MINERAÇÃO DE DADOS COMPLEXOS

Curso de aperfeiçoamento

UNICAMP

INF-0618
Tópicos em Aprendizado de Máquina II

Aula 3 – Treinamento
Hiperparâmetros e Otimização

Profa. Fernanda Andaló

2018

Instituto de Computação - Unicamp

Roteiro

Underfit e Overfit

Regularização

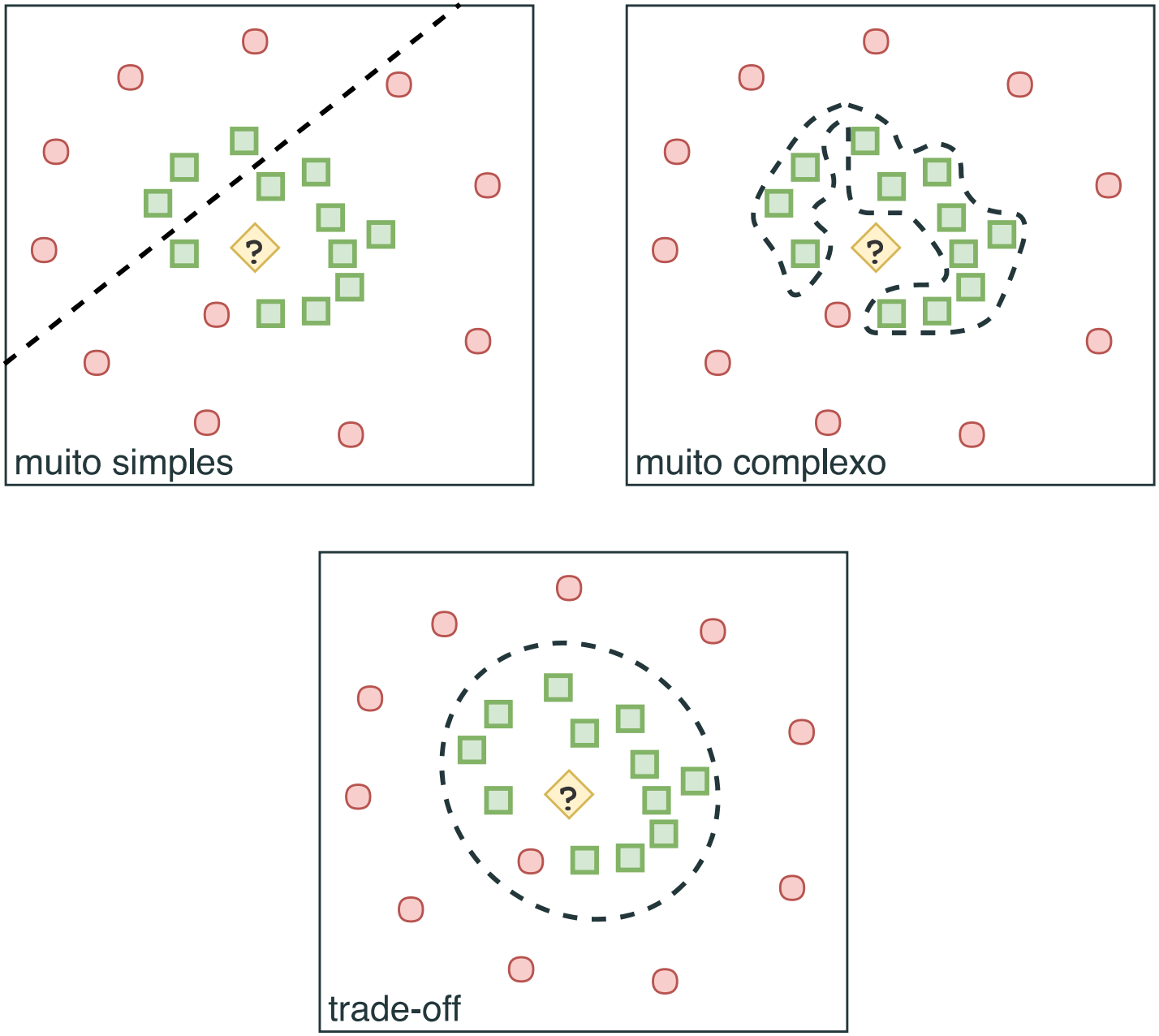
Pré-processamento dos dados

Inicialização dos pesos

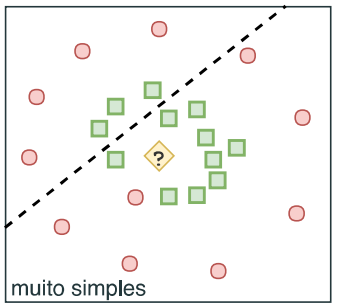
Otimização

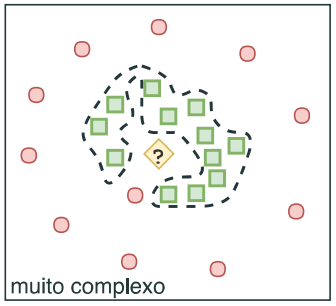
Underfit e Overfit

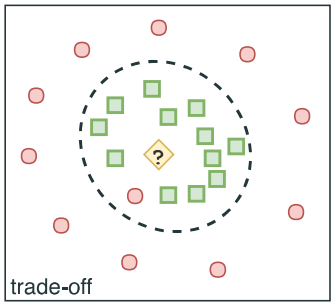
Underfit e Overfit

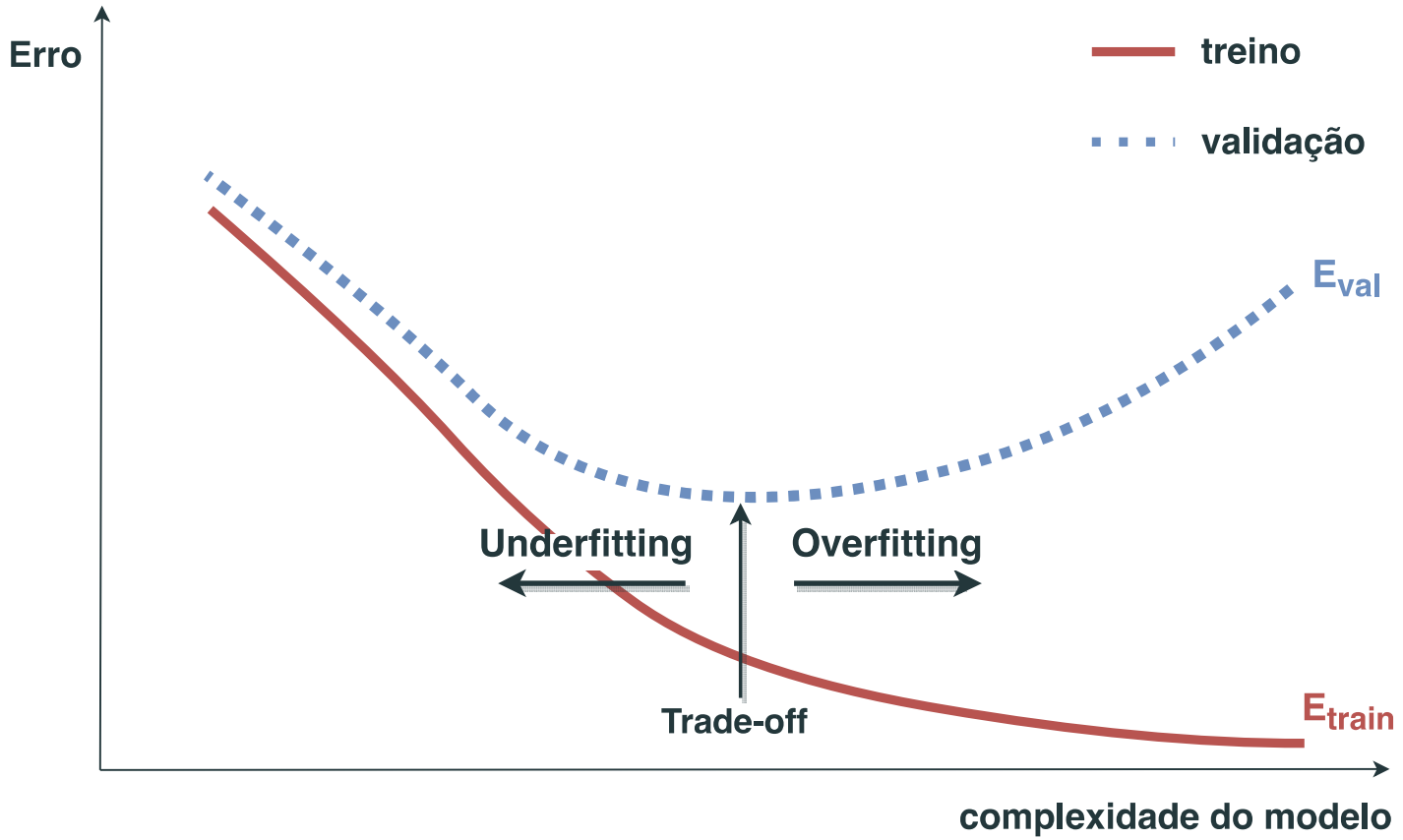


Underfit e Overfit

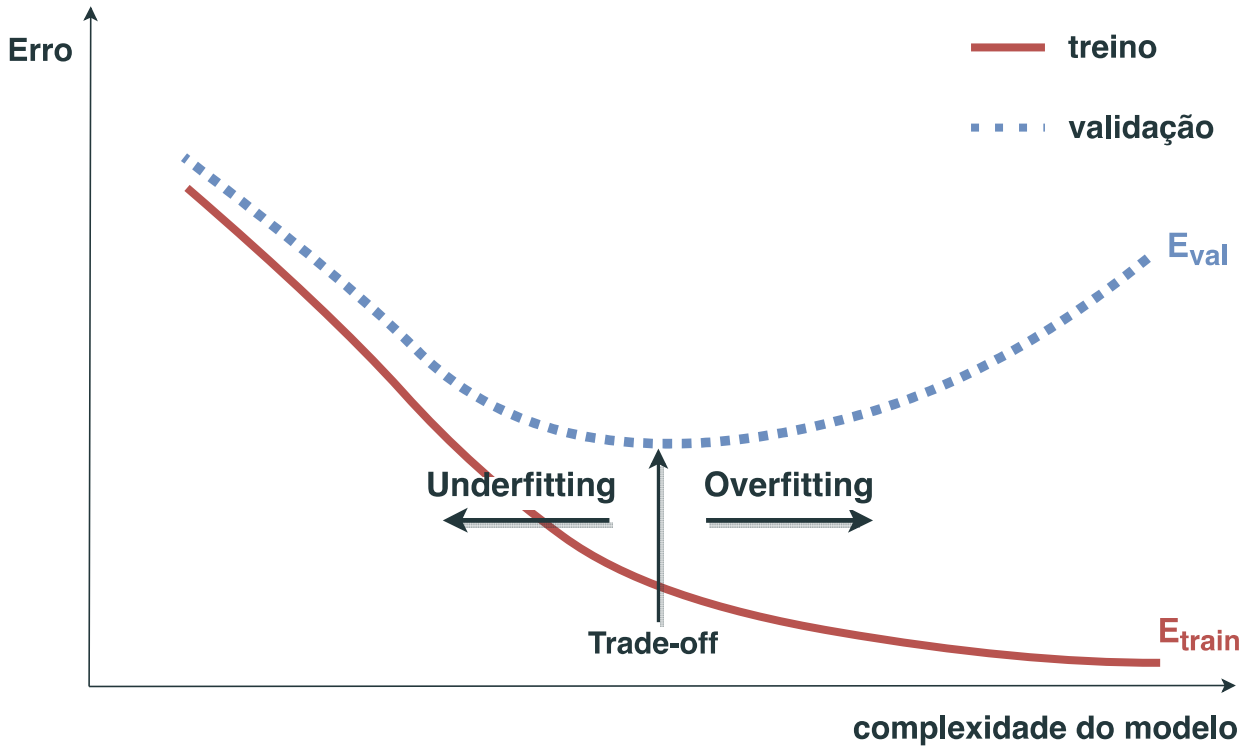








Underfit e Overfit



Underfit

E_{train} alto

$E_{val} \approx E_{train}$

Overfit

E_{train} baixo

$E_{val} \gg E_{train}$

Underfit e Overfit

Receita

Underfit: observar erro no conjunto de treinamento

- mais épocas de treinamento
- arquitetura diferente (mais profunda)

Underfit e Overfit

Receita

Underfit: observar erro no conjunto de treinamento

- mais épocas de treinamento
- arquitetura diferente (mais profunda)

Overfit: observar erro no conjunto de validação

- mais dados
- arquitetura diferente
- regularização

Regularização

Regularização

Regularização

Qualquer modificação feita na etapa de aprendizado, direcionada a reduzir o erro de generalização, mas não o erro de treinamento.

Regularização

Regularização

Qualquer modificação feita na etapa de aprendizado, direcionada a reduzir o erro de generalização, mas não o erro de treinamento.

Generalização

Quão bem os conceitos aprendidos na etapa de treinamento são aplicáveis a exemplos desconhecidos, a fim de fazer boas predições.

Regularização

Regularização

Qualquer modificação feita na etapa de aprendizado, direcionada a reduzir o erro de generalização, mas não o erro de treinamento.

Generalização

Quão bem os conceitos aprendidos na etapa de treinamento são aplicáveis a exemplos desconhecidos, a fim de fazer boas predições.

$\Rightarrow$ O objetivo da regularização é melhorar a generalização.

Navalha de Occam

Entities are not to be multiplied without necessity!

William of Occam
Franciscan friar and philosopher
1287—1347

Ou seja, entre duas hipóteses que levam às mesmas conclusões, a mais simples é a melhor.

Navalha de Occam

Entities are not to be multiplied without necessity!

William of Occam
Franciscan friar and philosopher
1287—1347

Ou seja, entre duas hipóteses que levam às mesmas conclusões, a mais simples é a melhor.

Em Machine Learning: entre dois modelos semelhantes (com mesma acurácia no treinamento), escolha o mais simples.

Regularização

Técnicas

- Weight decay
  - Regularização L1
  - Regularização L2

Regularização

Técnicas

- Weight decay
  - Regularização L1
  - Regularização L2
- Early stopping

Regularização

Técnicas

- Weight decay
  - Regularização L1
  - Regularização L2
- Early stopping
- Dropout

Regularização

Técnicas

- Weight decay
  - Regularização L1
  - Regularização L2
- Early stopping
- Dropout
- Data augmentation

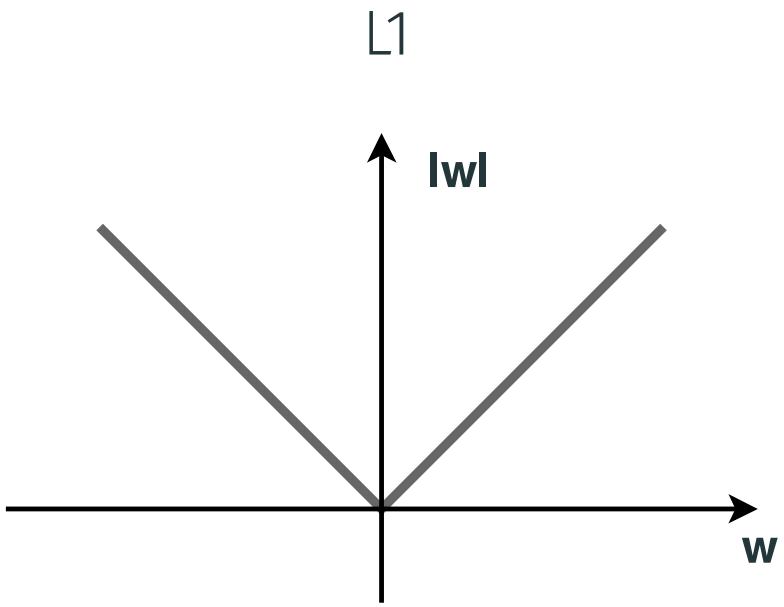
Regularização

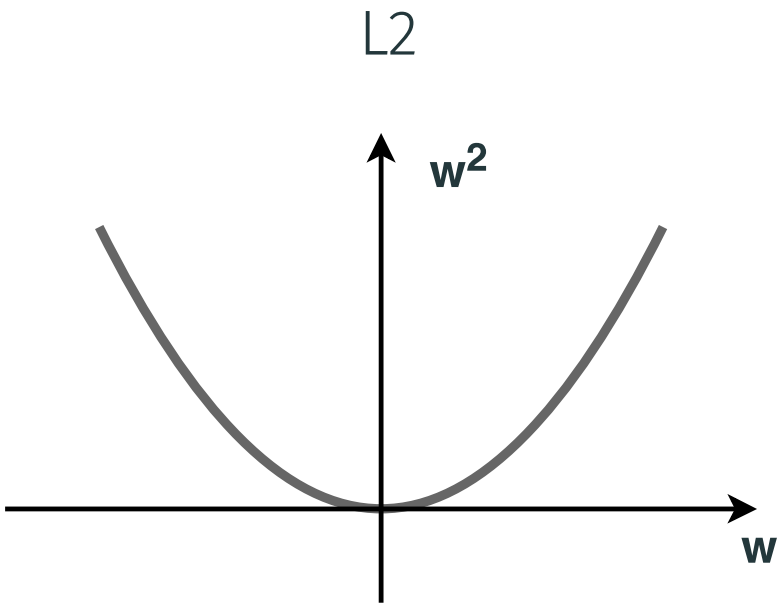
Weight decay

`minimizar(custo + λ regularização)`

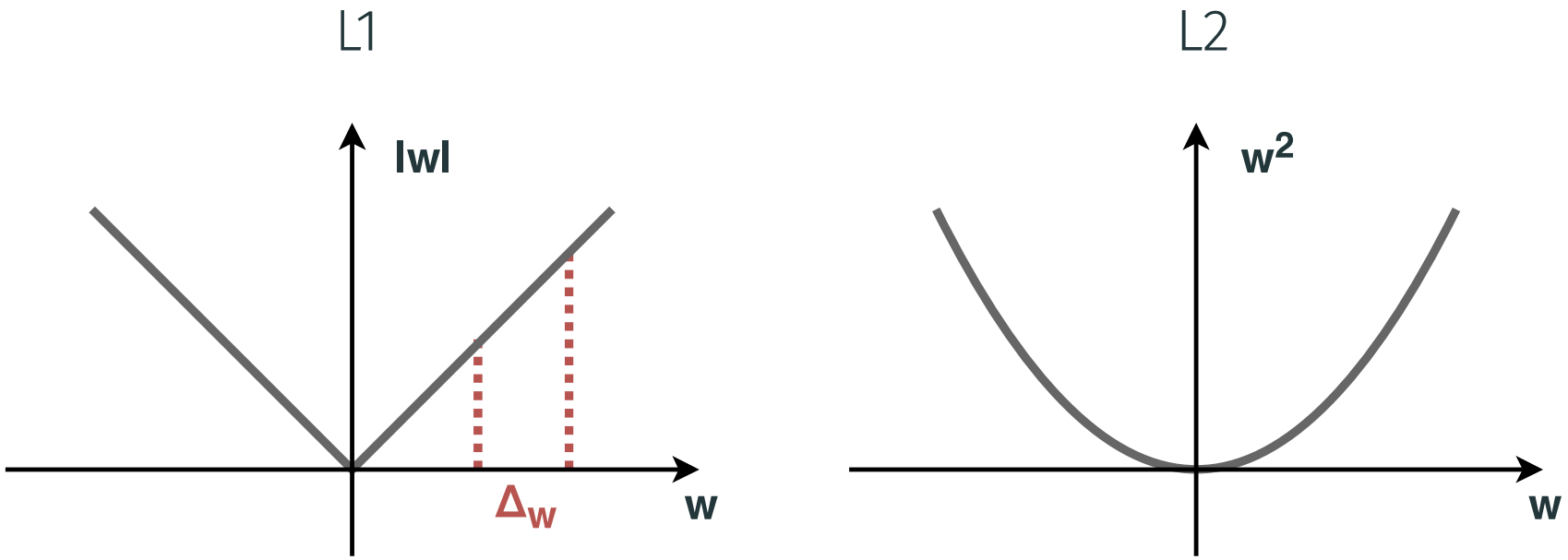
- λ : força da regularização
- regularização:
  - L1: $\sum_i \|w_i\|$
  - L2: $\sum_i w_i^2$

Weight decay

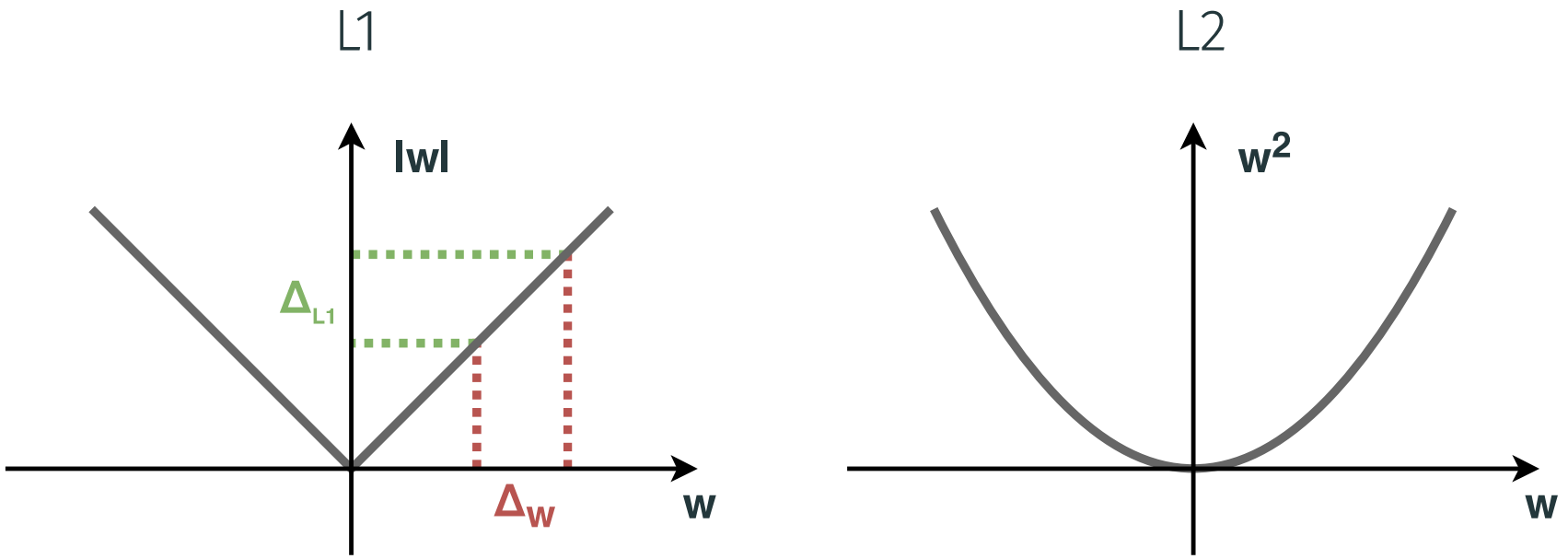




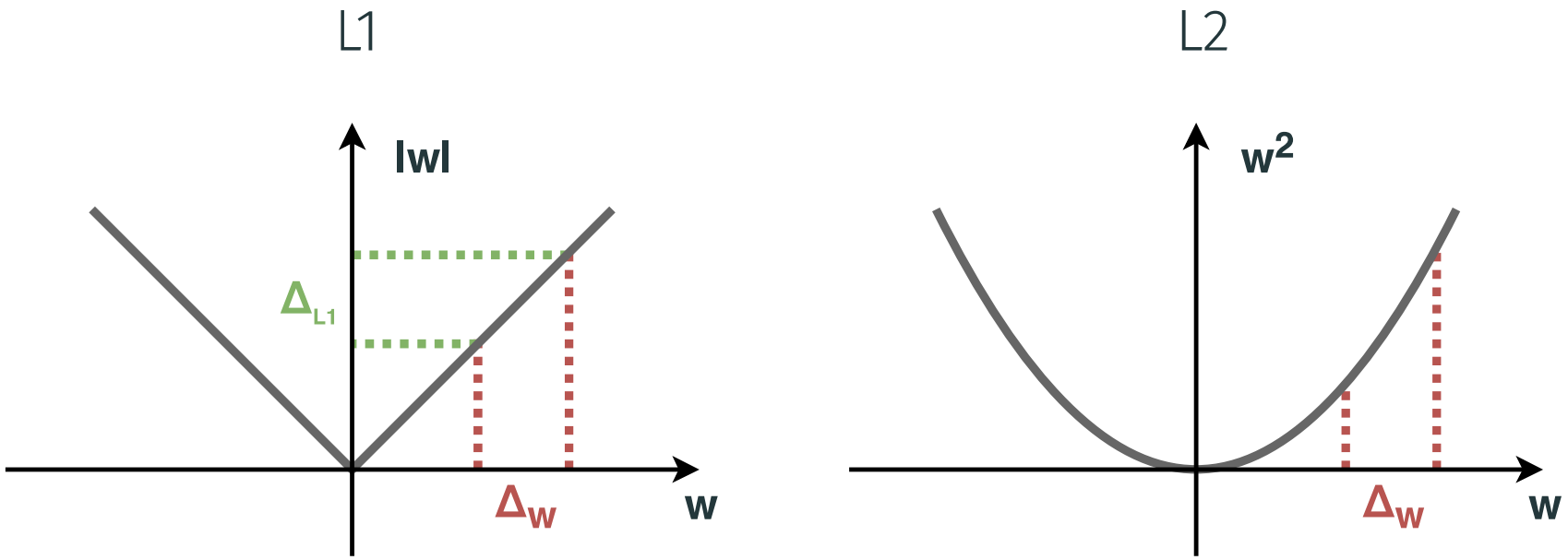
Weight decay



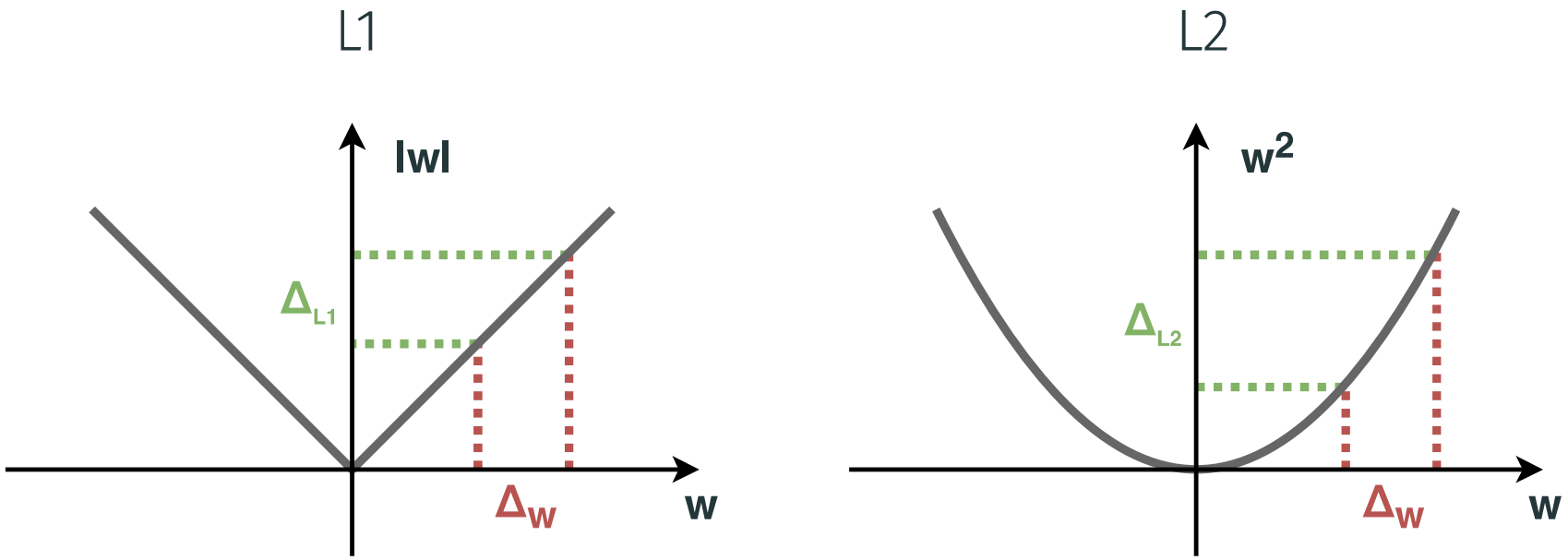
Weight decay



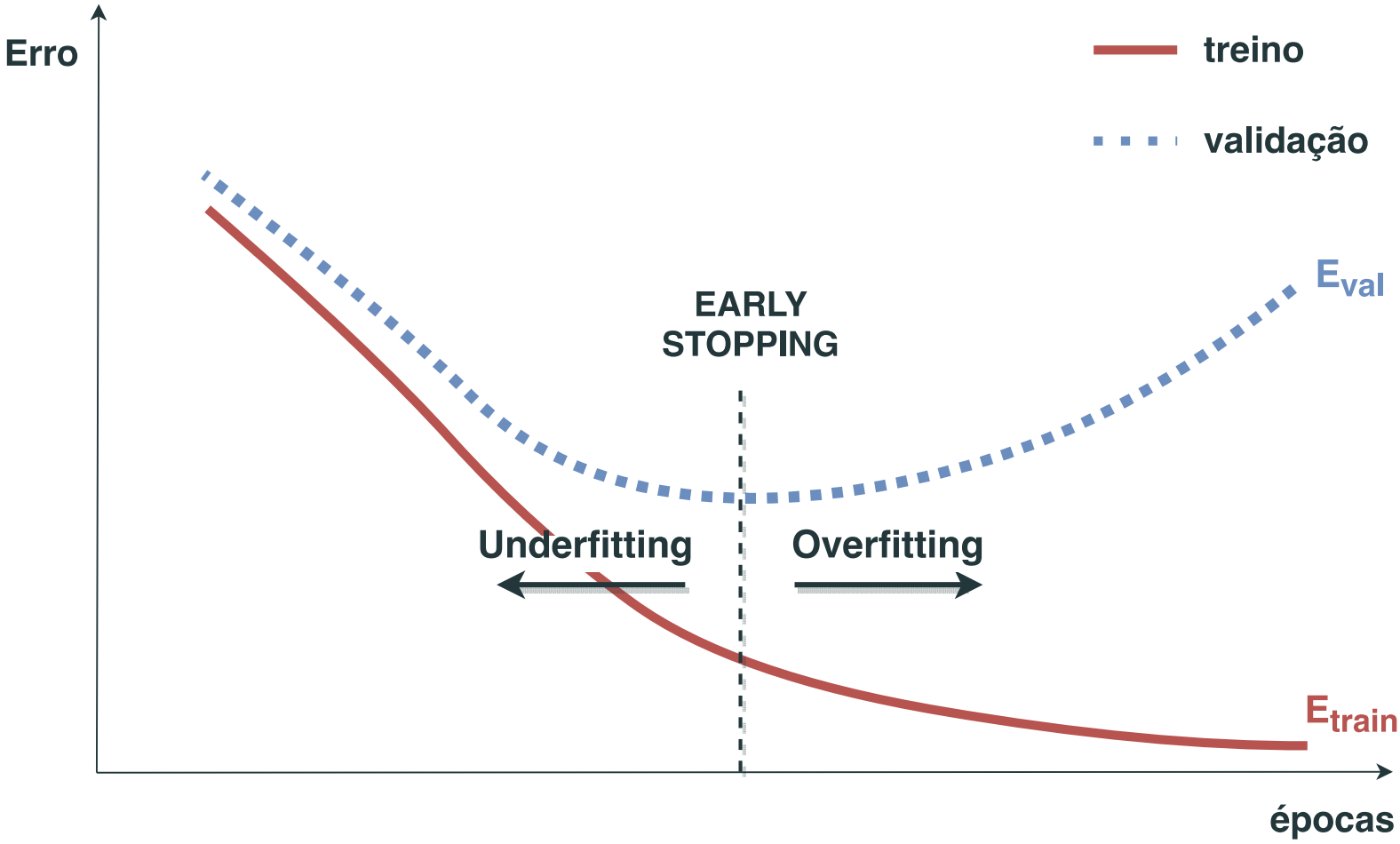
Weight decay



Weight decay

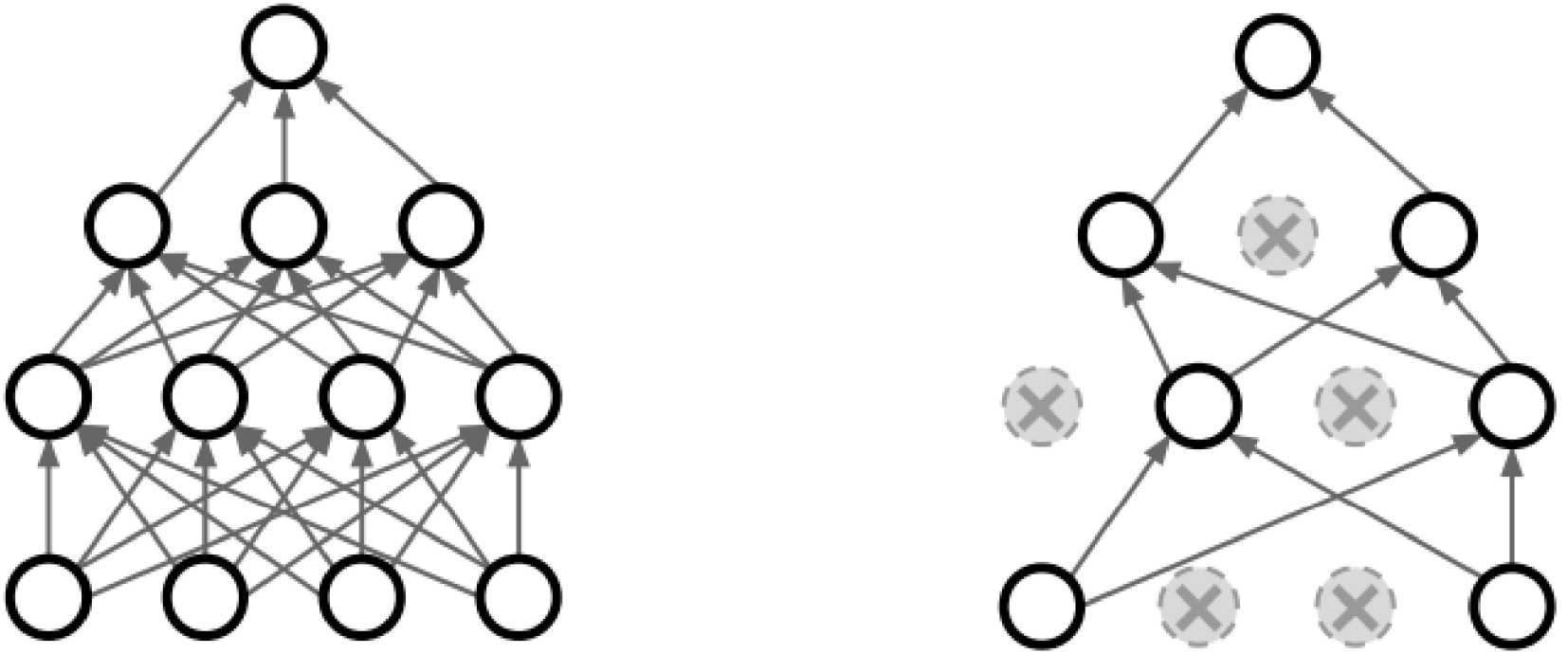


Early stopping

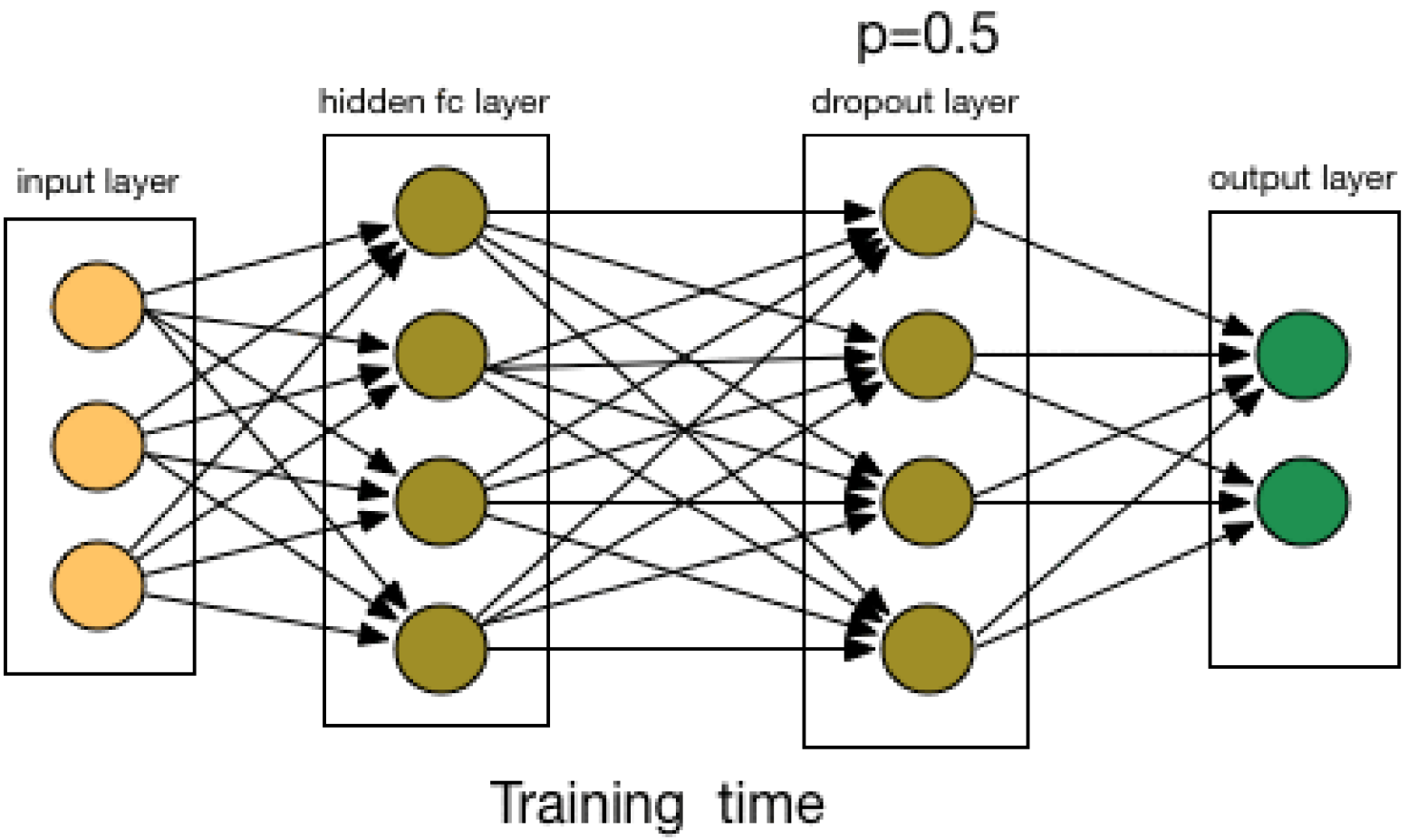


Dropout

Durante o treinamento:



Dropout


p=0.5
hidden fc layer
dropout layer
input layer
output layer
Training time

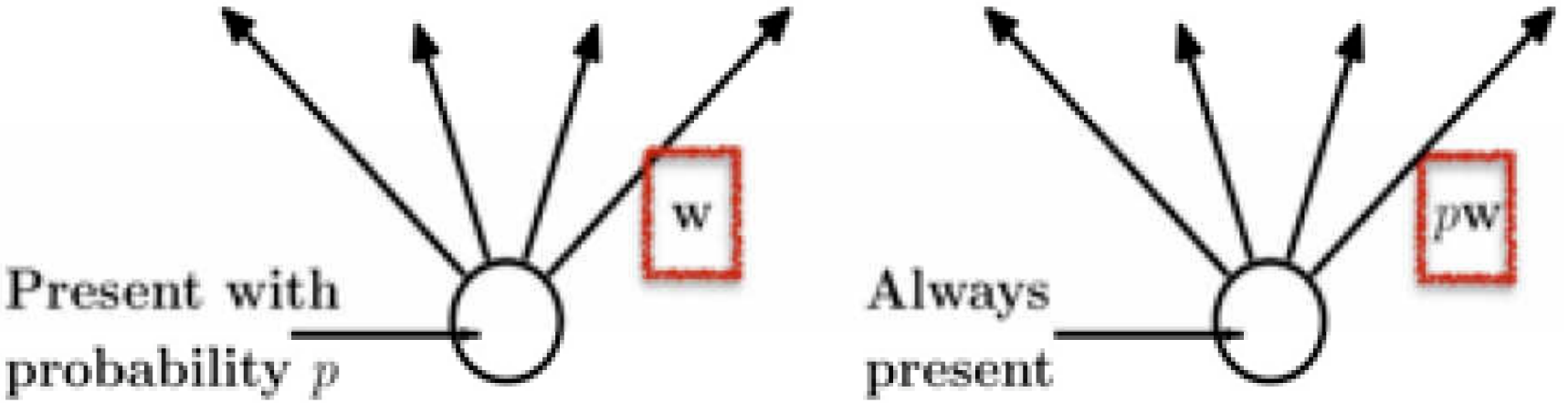
Dropout

- Força a rede a ter representações redundantes.
- Evita a co-adaptação de características.

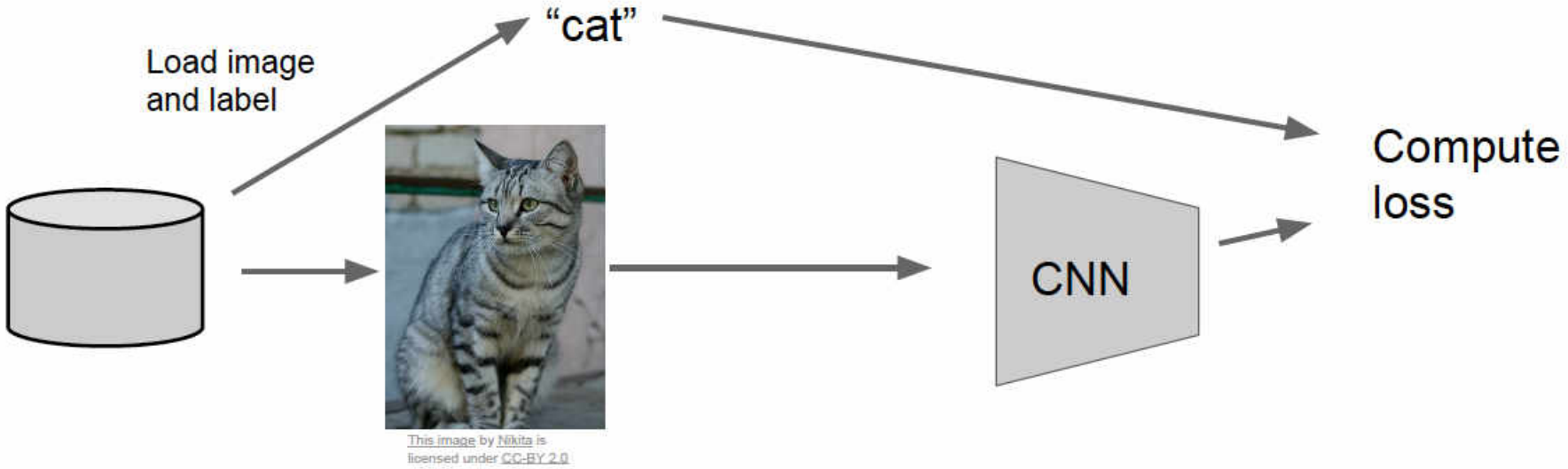
Dropout

Durante o teste:

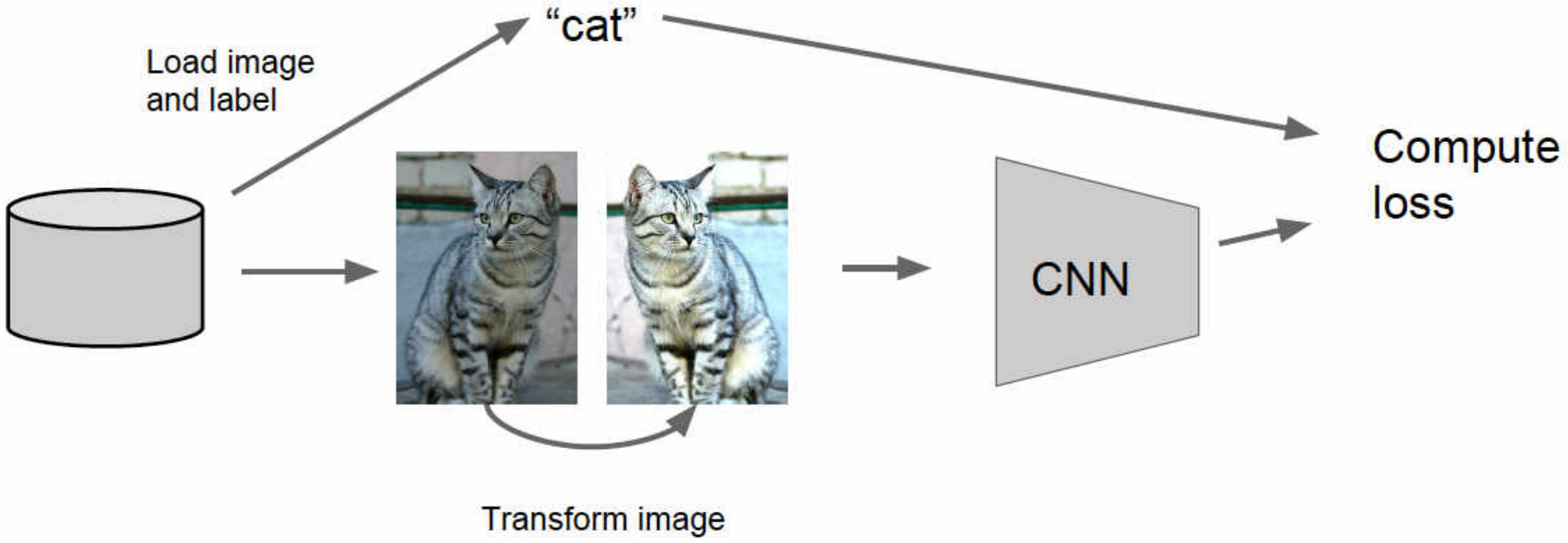
- Todos os neurônios estão ativos.
- Os pesos devem ser escalados (multiplicados) pelo *dropout rate*.



Data augmentation

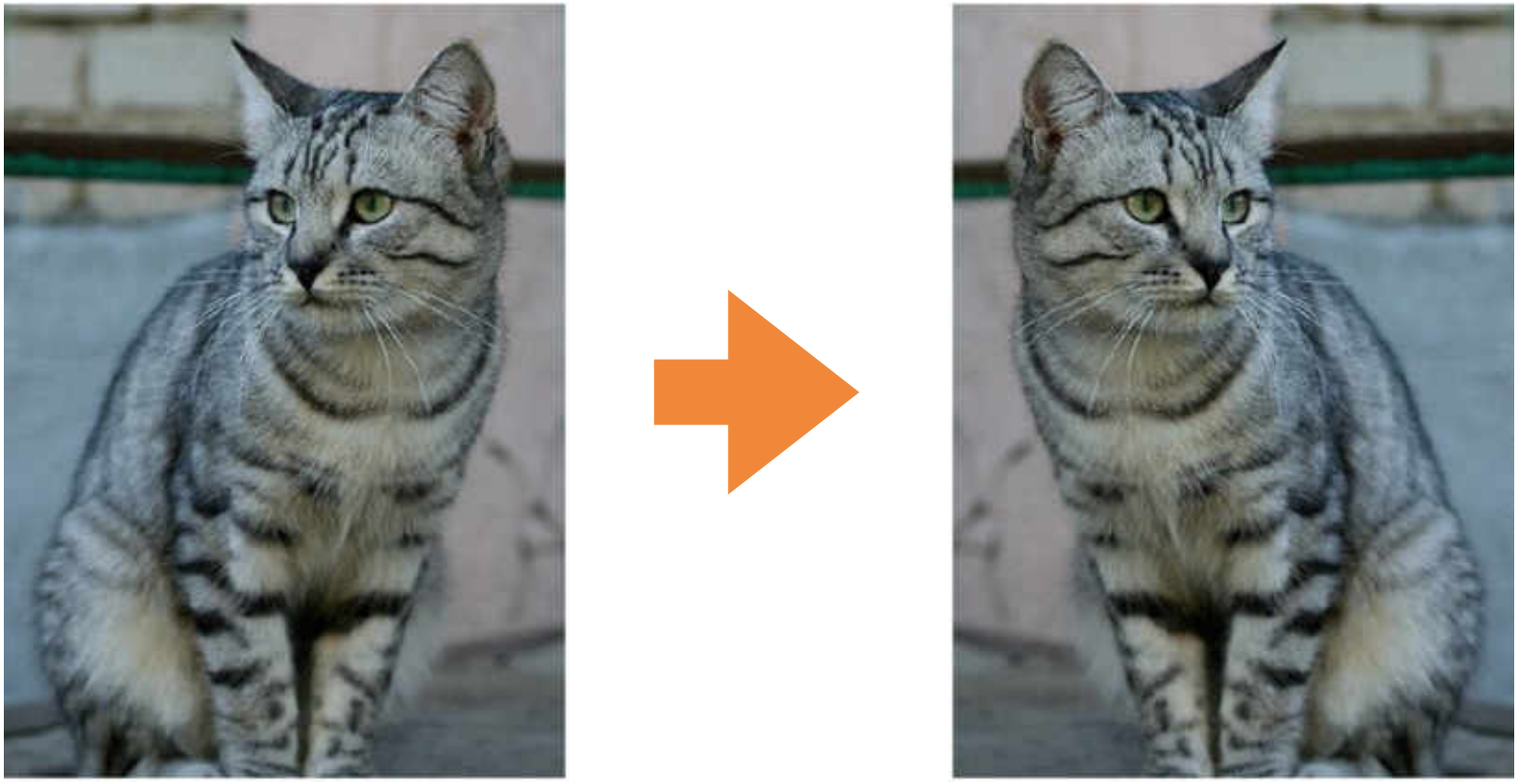


Data augmentation



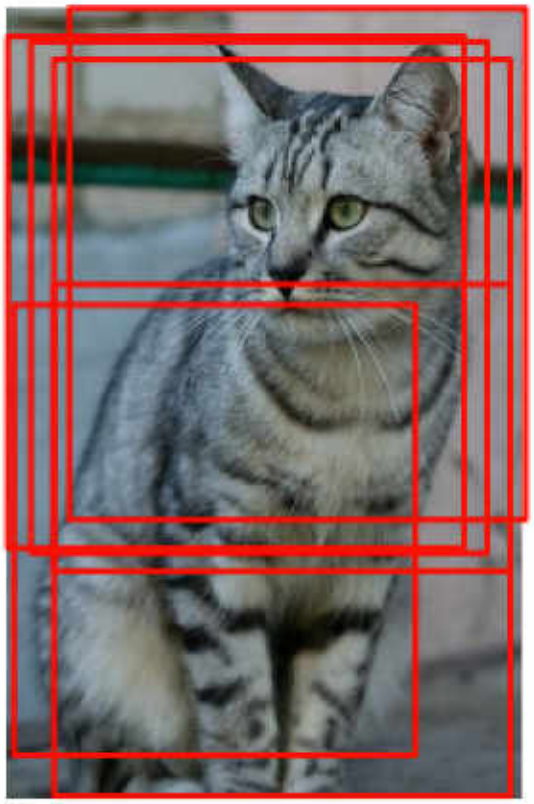
Data augmentation

- Flips horizontais



Data augmentation

- Crops e scales randômicos



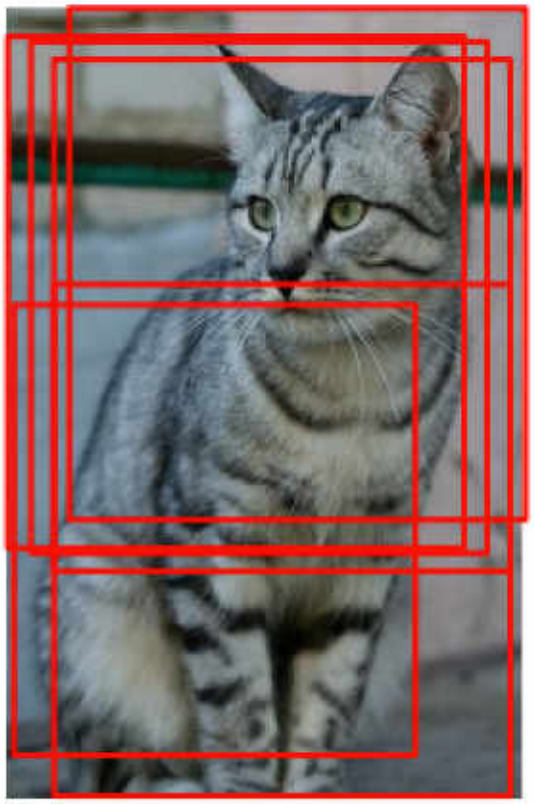
Exemplo ResNet

Treinamento

- Redimensionar imagem de treinamento, menor lado $= L \in [256, 480]$
- Crop randômico de tamanho 224×224

Data augmentation

- Crops e scales randômicos



Exemplo ResNet

Treinamento

- Redimensionar imagem de treinamento, menor lado $= L \in [256, 480]$
- Crop randômico de tamanho 224×224

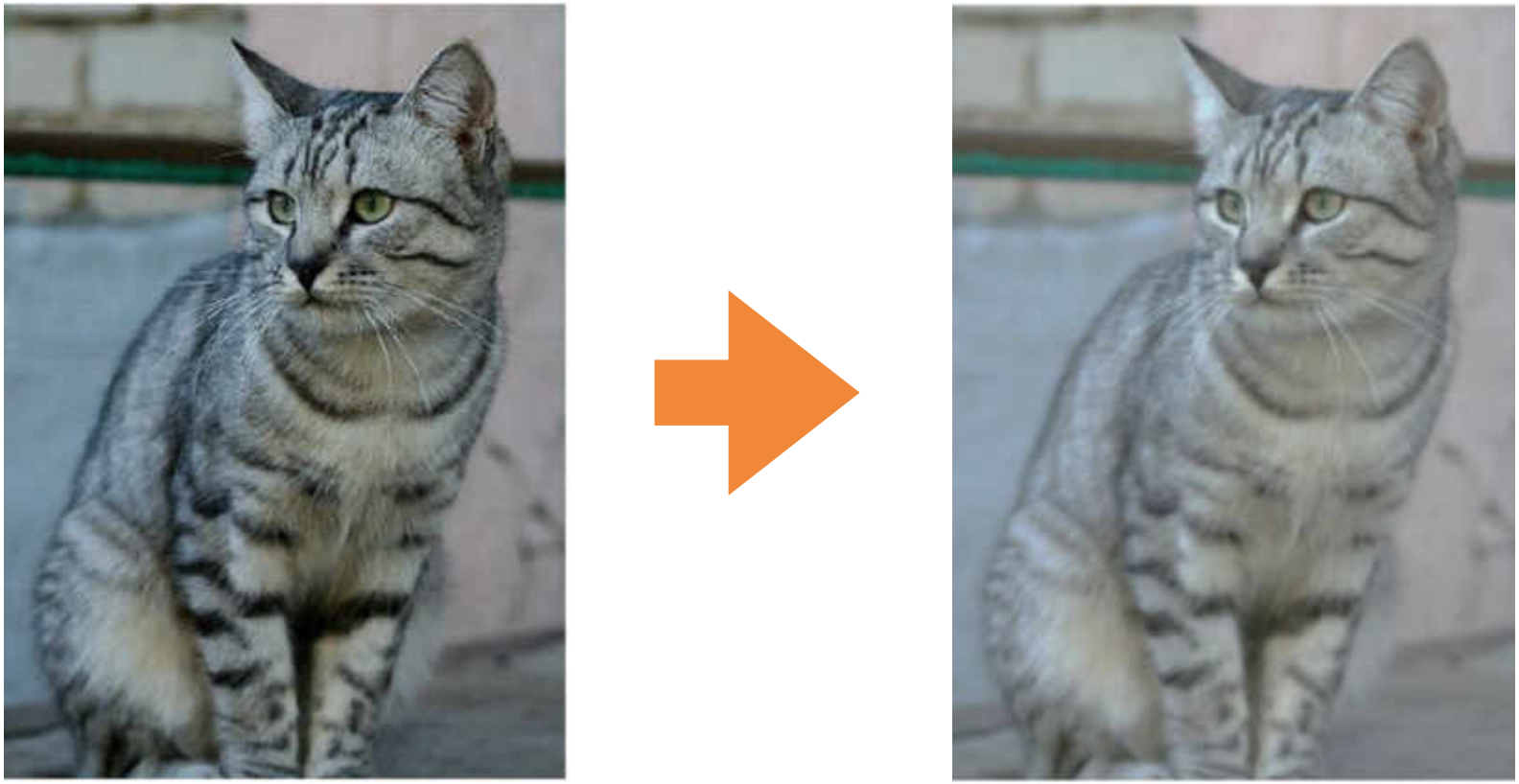
Teste: média dos scores para um conjunto de crops

- Redimensionar imagem de teste 5 vezes: $\{224, 256, 384, 480, 640\}$
- Para cada tamanho: 10 crops 224×224: 4 cantos + centro + flips

Data augmentation

- Variações na cor

  Exemplo simples: randomizar contraste e brilho



Regularização

Data augmentation

- Variações na cor

  Exemplo mais complexo (AlexNet, ResNet, ...):

  - Calcular PCA para todas os pixels $[R, G, B]$ das imagens de treinamento
  - Sortear um "offset de cor" ao longo das direções dos componentes principais
  - Adicionar o offset a todos os pixels de uma imagem de treinamento

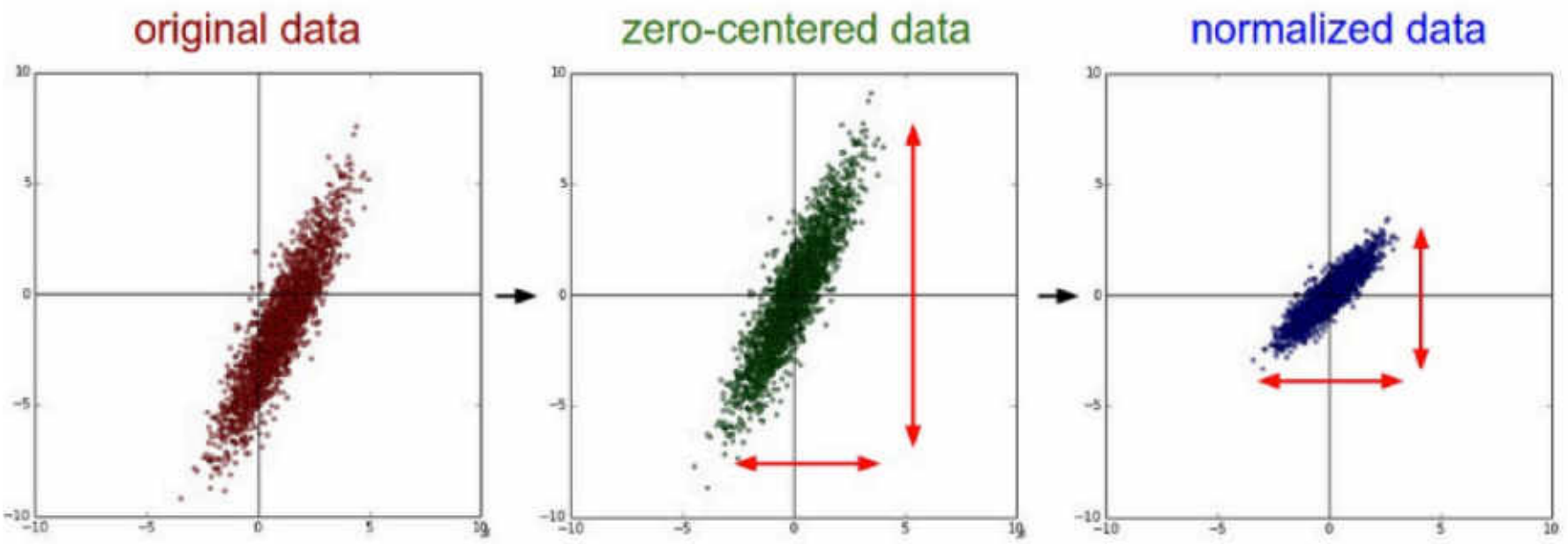
Regularização

Data augmentation

- Combinações randômicas
  - Translação
  - Rotação
  - Stretching
  - Cisalhamento
  - ...

Pré-processamento dos dados

Pré-processamento dos dados



Exemplos:

- AlexNet: subtração da imagem média
- VGGNet: subtração da média dos pixels por canal
- Atenção: a média deve ser calculada no conjunto de treinamento!

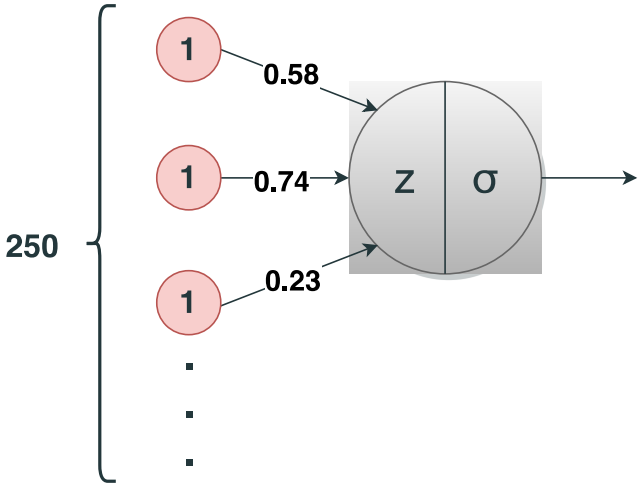
Inicialização dos pesos

Inicialização dos pesos

- Como os pesos são inicializados?
- Qual o impacto no treinamento?
- Podemos mudar a inicialização?

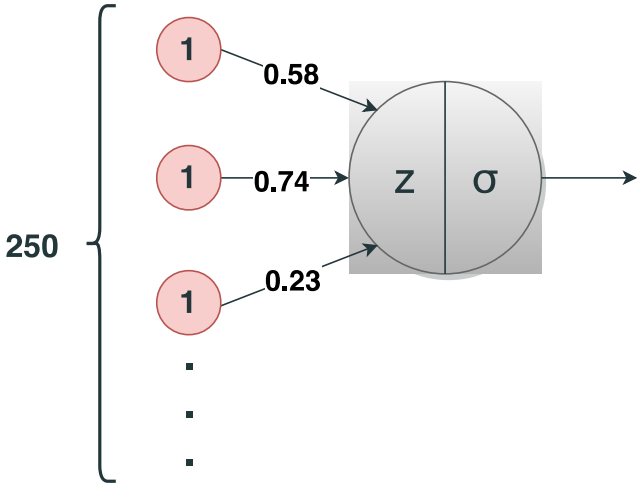
Inicialização dos pesos

Exemplo de inicialização randômica dos pesos

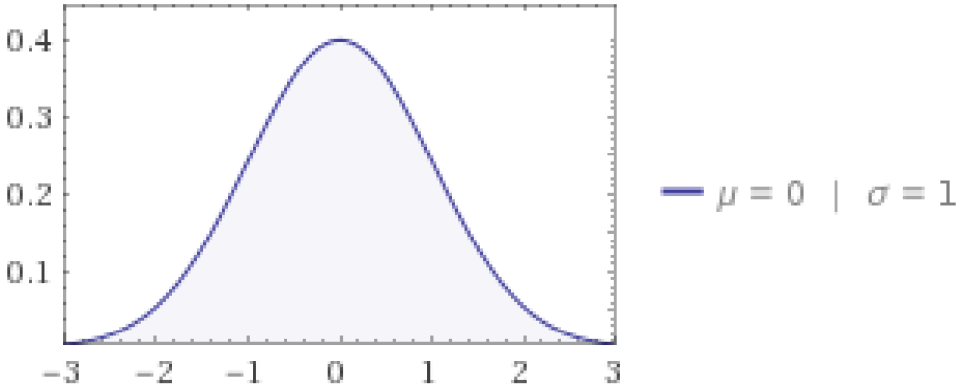


Inicialização dos pesos

Exemplo de inicialização randômica dos pesos

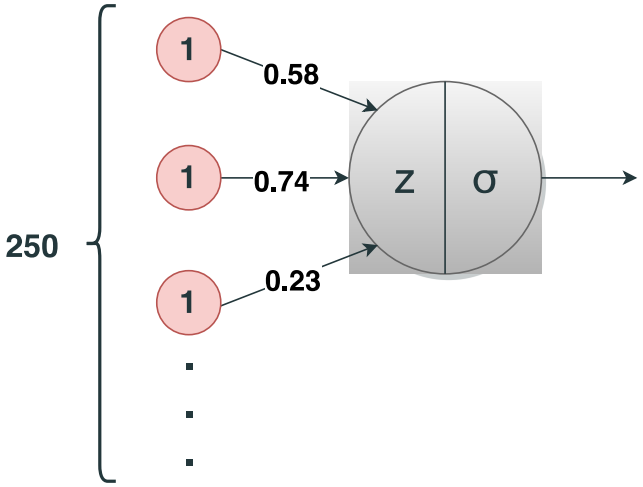


Pesos sorteados de

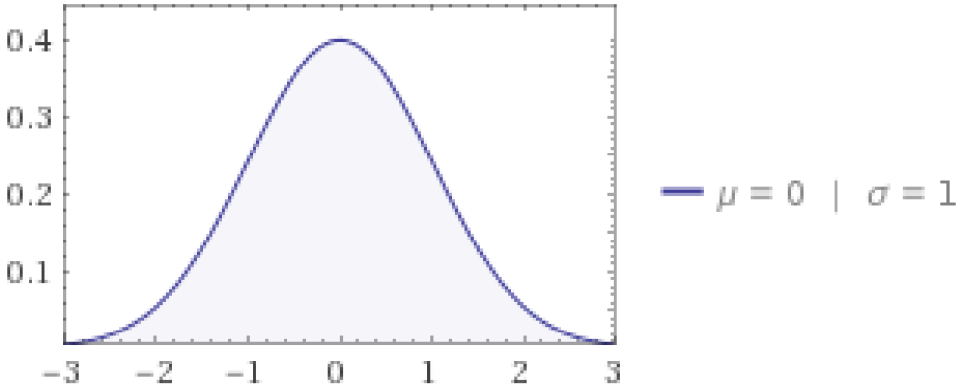


Inicialização dos pesos

Exemplo de inicialização randômica dos pesos



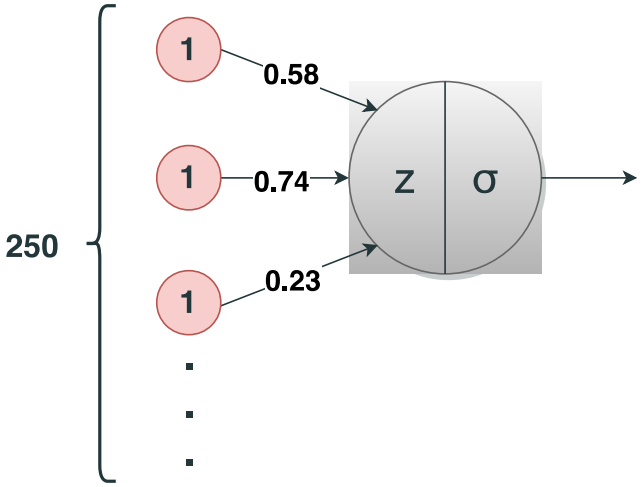
Pesos sorteados de



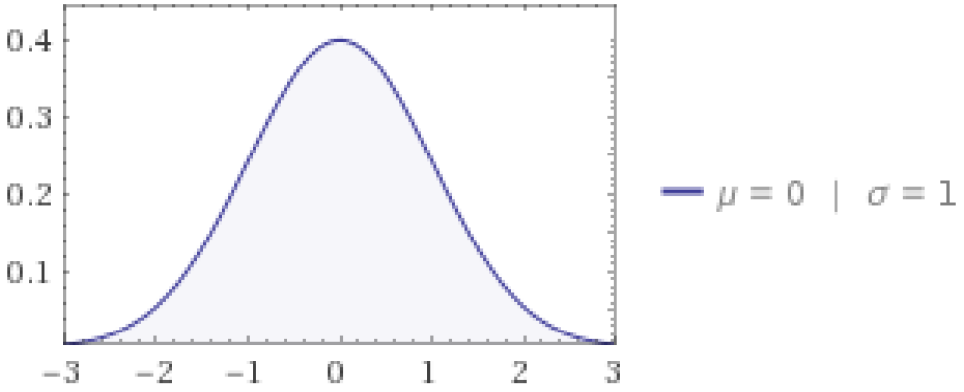
- $var(z) = ?$

Inicialização dos pesos

Exemplo de inicialização randômica dos pesos



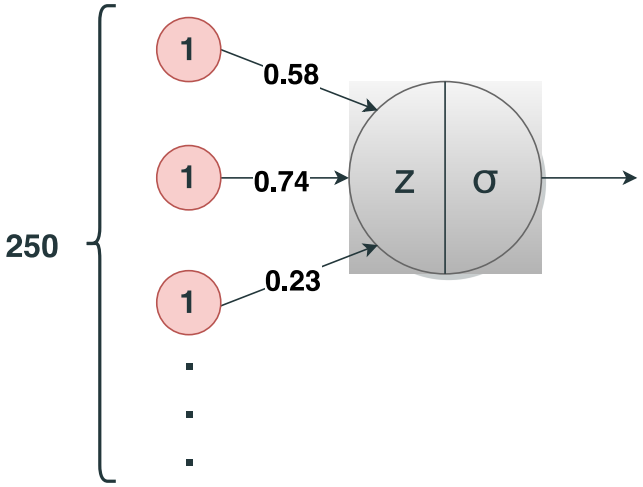
Pesos sorteados de



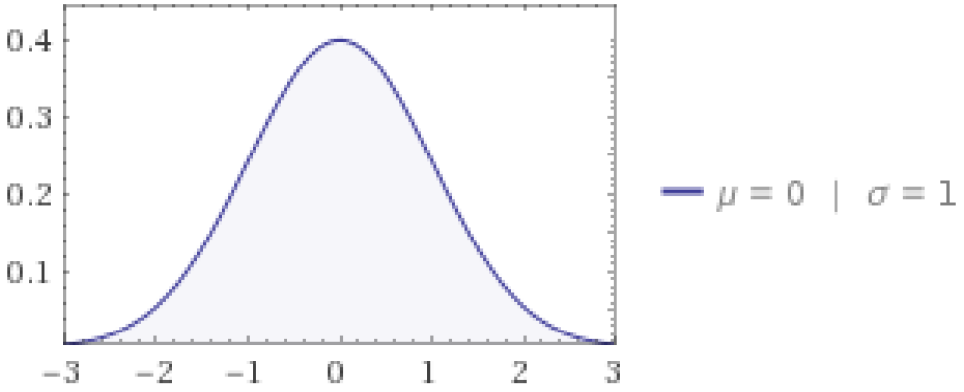
- $var(z) = 250$

Inicialização dos pesos

Exemplo de inicialização randômica dos pesos



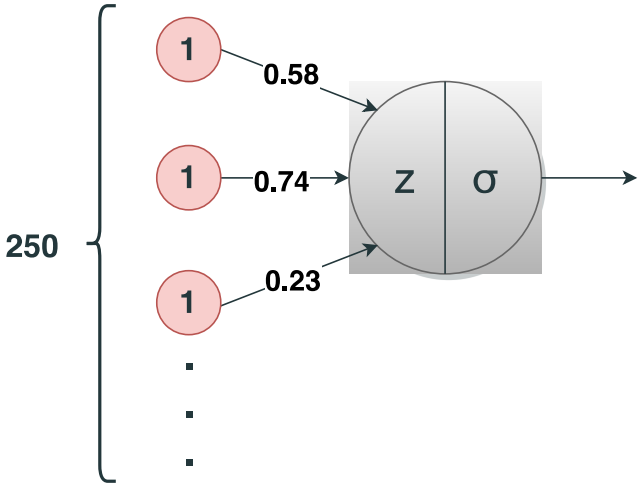
Pesos sorteados de



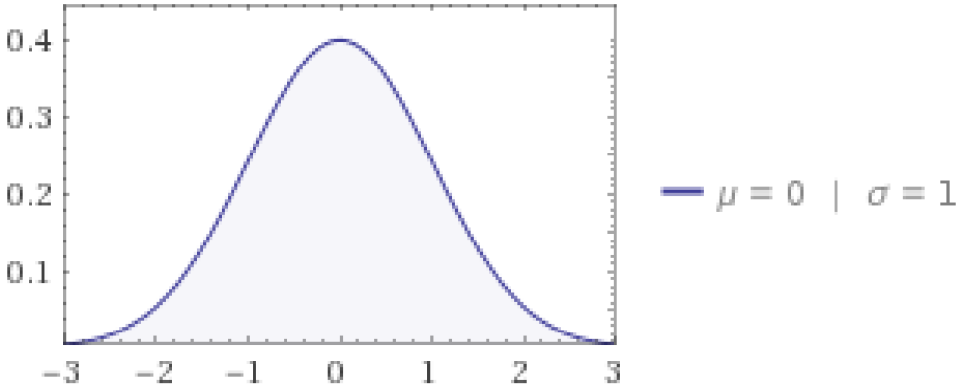
- $var(z) = 250$
- $stdv(z) = \sqrt{250} = 15.81$

Inicialização dos pesos

Exemplo de inicialização randômica dos pesos

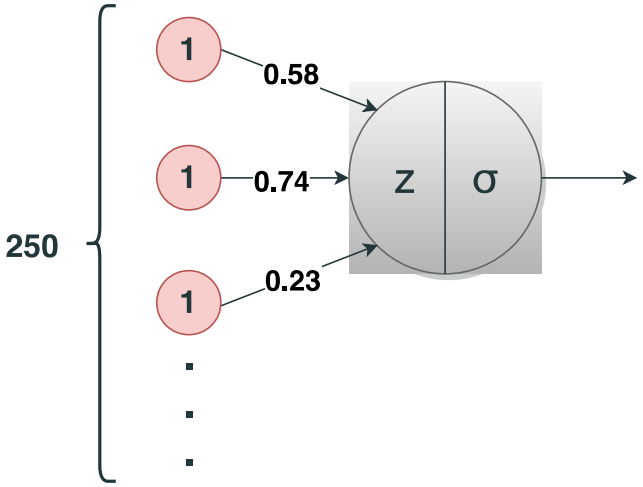


Pesos sorteados de

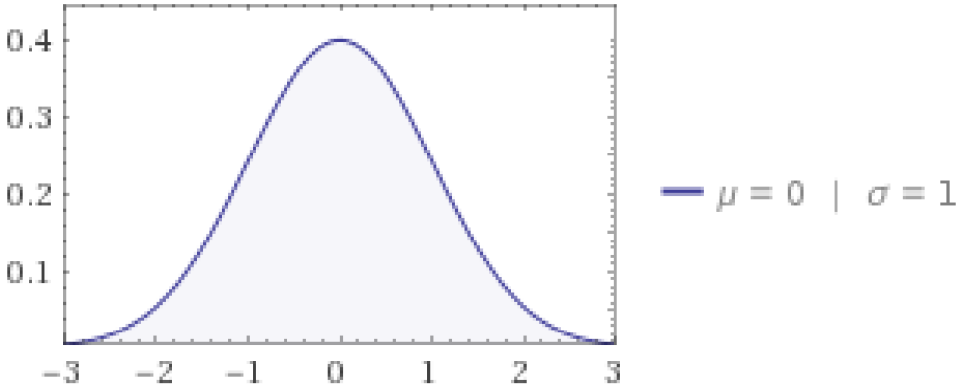


- $var(z) = 250$
- $stdv(z) = \sqrt{250} = 15.81$
- Distribuição de z

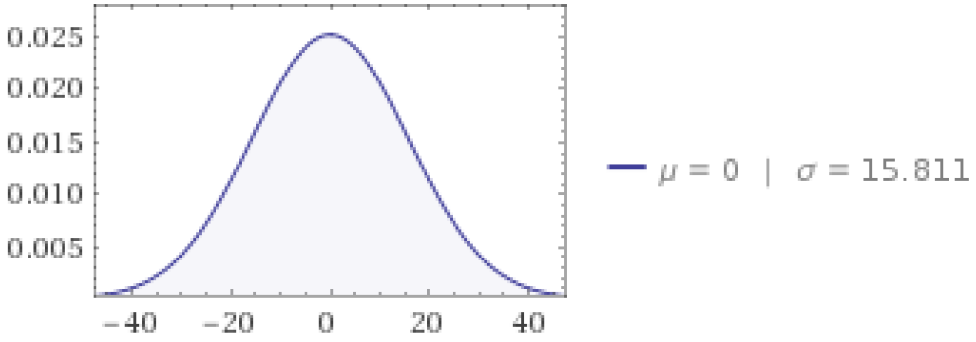
Exemplo de inicialização randômica dos pesos



Pesos sorteados de

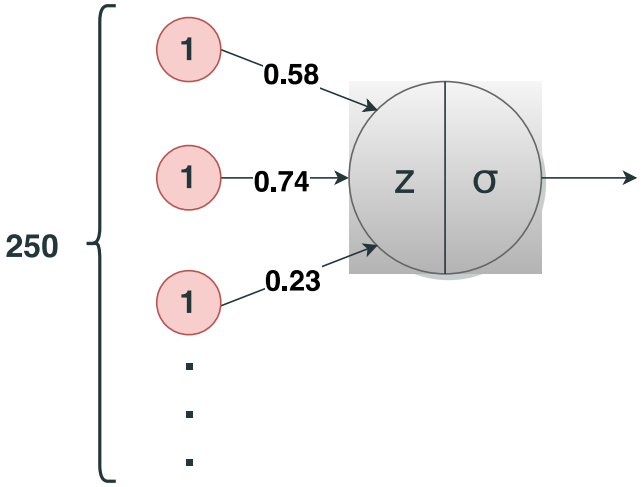


- $var(z) = 250$
- $stdv(z) = \sqrt{250} = 15.81$
- Distribuição de z

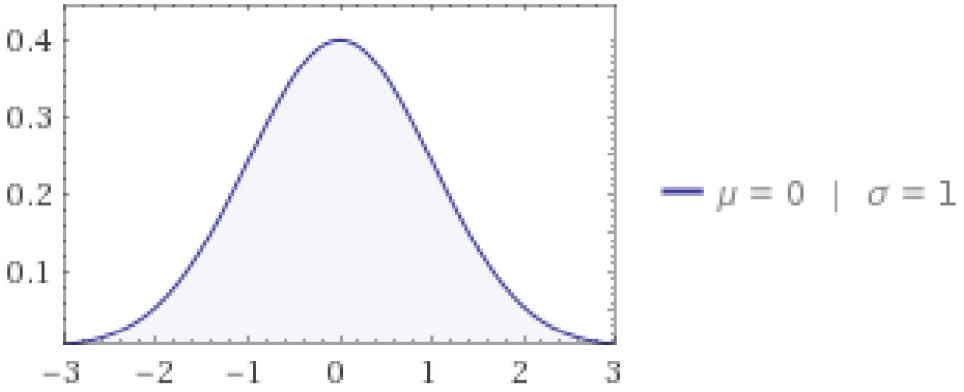


Inicialização dos pesos

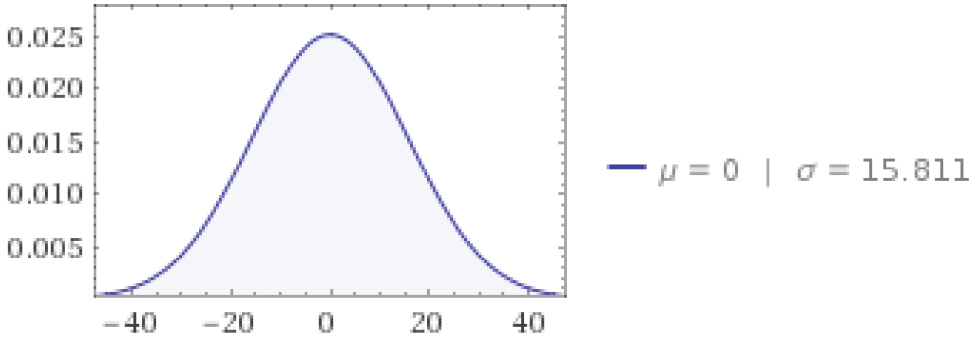
Exemplo de inicialização randômica dos pesos



Pesos sorteados de

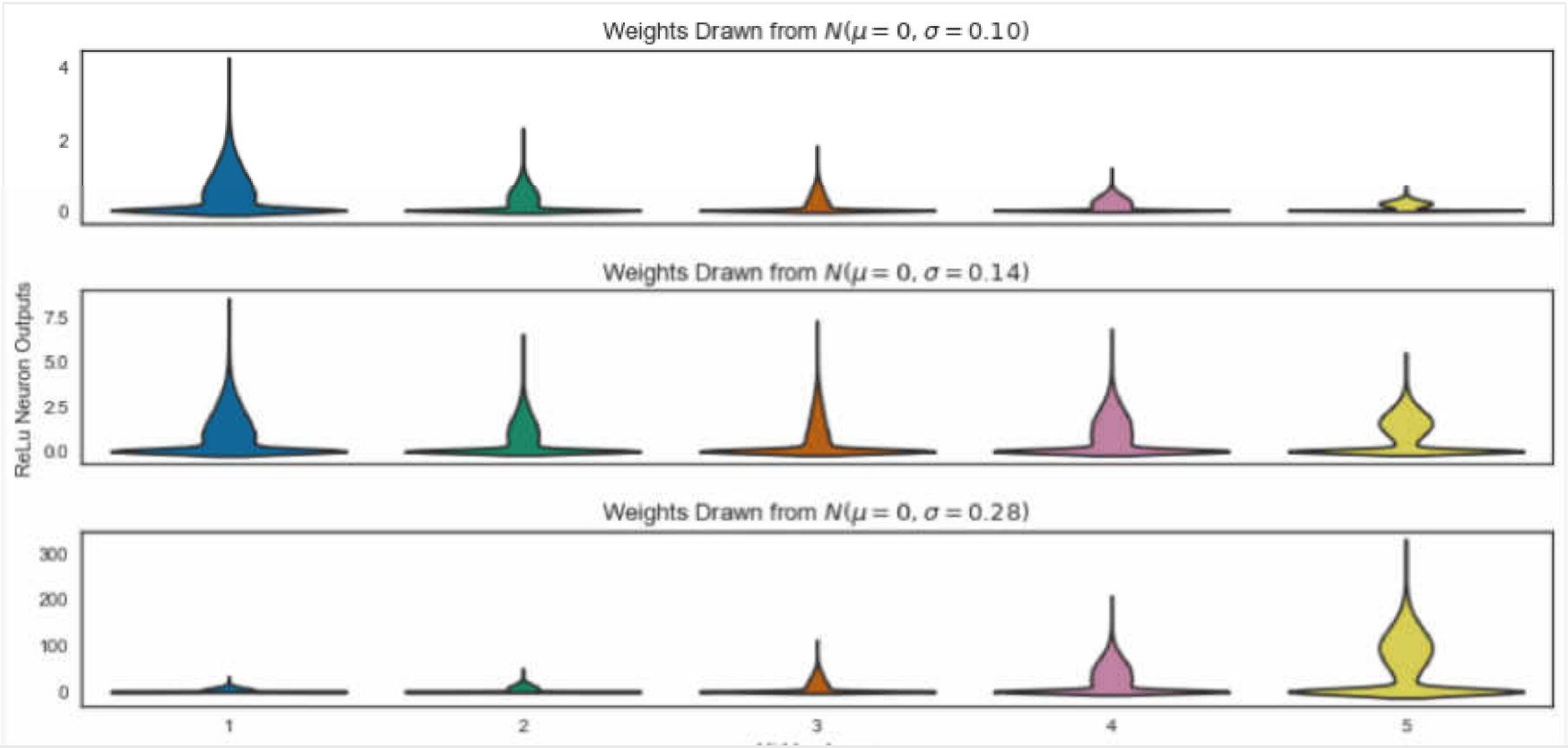


- $var(z) = 250$
- $stdv(z) = \sqrt{250} = 15.81$
- Distribuição de z
- Após σ, neurônio estará saturado.



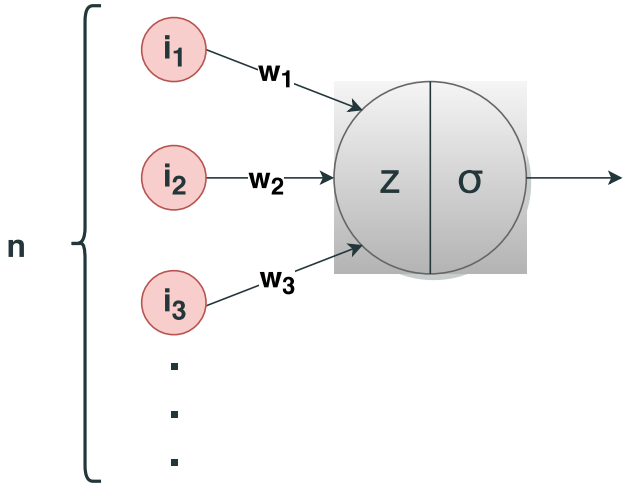
Exemplo de inicialização randômica dos pesos

Ativações de camadas intermediárias de um MLP, após um batch de 1000 imagens do MNIST, com três diferentes inicializações de peso randômicas



Inicialização dos pesos

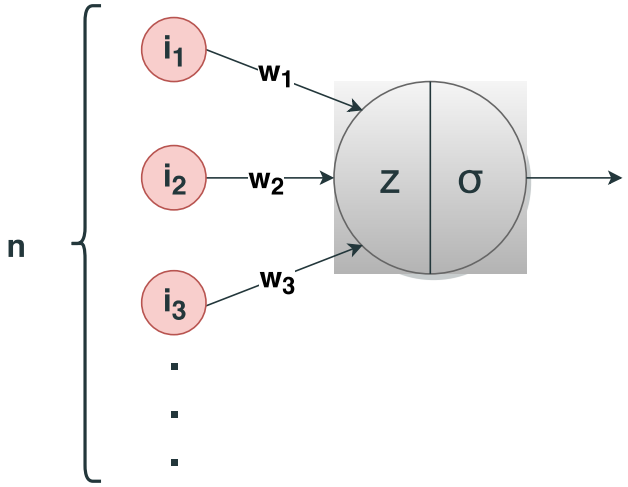
Inicialização Xavier (ou Glorot)



- $var(w) = \frac{2}{n}$ ou $var(w) = \frac{2}{n_{in} + n_{out}}$

Inicialização dos pesos

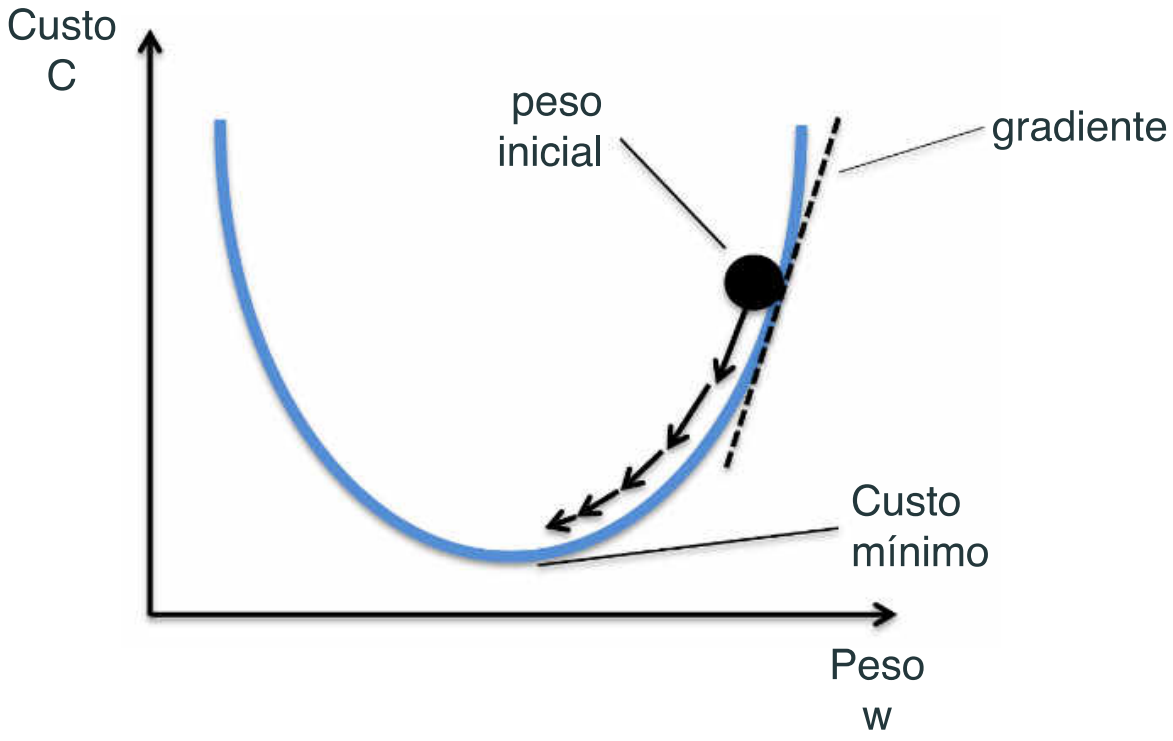
Inicialização Xavier (ou Glorot)



- $var(w) = \frac{2}{n}$ ou $var(w) = \frac{2}{n_{in} + n_{out}}$
- Depois da inicialização randômica, fazer $w_i = w_i * \sqrt{var(w)}$

Otimização

Gradient Descent



Objetivo: Fazer update dos parâmetros θ.

$$\theta = \theta - \eta \nabla C(\theta)$$

Vanilla Gradient Descent

$$\theta = \theta - \eta \nabla C(\theta)$$

```
for i in range(nb_epochs):
    params_grad = evaluate_gradient(loss_function, data, params)
    params = params - learning_rate * params_grad
```

- Update de parâmetros: gradiente da função de custo em relação aos parâmetros θ é computado para todo o conjunto de treinamento.
- Lento e não aplicável a datasets muito grandes.

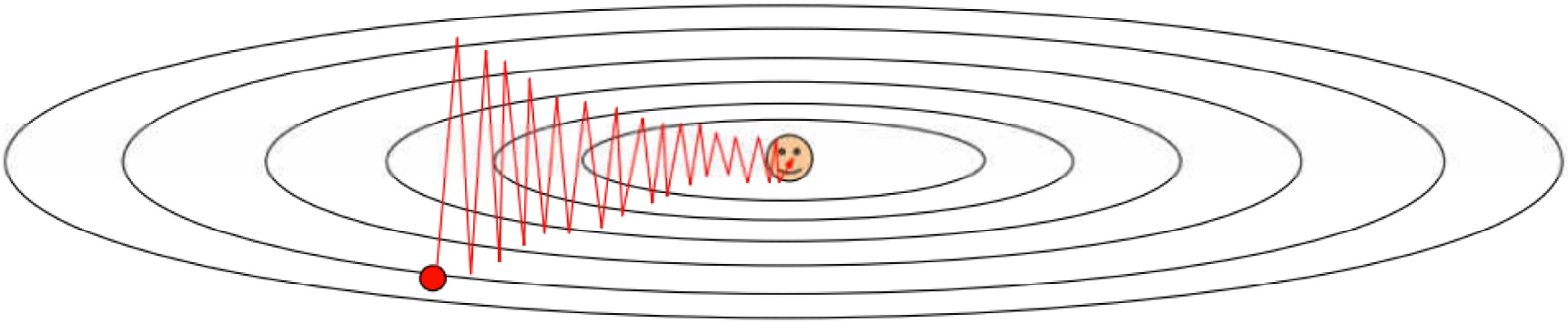
Stochastic Gradient Descent

$$\theta = \theta - \eta \nabla C(\theta; x_i; y_i)$$

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for example in data:
        params_grad = evaluate_gradient(loss_function, example, params)
        params = params - learning_rate * params_grad
```

- Update de parâmetros: para cada sample de treinamento.
- SGD faz updates frequentes com variância alta, causando flutuação da função objetivo.

Stochastic Gradient Descent



Mini batch Gradient Descent

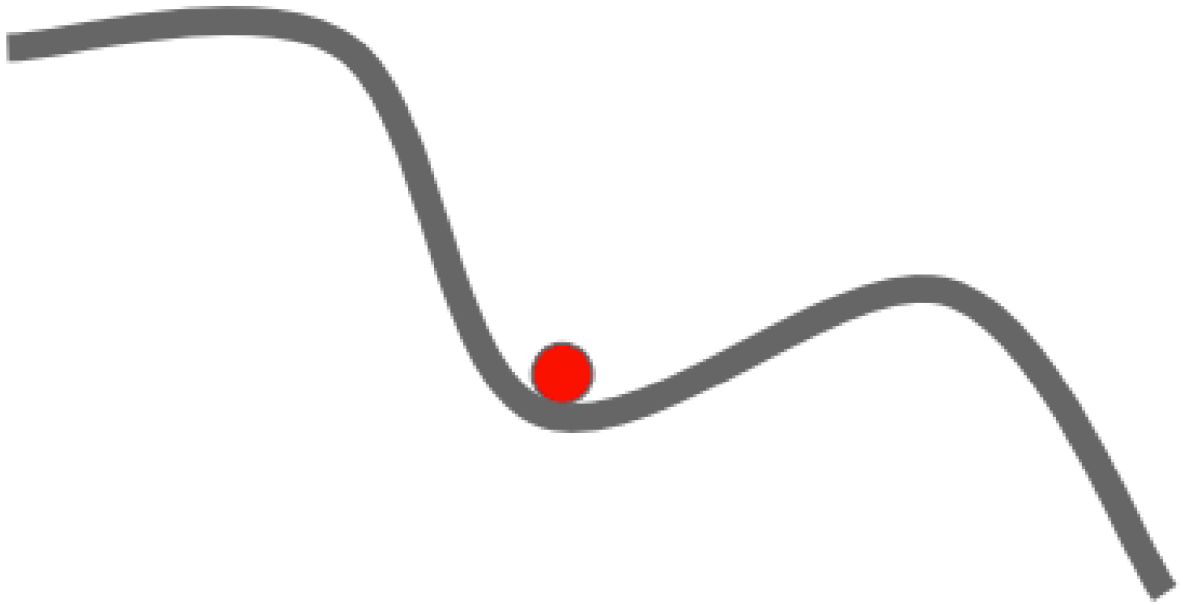
Batch de tamanho c: $\theta = \theta - \eta \nabla C(\theta; x_i : x_{i+c}; y_i : y_{i+c})$

```
for i in range(nb_epochs):
    np.random.shuffle(data)
    for batch in get_batches(data, batch_size=50):
        params_grad = evaluate_gradient(loss_function, batch, params)
        params = params - learning_rate * params_grad
```

- Update de parâmetros: para cada batch.
- Reduz a variância do update, levando a uma convergência mais estável.
- Mais utilizado para treinamento de CNNs.

Desafios do Gradient Descent

O que ocorre se a função objetivo tiver um mínimo local?



⇒ Gradiente zero, Gradient Descent fica preso.

Desafios do Gradient Descent

- Não ficar preso em mínimo locais.
- Escolher um learning rate η apropriado.
- Escolher como variar η durante o treinamento.

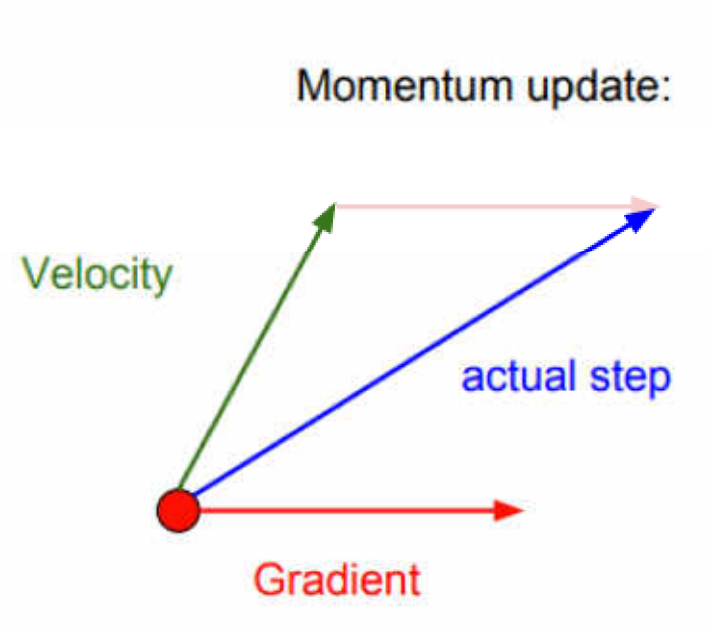
Momentum

$$V_t = \gamma V_{t-1} + \eta \nabla C(\theta)$$

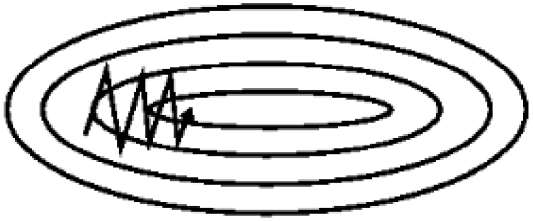
$$\theta = \theta - V_t$$

- Acelera o SGD, navegando em direções relevantes e suavizando oscilações em direções irrelevantes.
- Adiciona uma fração γ do último update ao update atual.
- Default: $\gamma = 0.9$

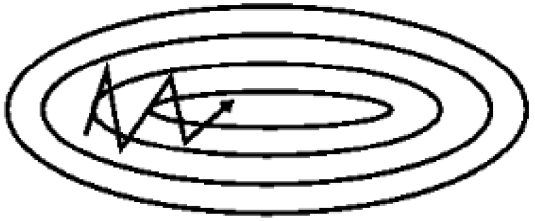
Momentum



Sem momentum



Com momentum



Adagrad

- Utiliza um learning rate diferente para cada parâmetro θ_i a cada passo t (parâmetro $\theta_{t,i}$).

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$g_{t,i}$: gradiente da loss em respeito a $\theta_{t,i}$

$G_{t,ii}$: matriz onde cada elemento da diagonal é a soma dos quadrados dos gradientes em respeito a θ_i até o passo t

Adagrad

- Utiliza um learning rate diferente para cada parâmetro θ_i a cada passo t (parâmetro $\theta_{t,i}$).

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

$g_{t,i}$: gradiente da loss em respeito a $\theta_{t,i}$

$G_{t,ii}$: matriz onde cada elemento da diagonal é a soma dos quadrados dos gradientes em respeito a θ_i até o passo t

- Vantagem: Learning rate não precisa ser ajustado.
- Desvantagem: Por causa da soma positiva no denominador, o learning rate encolhe até se tornar 0.

Corrigindo o Adagrad

- Ao invés de acumular o quadrado de todos os gradientes passados, Adadelta restringe o acúmulo dos gradientes a uma janela.
- RMSprop divide o learning rate por uma média dos gradientes quadrados com caimento exponencial.

Adam

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$, m_t é a média dos gradientes até o passo t.

$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$, v_t é a variância dos gradientes até o passo t.

Adam

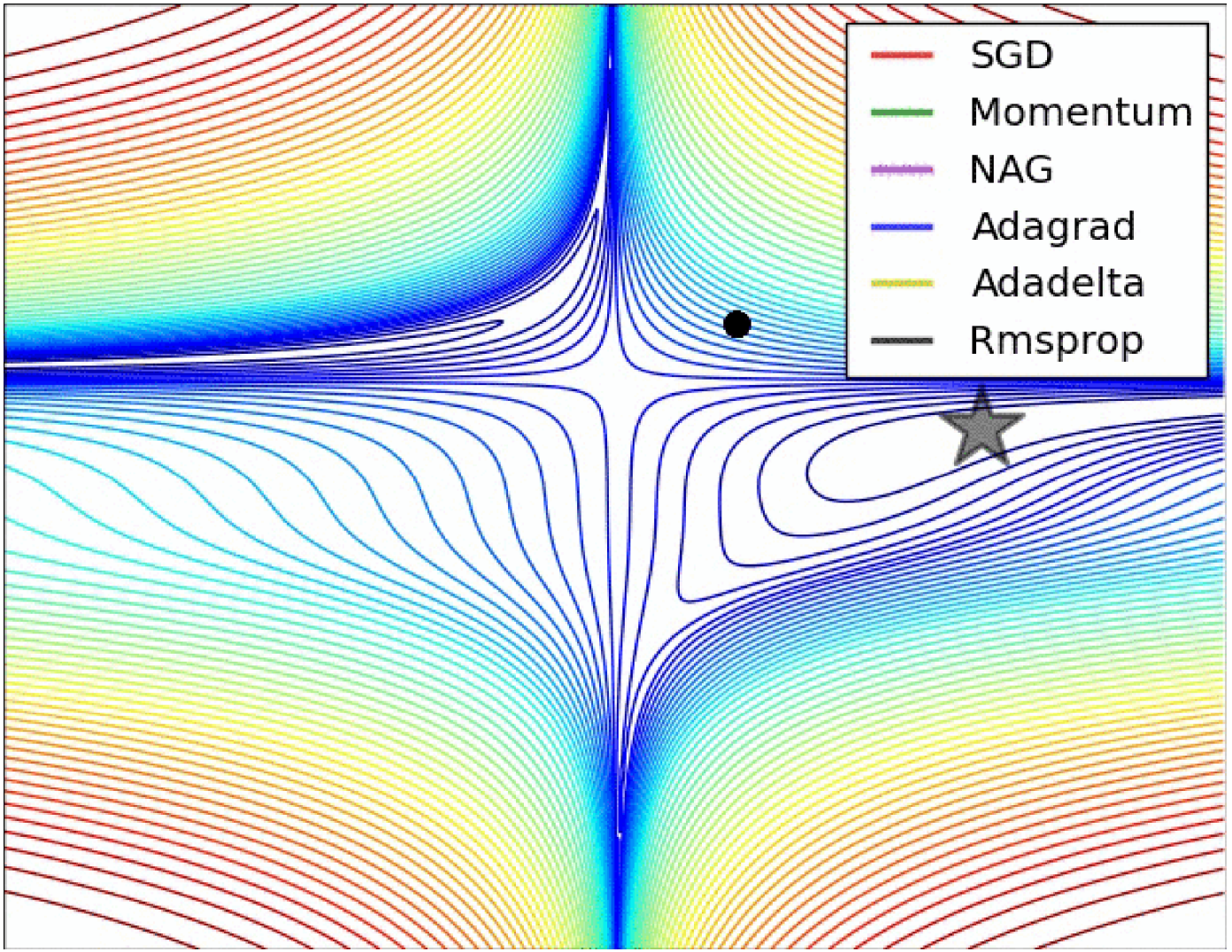
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$, m_t é a média dos gradientes até o passo t.

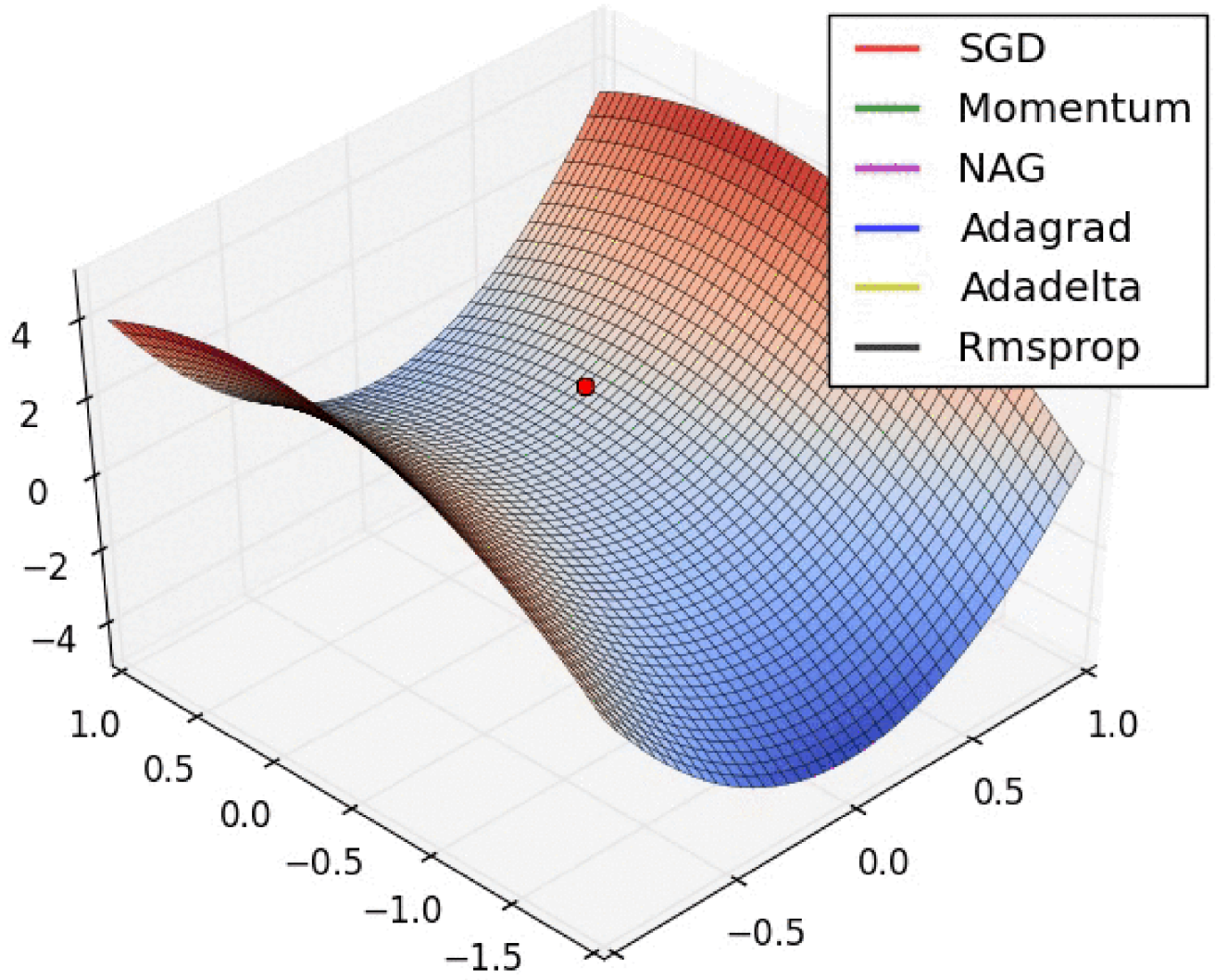
$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$, v_t é a variância dos gradientes até o passo t.

- É o mais utilizado, pois funciona bem na prática em comparação aos outros métodos.
- Leva a uma convergência rápida.
- Default: $\beta_1 = 0.9$ e $\beta_2 = 0.999$.

Otimização



Otimização


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5