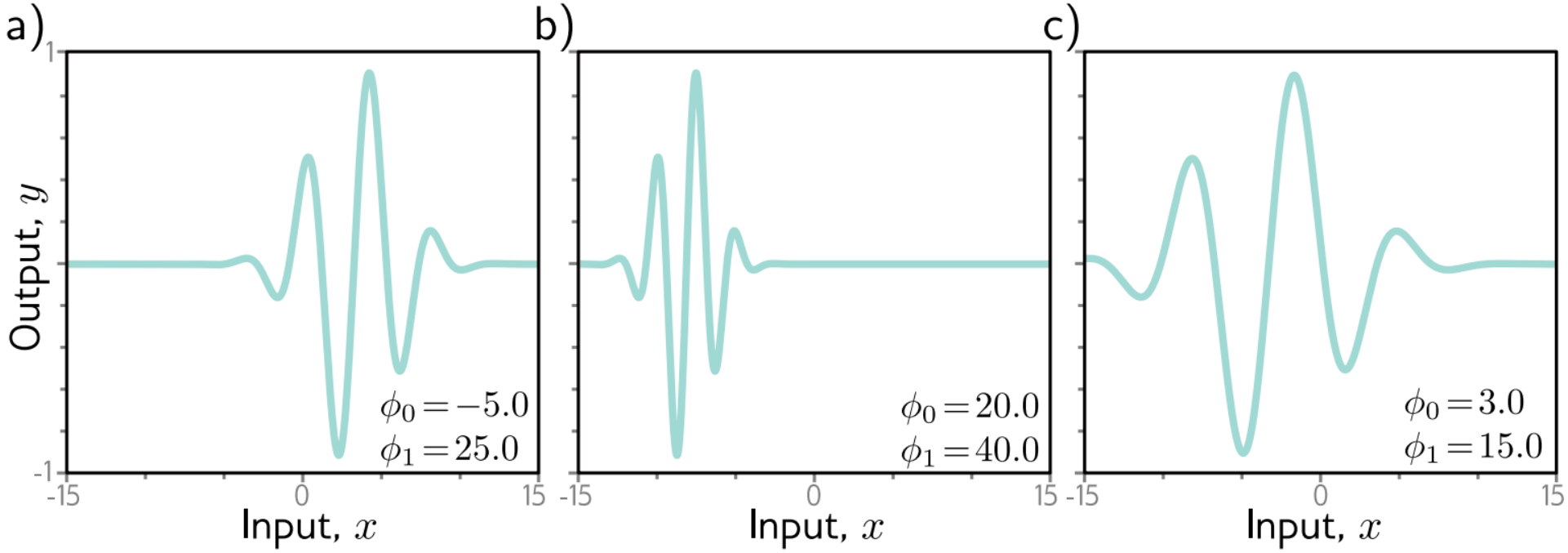
Você consegue esboçar essa função $\hat{y}$?



Situação 2 – Função Não Convexa

$$\hat{y} = \sin(\theta_0 + 0.06 * \theta_1 * x) * \exp\left(-\frac{(\theta_0 + 0.06 * \theta_1 * x)^2}{32}\right)$$

$$J(\theta) = (y - \hat{y})^T (y - \hat{y})$$

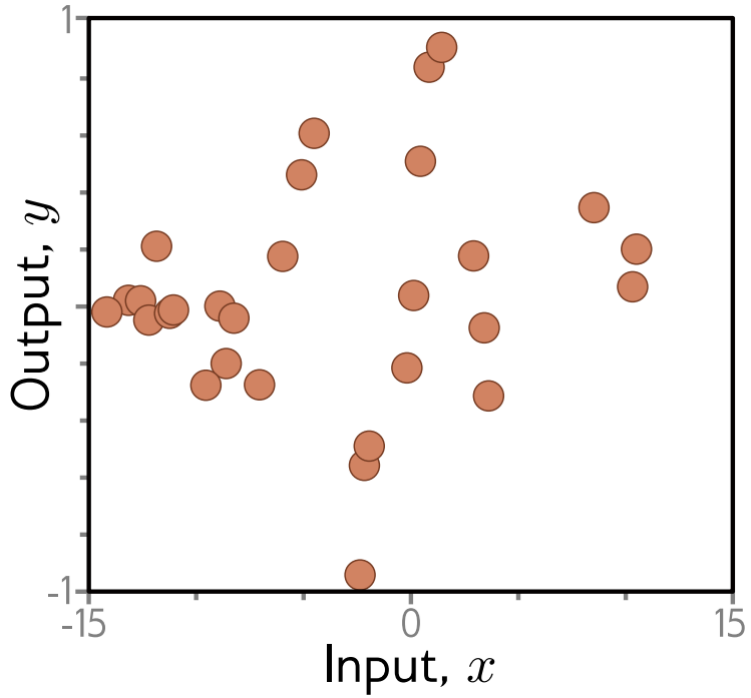


Modelo de Gabor: Modelo não-linear que mapeia o escalar x para um escalar y e tem dois parâmetros. θ_0 determina a posição do centro da função e θ_1 torna a função mais "espremida" no eixo x

Situação 2 – Função Não Convexa

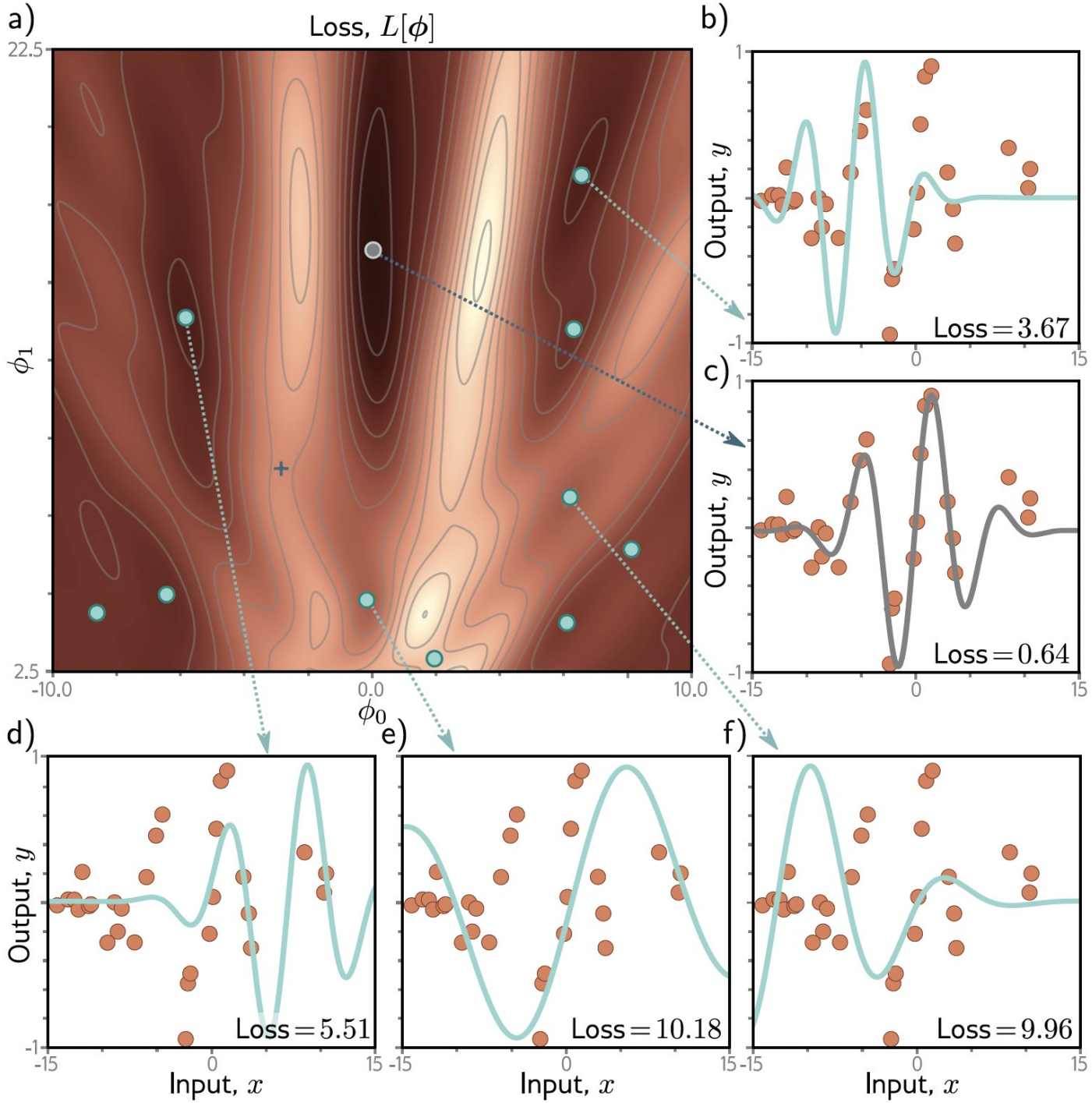
$$\hat{y} = \sin(\theta_0 + 0.06 * \theta_1 * x) * \exp\left(-\frac{(\theta_0 + 0.06 * \theta_1 * x)^2}{32}\right)$$

$$J(\theta) = (y - \hat{y})^T (y - \hat{y})$$



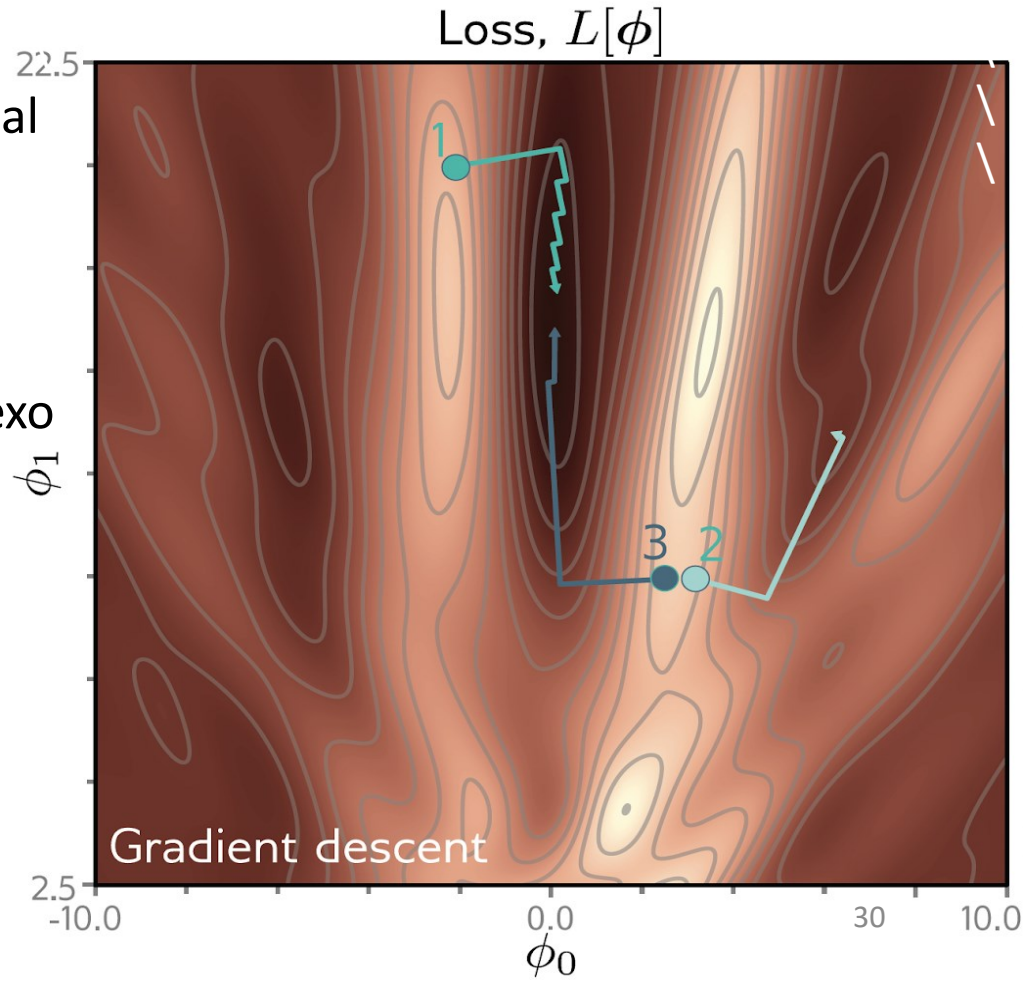
Dados de Treinamento

a) Loss Landscape
b) Mínimo Local
c) Mínimo Global
d) Mínimo Local
e) Mínimo Local
f) Mínimo Local



Situação 2 – Função Não Convexa

- Descida de Gradiente funciona para esses casos?
  - Inicializando em 1 ou 3 chegamos no mínimo global
  - Inicializando em 2 ficamos presos em um mínimo local
- Precisamos sair de um “bom vale”
  - Ou seja, precisamos estudar como inicializar nossos parâmetros caso o Loss Landscape não seja convexo
  - *Spoiler*: em redes neurais profundas “*nunca*” é convexo ☺



Variações da Descida de Gradiente

- Podemos variar o número de instâncias que são usadas para estimar o gradiente
  - N – *Batch Gradient Descent*
  - $[N-1, 2]$ – *Mini Batch Gradient Descent*
  - 1 – *Stochastic Gradient Descent*

Variações da Descida de Gradiente

- Podemos variar o número de instâncias que são usadas para estimar o gradiente
  - N – *Batch Gradient Descent*
  - $[N - 1, 2]$ – *Mini B*
  - 1 – *Stochastic Grad*

Por que não usar sempre todas as instâncias?

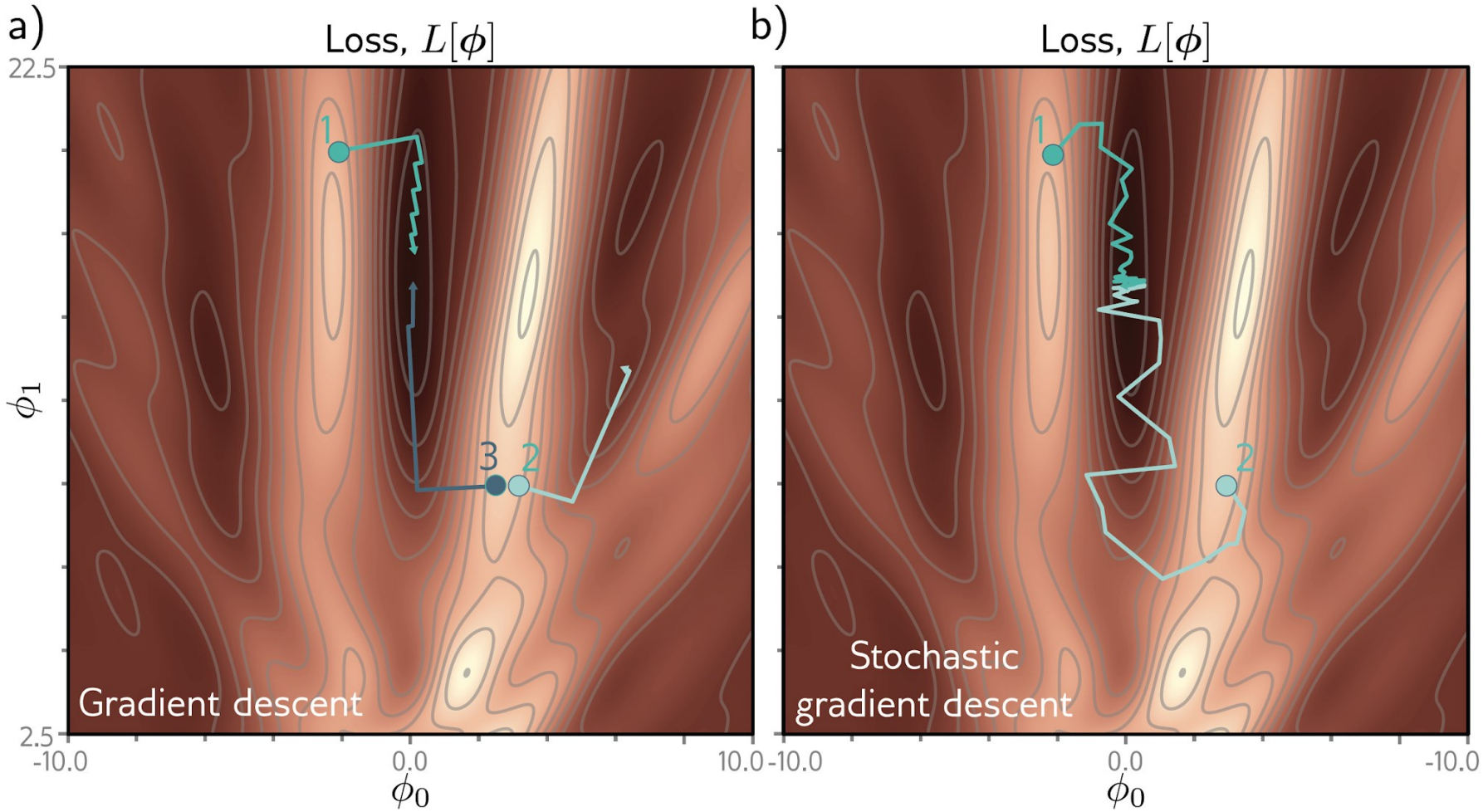
Variações da Descida de Gradiente

- Podemos variar o número de instâncias que são usadas para estimar o gradiente
  - N – *Batch Gradient Descent*
  - $[N - 1, 2]$ – *Mini B*
  - 1 – *Stochastic Grad*

Por que não usar sempre todas as instâncias?

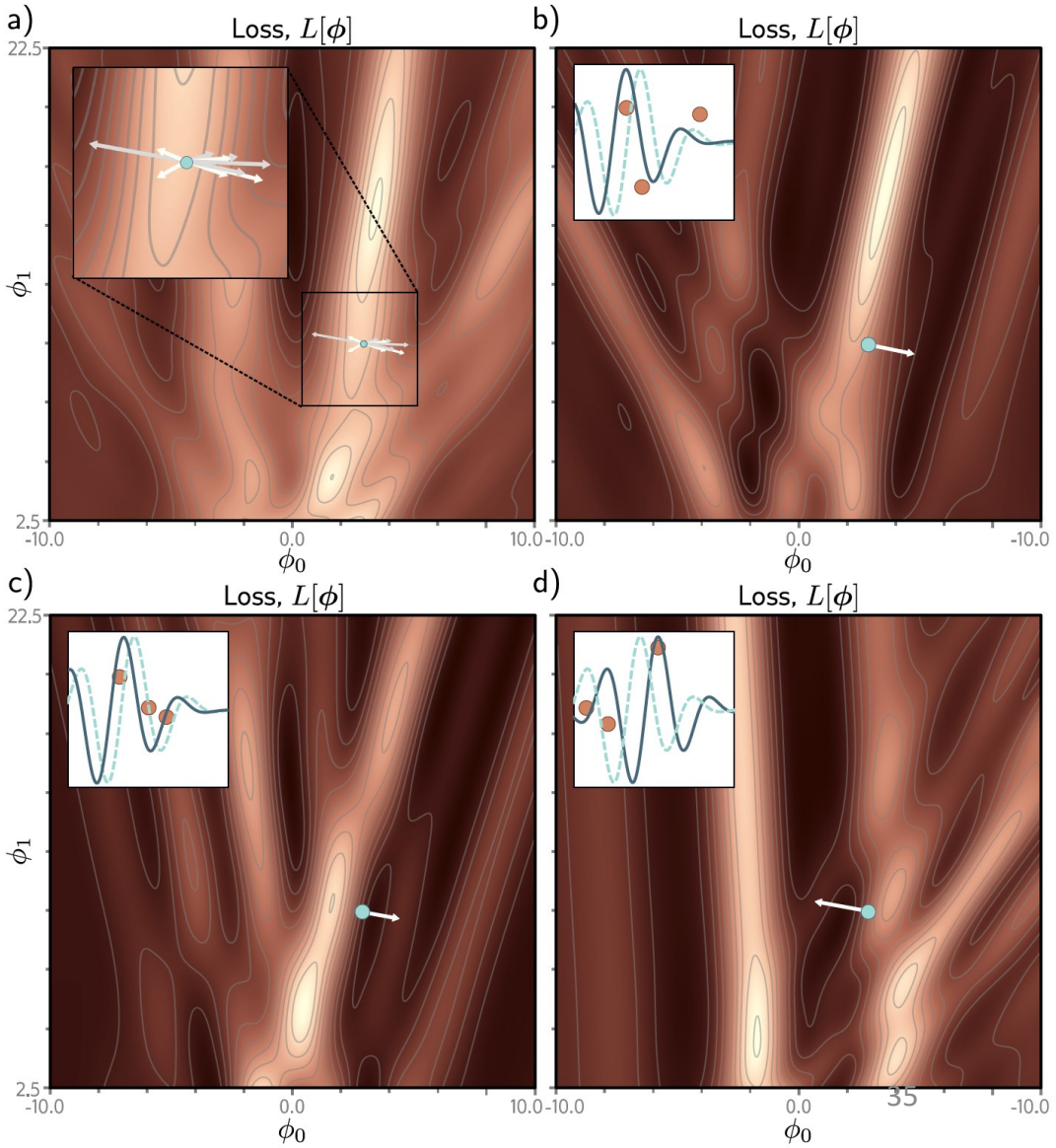
1 – Usar apenas uma instância pode introduzir ruído no treinamento, auxiliando o processo de otimização a sair de mínimos locais
2 – Limitações de *Hardware*

Stochastic Gradient Descent



Stochastic Gradient Descent

- a) Loss calculado para todo o dataset e os gradientes dos possíveis mini-batches exibidos na figura
- b, c, d) exemplos de minibatchs com tamanho 3



Stochastic Gradient Descent

- Instâncias são amostradas sem reposição
- Uma época ocorre quando todas as instâncias foram vistas
- Adiciona ruído no treinamento
  - Batch Gradient Descent é determinístico, o resultado depende unicamente da inicialização (e de η)
- É mais leve computacionalmente
- Pode escapar de mínimos locais e pontos de sela
- Oferece resultados que generalizam mais (observação empírica)
- Não converge (Por quê?)

Momentum

- Uma das modificações mais comuns da descida de gradiente envolve considerar o gradiente dos passos anteriores
- A ideia é aumentar o tamanho dos passos quando a direção do gradiente não mudar muito

$$\theta_{(t+1)} = \theta_{(t)} - \eta \mathcal{V}_{(t)}$$

$$\mathcal{V}_{(t)} = \beta \mathcal{V}_{(t-1)} + (1 - \beta) \nabla_{\theta_{(t)}} \mathcal{L}(\theta)$$

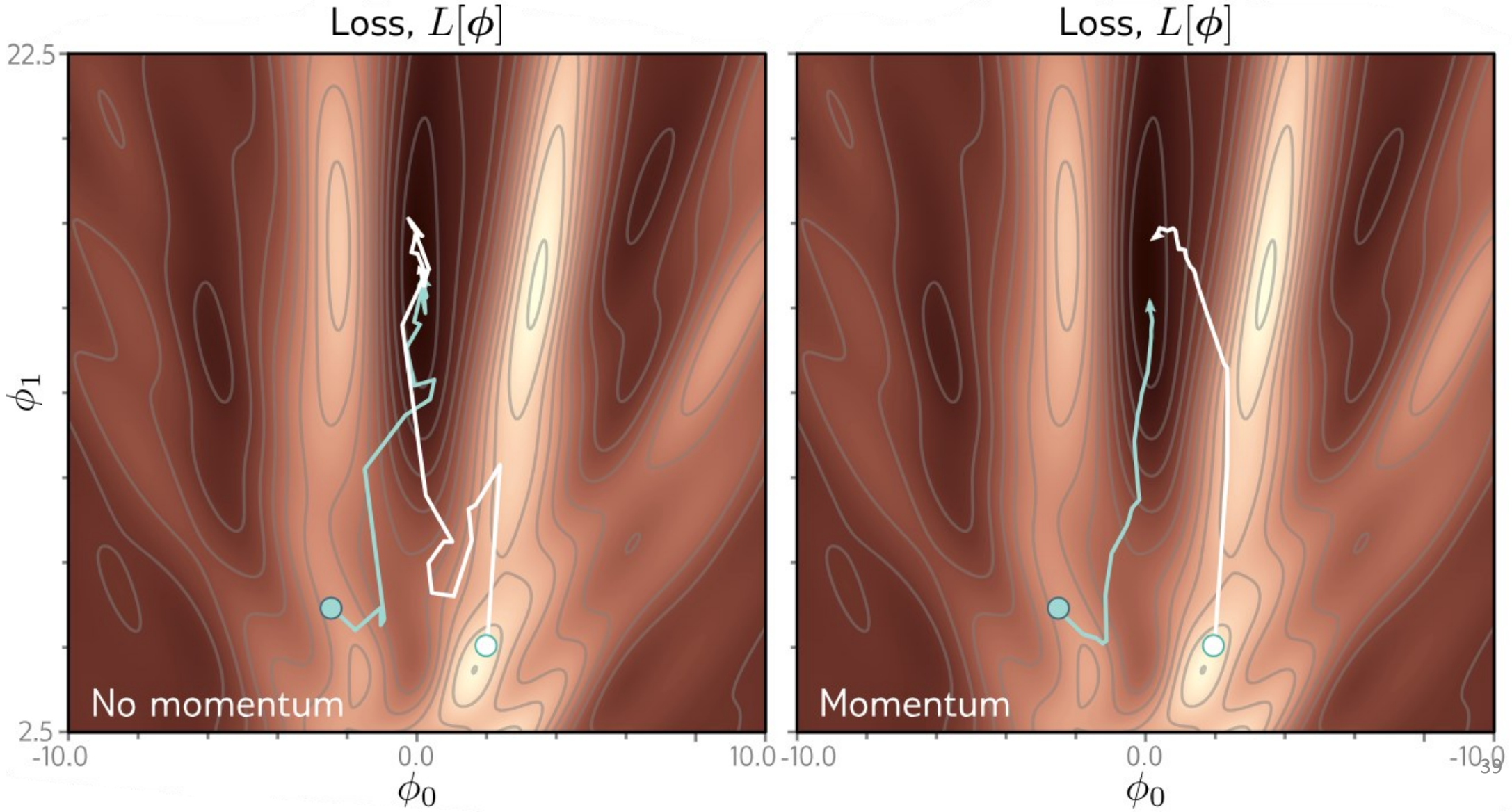
Momentum

$$\theta_{(t+1)} = \theta_{(t)} - \eta \mathcal{V}_{(t+1)}$$

$$\mathcal{V}_{(t+1)} = \beta \mathcal{V}_{(t)} + (1 - \beta) \nabla_{\theta_{(t)}} \mathcal{L}(\theta)$$

- $\beta \in [0,1)$: controla quanto de *momentum* será considerado
- Repare que a formalização é recursiva, ou seja, estamos considerando uma soma ponderada de todos os gradientes anteriores
- A "taxa de aprendizado efetiva" aumenta se os últimos gradientes apontaram para a mesma direção e diminui caso contrário
- Tem efeito de suavizar a trajetória e reduzir oscilações no gradiente

Momentum


Loss, $L[\phi]$
22.5
2.5
ϕ_1
No momentum
-10.0
0.0
10.0
ϕ_0
Loss, $L[\phi]$
Momentum
-10.0
0.0
ϕ_0

Nesterov Accelerated Momentum

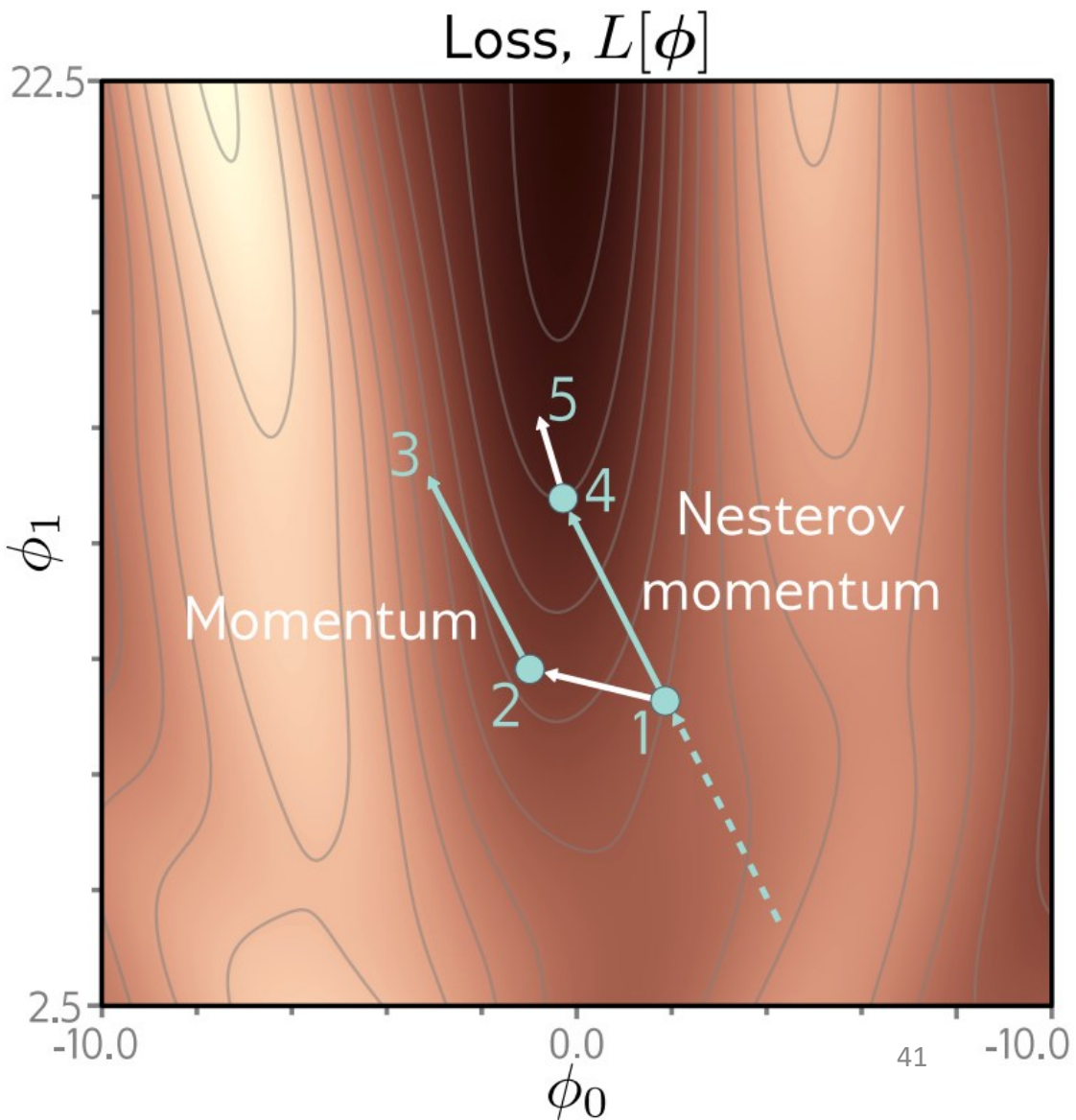
- *Momentum* pode ser considerado um preditor do ponto onde o SGD se moveria na próxima atualização
- Para tentar acelerar as atualizações, podemos computar o gradiente no ponto para onde o *momentum* aponta
- Perceba na formalização abaixo que o gradiente é avaliado em $\theta - \boldsymbol{\eta \mathcal{V}_{(t)}}$

$$\theta_{(t+1)} = \theta_{(t)} - \eta \mathcal{V}_{(t+1)}$$

$$\mathcal{V}_{(t+1)} = \beta \mathcal{V}_{(t)} + (1 - \beta) \nabla_{\theta_{(t)}} \mathcal{L}(\theta - \boldsymbol{\eta \mathcal{V}_{(t)}})$$

Nesterov Accelerated Momentum

- O movimento anterior é exibido na figura em tracejado e levou o θ até 1
- *Momentum* considera o gradiente no ponto 1 para fazer a atualização (indicado em branco)
- *Nesterov Accelerated Momentum* considera o gradiente no ponto 4 (primeiro aplica a atualização do *Momentum)*



Normalização de Gradiente

- Descida de Gradiente com passo de tamanho fixo pode apresentar problemas
  - Atualizações grandes quando o gradiente naquela direção é grande
    - Talvez devêssemos ir com mais calma
  - Atualizações pequenas quando o gradiente naquela direção é pequeno
    - Talvez precisássemos explorar mais
- Quando o gradiente é muito mais íngreme em um direção é difícil definir uma taxa de aprendizado adequada
- Solução: Normalizar o gradiente

Normalização de Gradiente

- A solução direta, toda atualização vai mover um passo de tamanho fixo (determinado por η)
- Para fazer isso precisamos de dois termos:

$$m_{(t+1)} = \nabla_\theta \mathcal{L}(\theta)$$
$$v_{(t+1)} = \left(\nabla_\theta \mathcal{L}(\theta)\right)^2$$

A regra de atualização muda para:

$$\theta_{(t+1)} = \theta_{(t)} - \eta \frac{m_{(t+1)}}{\sqrt{v_{(t+1)}} + \epsilon}$$

Normalização de Gradiente

$$\theta_{(t+1)} = \theta_{(t)} - \eta \frac{m_{(t+1)}}{\sqrt{v_{(t+1)}} + \epsilon}$$

- Repare que com a normalização, o termo que multiplica a taxa de aprendizado será composto apenas por um vetor de 1s e -1s
- O tamanho do passo é totalmente determinado pela taxa de aprendizado

AdaGrad

- *Adaptative (per-parameter) Gradient*
- Ideia: ter uma taxa de aprendizado diferente para cada parâmetro

- $G_{t+1} = G_t + (\nabla_\theta L)^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1}+\epsilon}} \nabla_\theta L$

AdaGrad

- *Adaptative (per-parameter) Gradient*
- Ideia: ter uma taxa de aprendizado diferente para cada parâmetro

- $G_{t+1} = G_t + (\nabla_\theta L)^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_{t+1}+\epsilon}} \nabla_\theta L$

- O que acontece com a taxa de aprendizado depois de algum tempo?

RMSProp

- *Root Mean Square Propagation*
- Algoritmo não publicado mas disponível em praticamente todos os frameworks
  - Apresentado em um slide de aula do Geoff Hinton
- Ideia: Normalizar o gradiente pela raiz quadrada de uma média móvel do gradiente ao quadrado
- Adiciona o parâmetro ρ para decaimento, evitando o problema do AdaGrad

- $\mathbb{E}[g^2]_{t+1} = \rho\mathbb{E}[g^2]_t + (1-\rho)(\nabla_\theta L)^2$
- $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\mathbb{E}[g^2]_{t+1}+\epsilon}}\nabla_\theta L$

Adam

- *Adaptative Momentum Estimation*
- Junta as ideias de Normalização de Gradiente e de *Momentum* (aplicado a estimativa do gradiente e ao gradiente elevado ao quadrado)

Adam

$$m_{(t+1)} = \beta m_{(t)} + (1-\beta)\nabla_\theta \mathcal{L}(\theta)$$
$$v_{(t+1)} = \gamma v_{(t)} + (1-\gamma)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2$$

$\beta \in [0,1), \gamma \in [0,1)$

Adam

$$m_{(t+1)} = \beta m_{(t)} + (1-\beta)\nabla_\theta \mathcal{L}(\theta)$$
$$v_{(t+1)} = \gamma v_{(t)} + (1-\gamma)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2$$

$\beta \in [0,1), \gamma \in [0,1)$

Como esses termos começam zerados, podemos fazer uma correção em m_{t+1} e v_{t+1} para aumentar um pouco a magnitude das primeiras atualizações

Adam

$$m_{(t+1)} = \beta m_{(t)} + (1-\beta)\nabla_\theta \mathcal{L}(\theta)$$
$$v_{(t+1)} = \gamma v_{(t)} + (1-\gamma)\left(\nabla_\theta \mathcal{L}(\theta)\right)^2$$

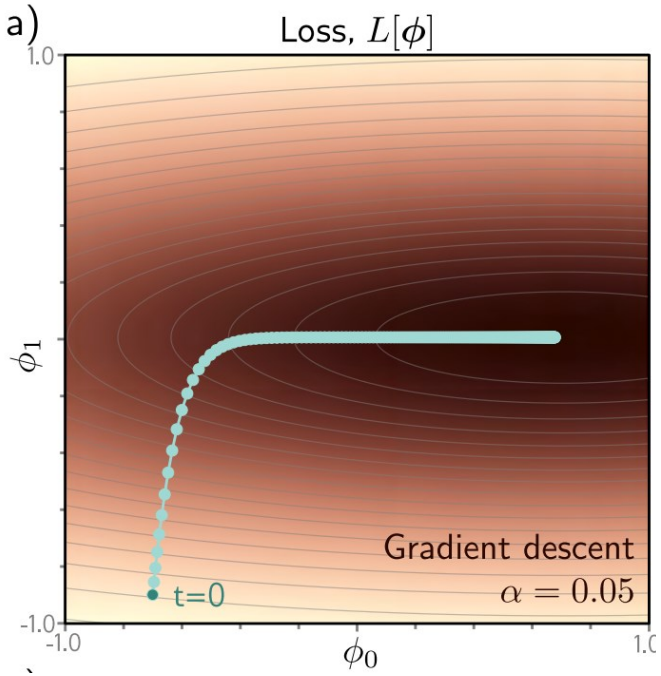
- $\tilde{m}_{(t+1)} = \frac{m_{(t+1)}}{1-\beta^{t+1}}$
- $\tilde{v}_{(t+1)} = \frac{v_{(t+1)}}{1-\gamma^{t+1}}$

$$\theta_{t+1} = \theta_t - \eta \frac{\tilde{m}_{(t+1)}}{\sqrt{\tilde{v}_{(t+1)}} + \epsilon}$$

$\beta \in [0,1), \gamma \in [0,1)$

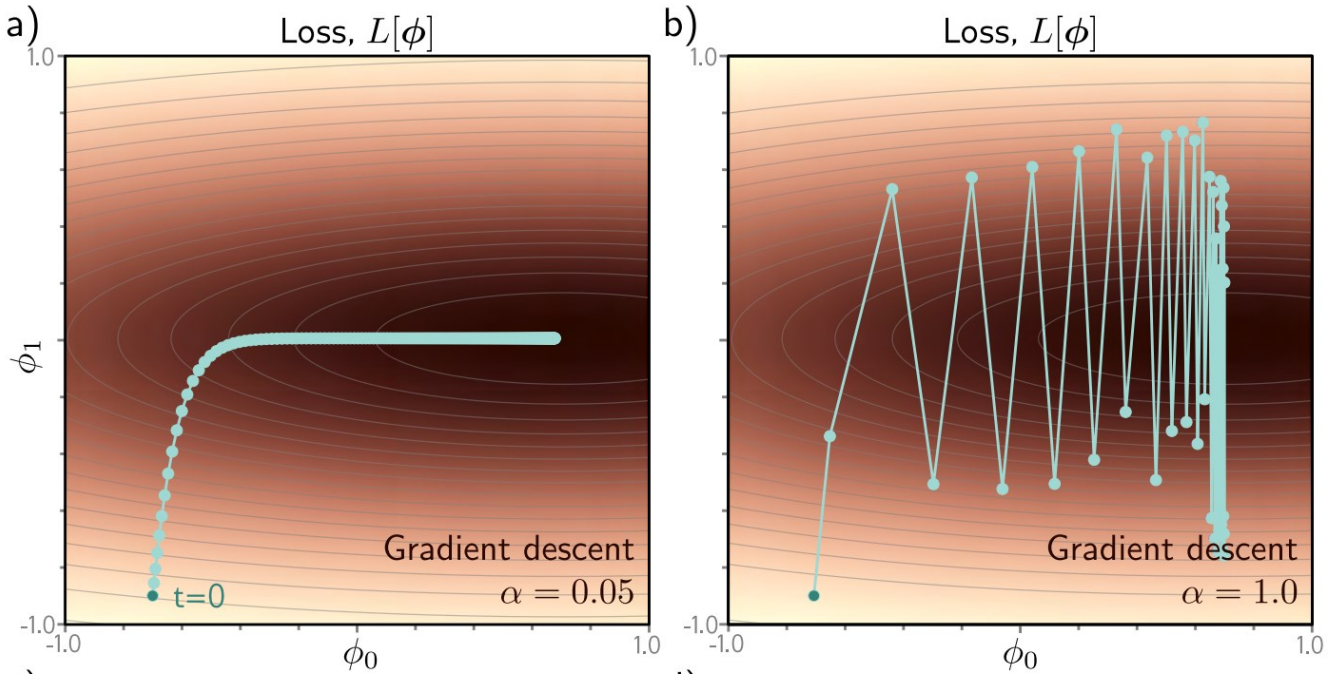
Adam

- Repare que essa Loss landscape muda rapidamente na vertical e muda mais devagar na horizontal
- a) Usando uma taxa de aprendizado que faz bom progresso na vertical, o caminho na horizontal é muito lento



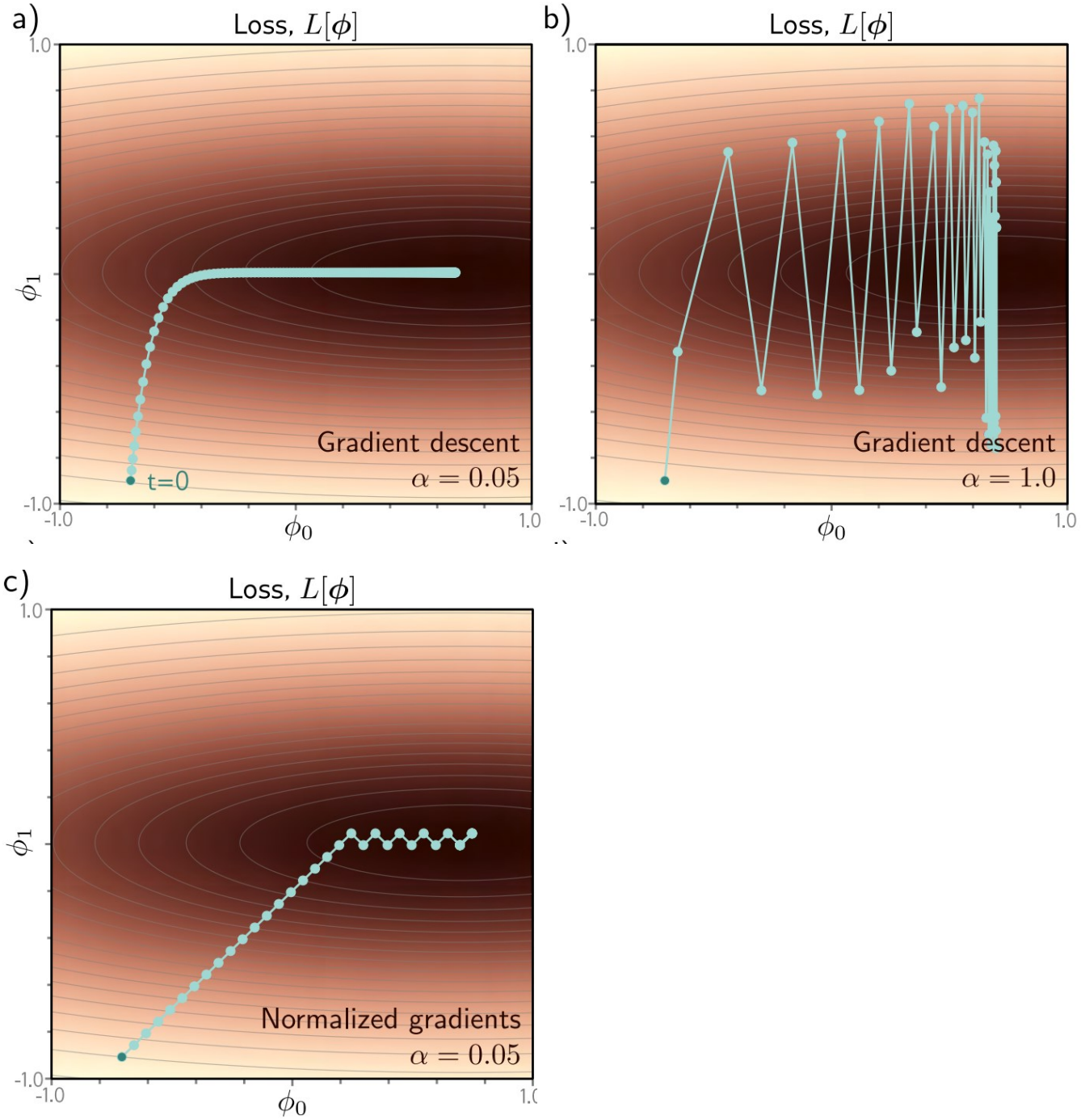
Adam

- Repare que essa Loss landscape muda rapidamente na vertical e muda mais devagar na horizontal
- a) Usando uma taxa de aprendizado que faz bom progresso na vertical, o caminho na horizontal é muito lento
- b) Usando uma taxa de aprendizado que faz bom progresso na horizontal, na vertical os passos são exagerados



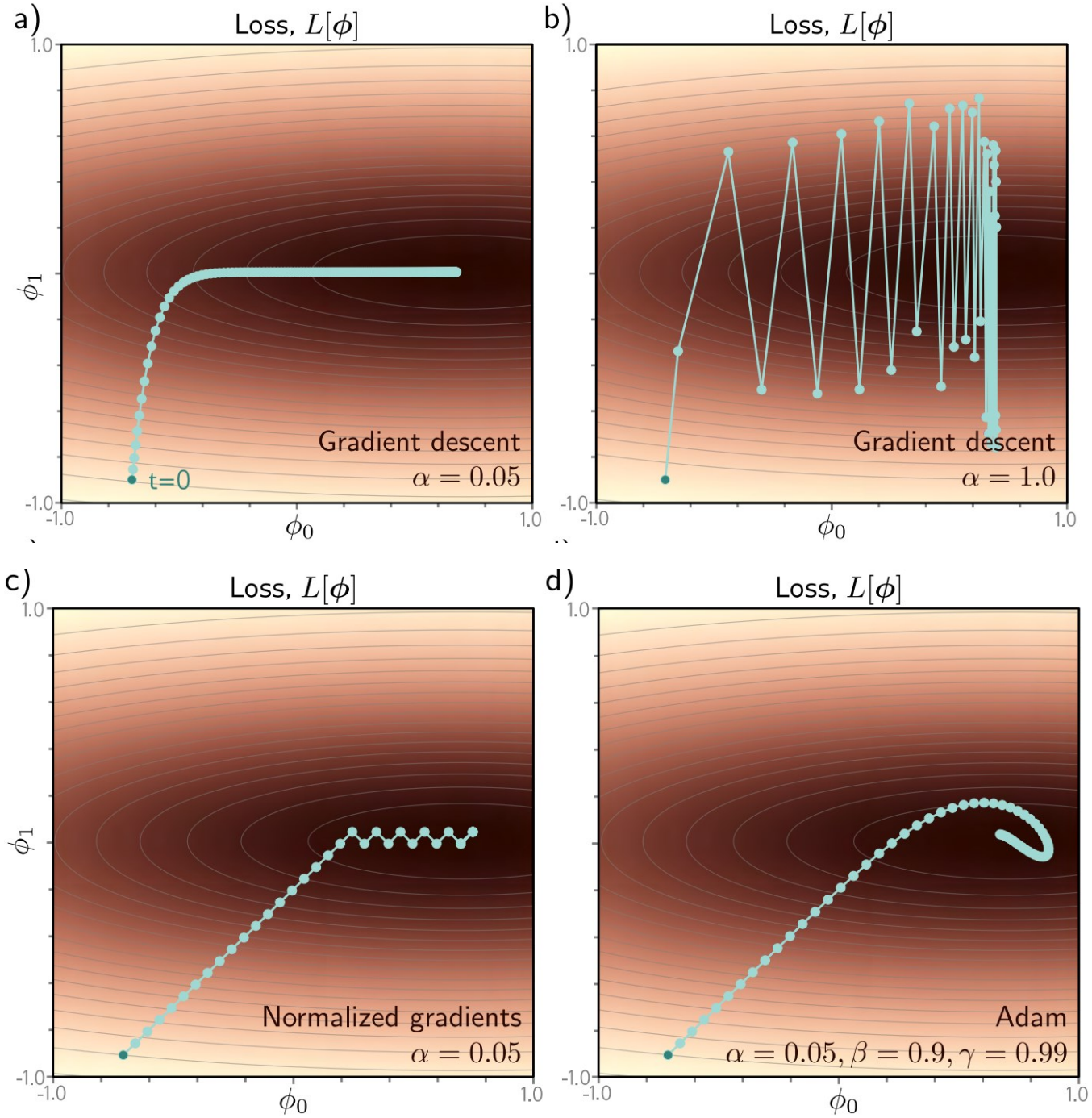
Adam

- Repare que essa Loss landscape muda rapidamente na vertical e muda mais devagar na horizontal
- a) Usando uma taxa de aprendizado que faz bom progresso na vertical, o caminho na horizontal é muito lento
- b) Usando uma taxa de aprendizado que faz bom progresso na horizontal, na vertical os passos são exagerados
- c) Normalizar os gradientes e dar passos fixos em cada uma das direções resolve o problema, mas faz com que o algoritmo tenha mais dificuldade de encontrar o mínimo global



Adam

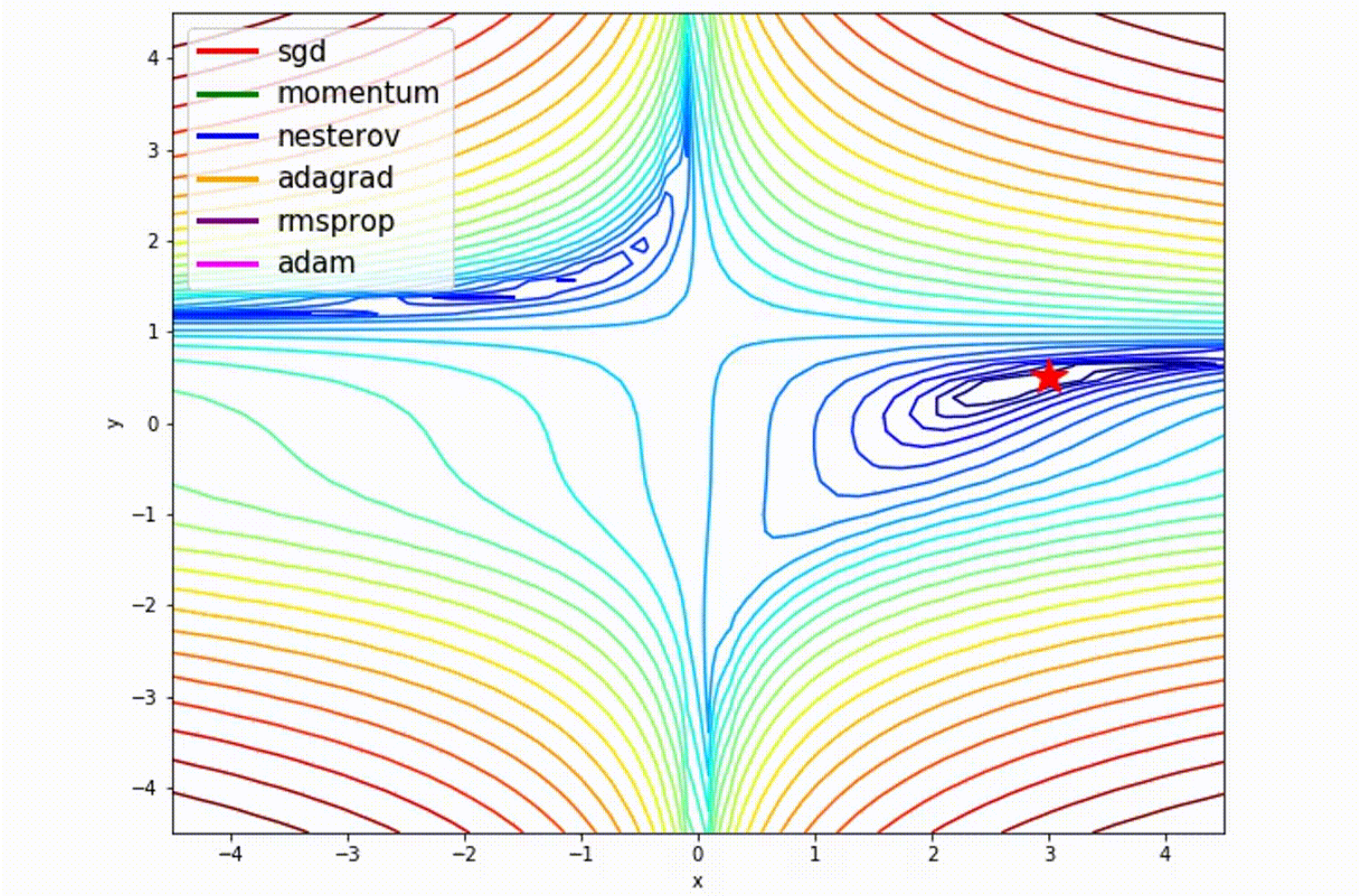
- Repare que essa Loss landscape muda rapidamente na vertical e muda mais devagar na horizontal
- a) Usando uma taxa de aprendizado que faz bom progresso na vertical, o caminho na horizontal é muito lento
- b) Usando uma taxa de aprendizado que faz bom progresso na horizontal, na vertical os passos são exagerados
- c) Normalizar os gradientes e dar passos fixos em cada uma das direções resolve o problema, mas faz com que o algoritmo tenha mais dificuldade de encontrar o mínimo global
- d) Usando *momentum* no gradiente estimado e no termo de normalização temos um caminho mais suave até o mínimo



Adam

- As magnitudes (tamanho) dos gradientes tendem a ser diferentes conforme as funções de ativação, camada da rede,...
- *Adam* tende a ser mais imune a essas diferenças e também menos sensível a escolha da taxa de aprendizado inicial

Comparação de otimizadores



Referências:

- Sugere-se ***fortemente*** a leitura de:
  - Capítulos 4 e 8 de GOODFELLOW, I., BENGIO, Y., COURVILLE, A. Deep Learning. MIT Press, 775p., 2016. (https://www.deeplearningbook.org/)
    - https://www.deeplearningbook.org/contents/numerical.html
    - https://www.deeplearningbook.org/contents/optimization.html
  - Capítulo 6 de Understanding Deep Learning
    - https://udlbook.github.io/udlbook/