

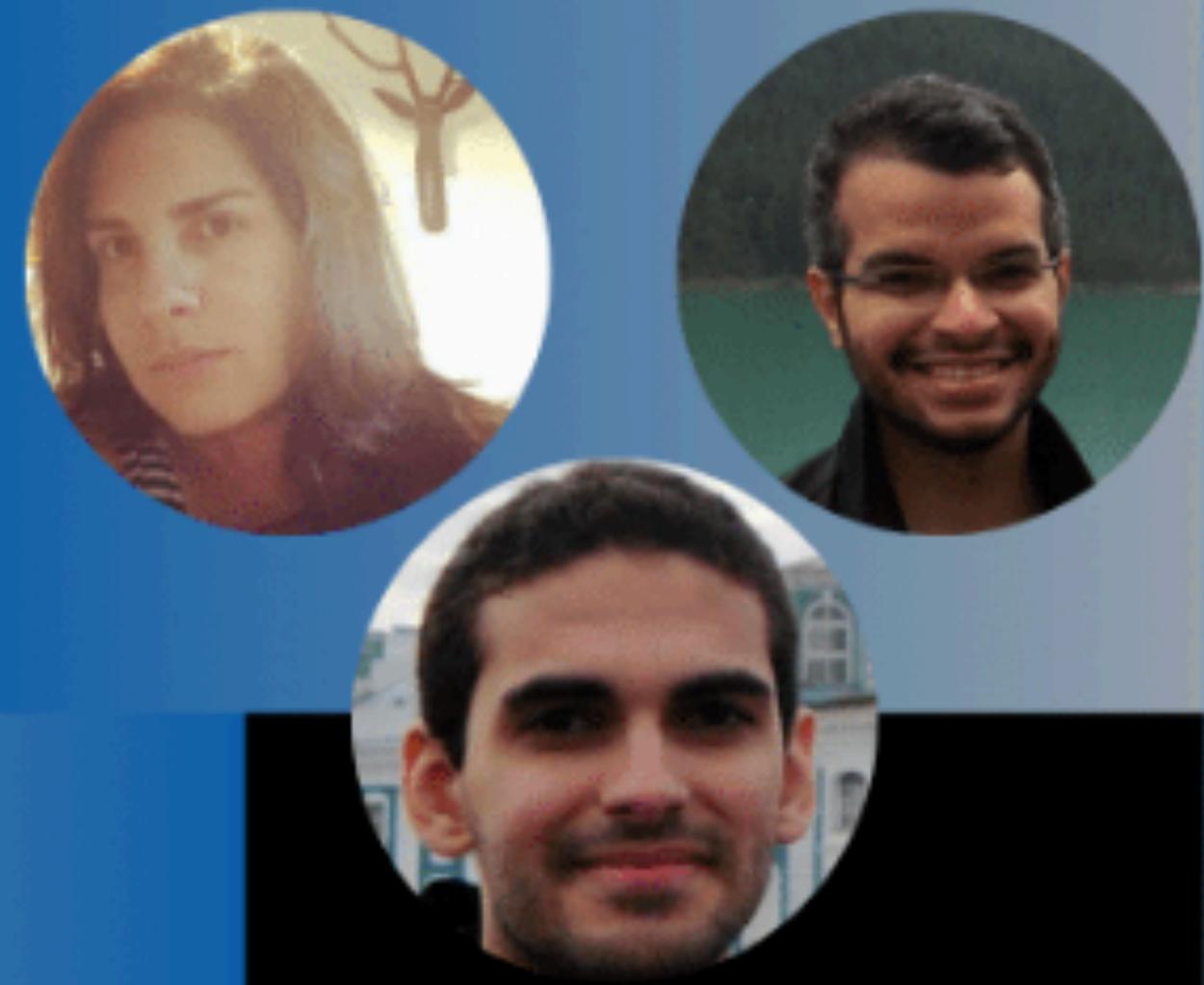
CURSO BÁSICO DE
INTELIGÊNCIA
ARTIFICIAL E
BATE-PAPO COM
CONVIDADOS
ESPECIAIS

INTELIGÊNCIA ARTIFICIAL PARA TODOS

DE 08/06 A 12/08



COM OS PROFESSORES DO
LABORATÓRIO ARIA/UFPB:
TELMO FILHO, THAÍS
GAUDENCIO E YURI
MALHEIROS



- CURSO SEM PRÉ-REQUISITOS
- [HTTP://ARIA.CI.UFPB.BR/IAPARATODOS](http://ARIA.CI.UFPB.BR/IAPARATODOS)
- INSCRIÇÃO PARA CERTIFICADO - DE 01/06 A 07/06: [HTTP://BIT.LY/SIGEVENTOS](http://BIT.LY/SIGEVENTOS)
- ENCONTROS: SEGUNDAS E QUARTAS
- HORÁRIO: 19:00 ÀS 20:00





[Início](#) [Sobre](#) [Projetos](#) [Membros](#) [Parceiros](#) [Publicações](#) [Contato](#)



LABORATÓRIO DE APLICAÇÕES EM INTELIGÊNCIA ARTIFICIAL

As experiências definem a aprendizagem. Assim, o ARIA constrói experiências para máquinas e para pessoas, formando especialistas na área de inteligência artificial e ciência de dados, desenvolvendo aplicações e pesquisando seus métodos.

[SAIBA MAIS](#)

aria.ci.ufpb.br

SE INSCREVE E JÁ APERTA NO SININHO, QUE VOCÊS PASSAM A RECEBER AS NOTIFICAÇÕES.

**NOSSOS ENCONTROS DURARÃO 1 HORA E, ASSIM QUE POSSÍVEL, DEIXAREMOS OS VÍDEOS
GRAVADOS NO CANAL.**

**NÃO PRECISA SE PREOCUPAR EM ESTAR LIGADO ÀS 19:00, MAS ESTANDO, ROLA TIRAR
DÚVIDA E PARTICIPAR, O QUE JÁ DEIXA A AULA MAIS ANIMADA.**

**SOBRE O MATERIAL DE ACOMPANHAMENTO: O ALUNO PRECISA SE LOGAR EM:
CLASSROOM.GOOGLE.COM**

DEPOIS, CLICAR EM PARTICIPAR DA TURMA (ÍCONE COM UM MAIS +)

POR FIM, ENTRAR COM O CÓDIGO DA TURMA: PXV3ANW

TAMBÉM EM: [HTTPS://ARIA.CI.UFPB.BR/IA-PARA-TODOS-MATERIAL/](https://aria.ci.ufpb.br/ia-para-todos-material/)
**ESPERAMOS QUE VOCÊS REALMENTE CURTAM O CURSO E APROVEITEM-NO AO MÁXIMO. NÃO
DEIXEM DE INTERAGIR CONOSCO, TAMBÉM, POR E-MAIL OU MENSAGEM NO NOSSO
INSTAGRAM (@APRENDIZAGEMDEMAQUINA)**

Redes Neurais

Yuri Malheiros

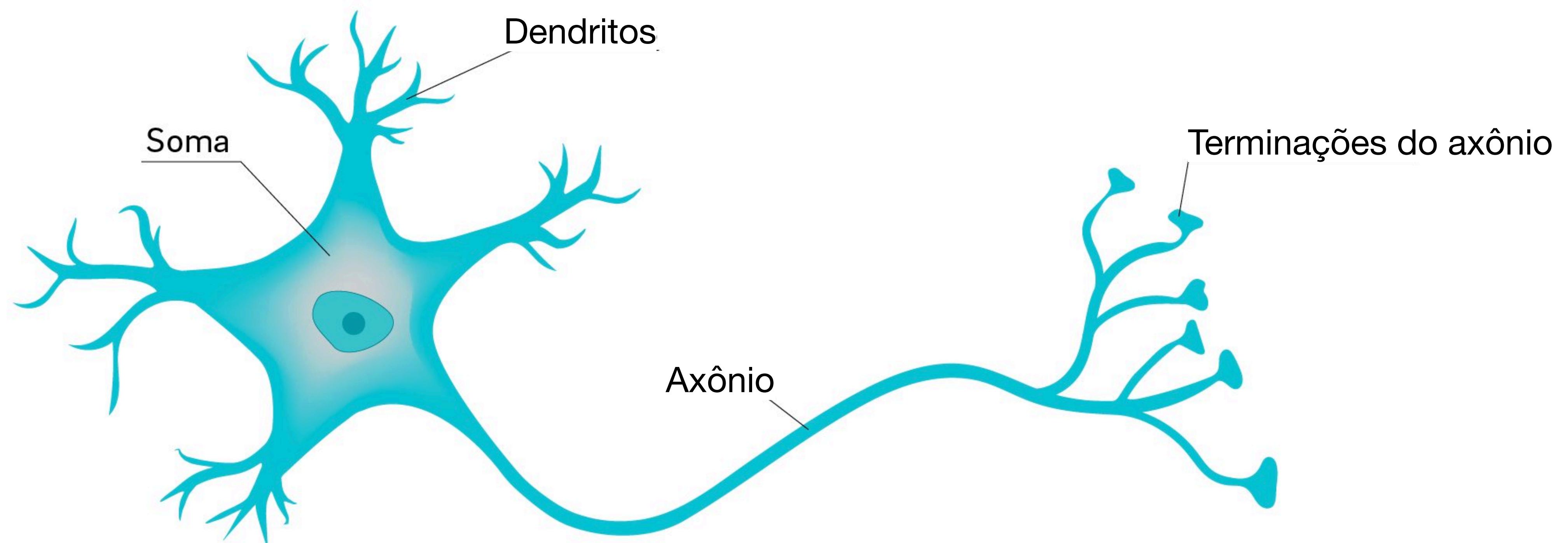
Redes Neurais

O cérebro é o principal órgão associado à inteligência e aprendizagem

Na busca pela construção de máquinas inteligentes era natural que o cérebro surgisse como um modelo a ser seguido

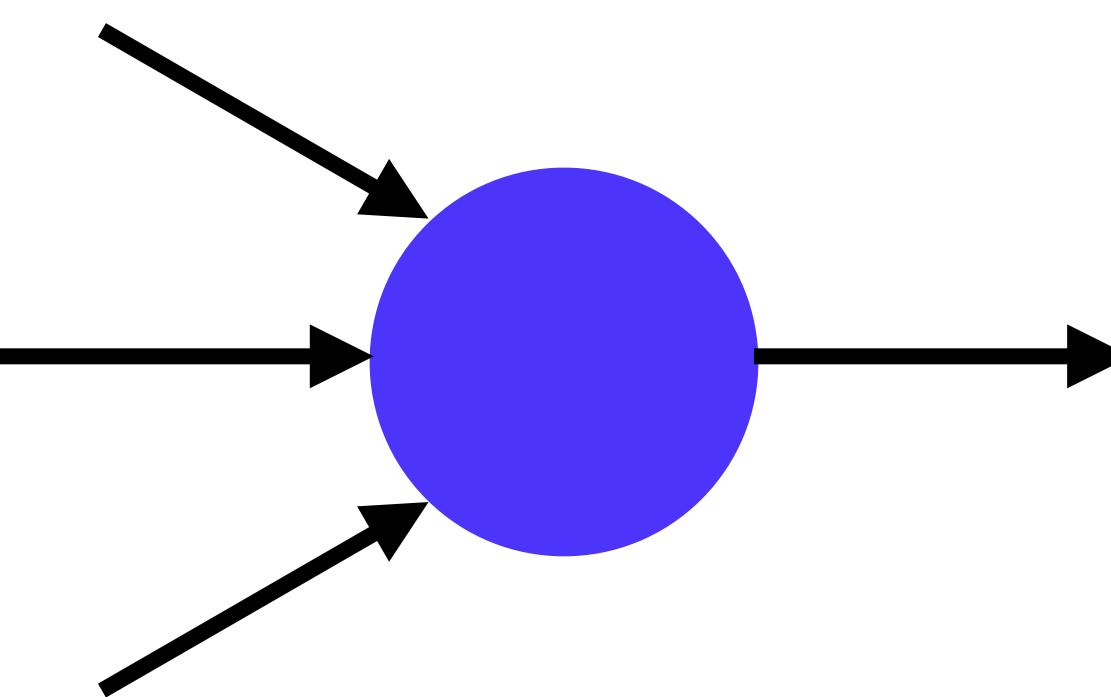
O cérebro é composto por uma rede complexa de aproximadamente 100 bilhões de neurônios interconectados

Redes Neurais

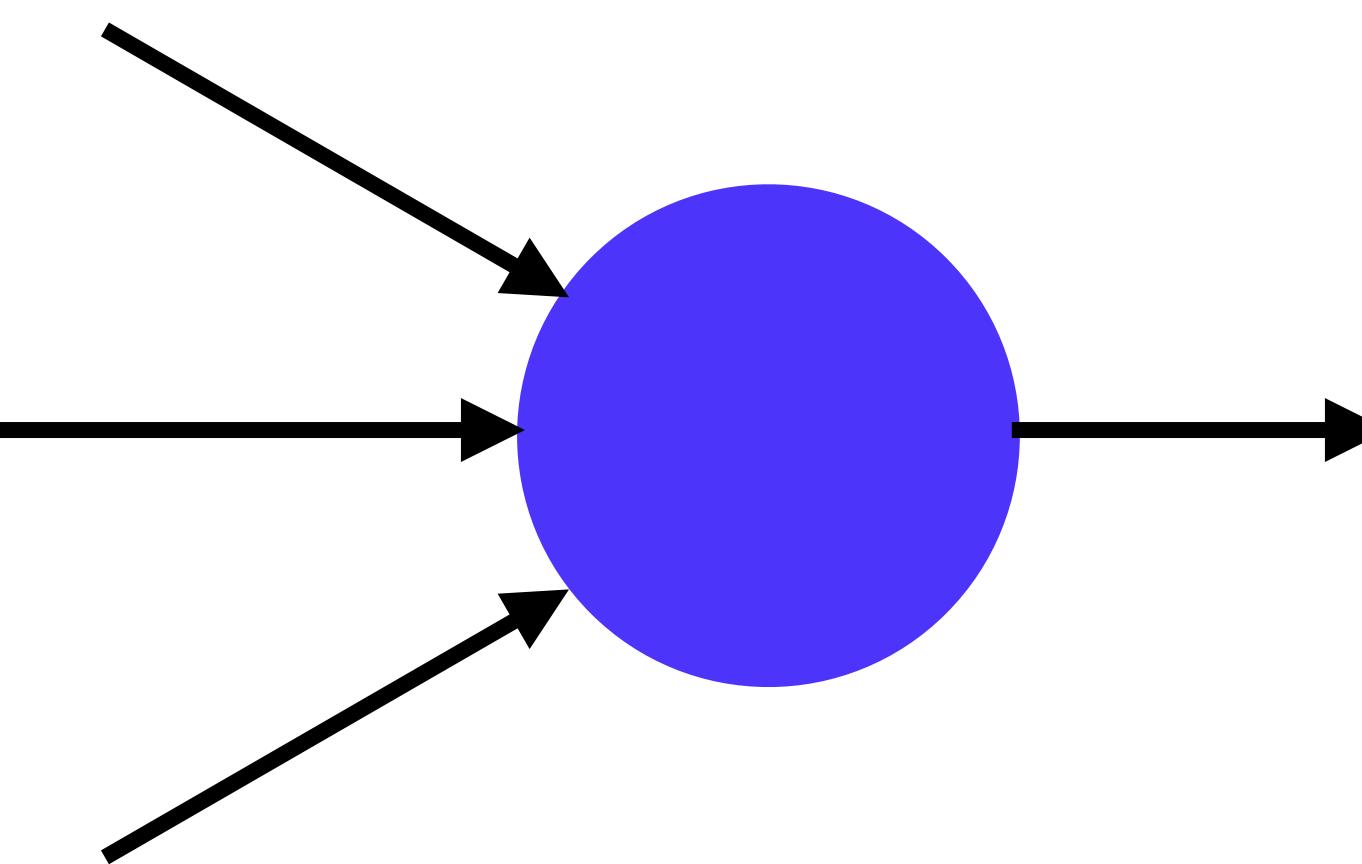


Redes Neurais

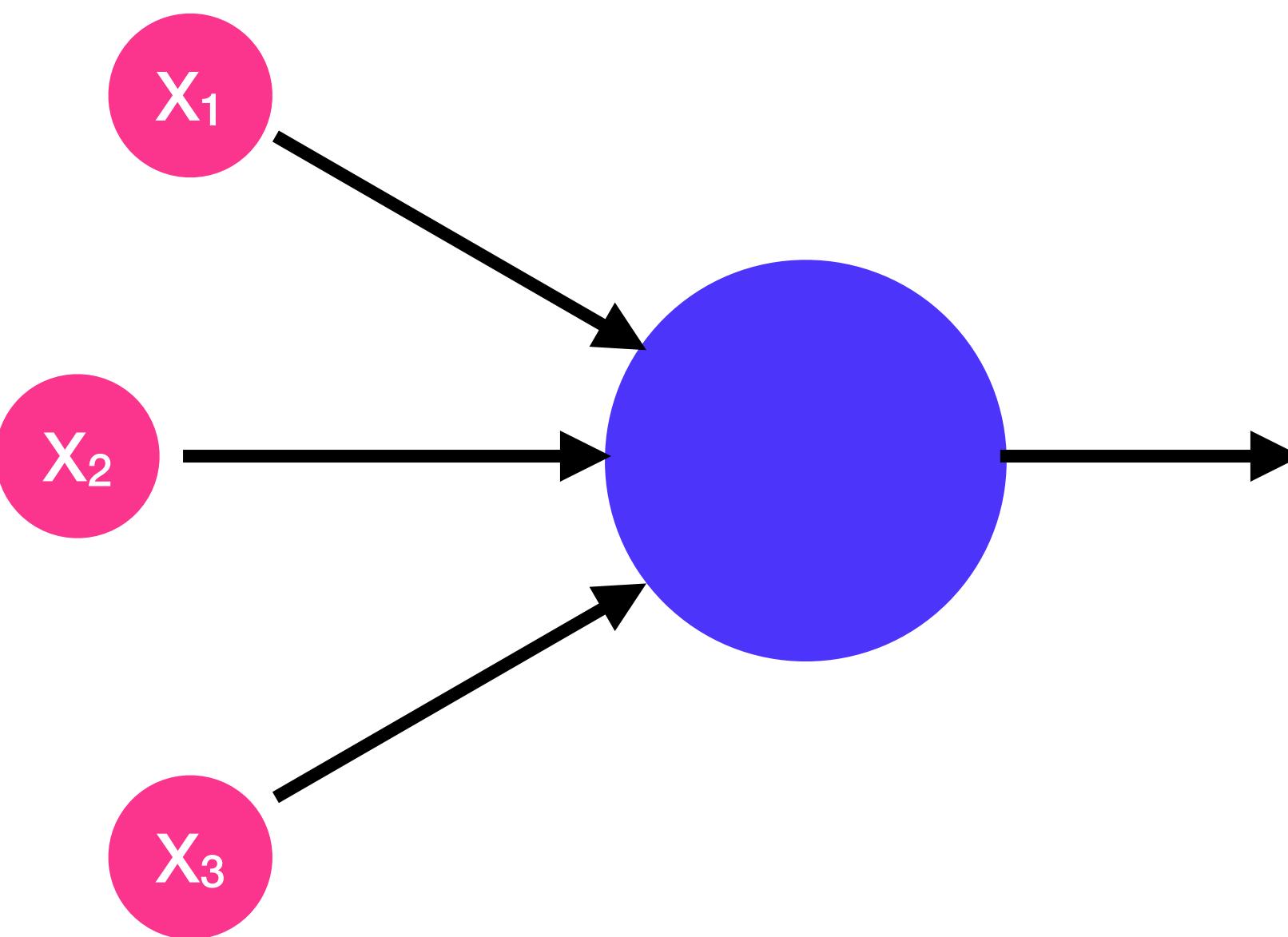
Em 1943, McCulloch e Pitts propuseram um modelo de neurônio artificial: **perceptron**



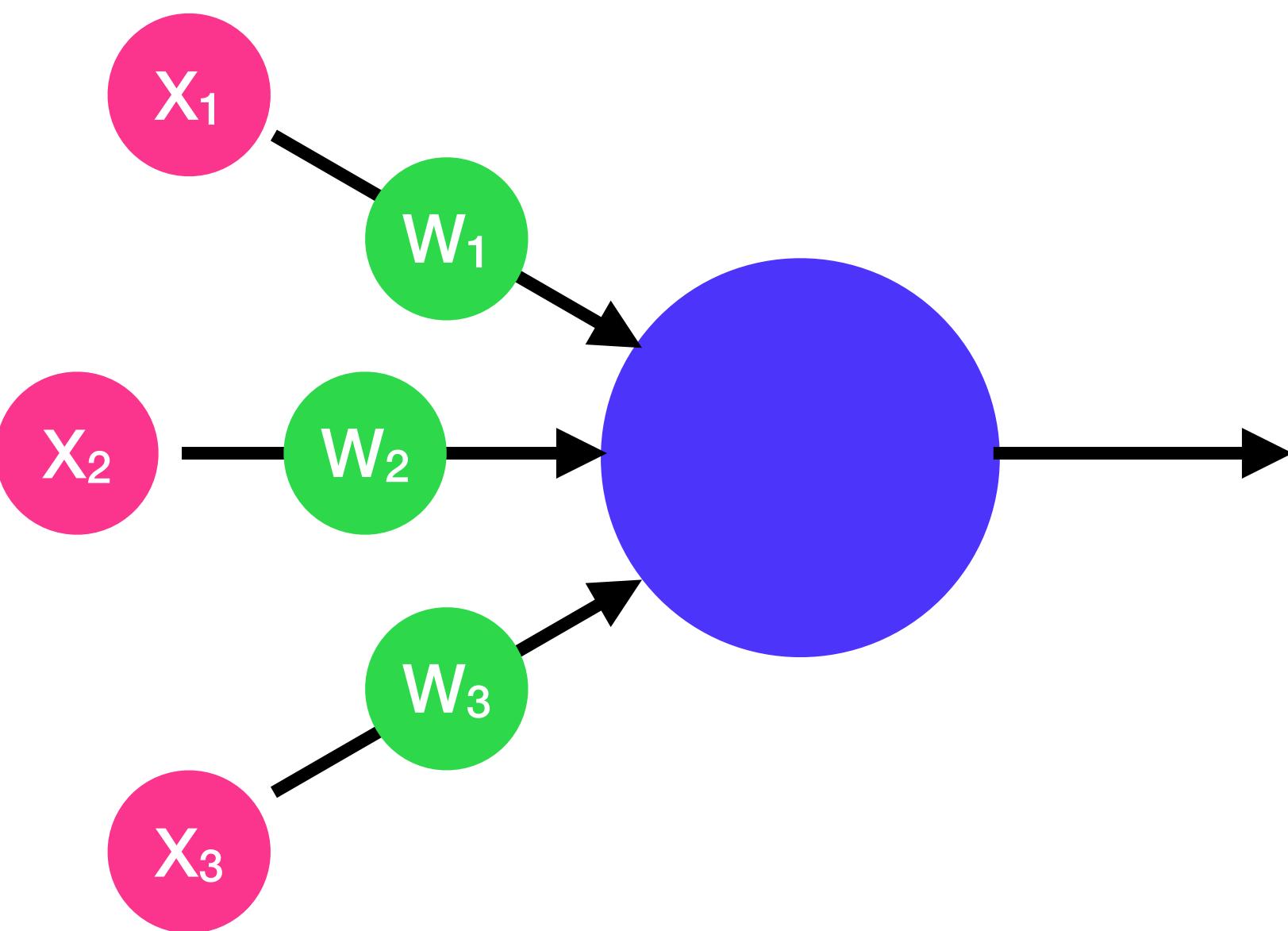
Perceptron



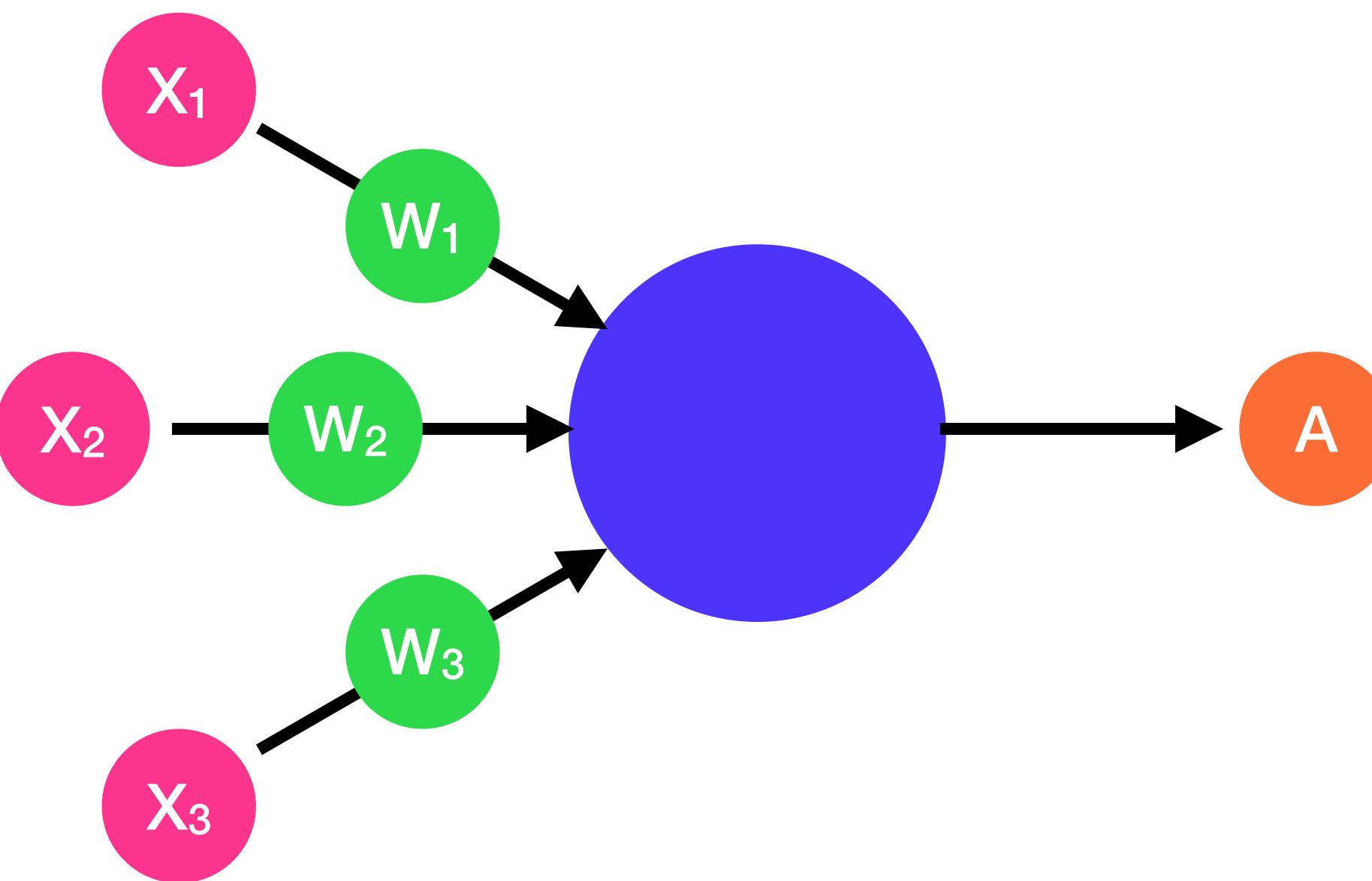
Perceptron



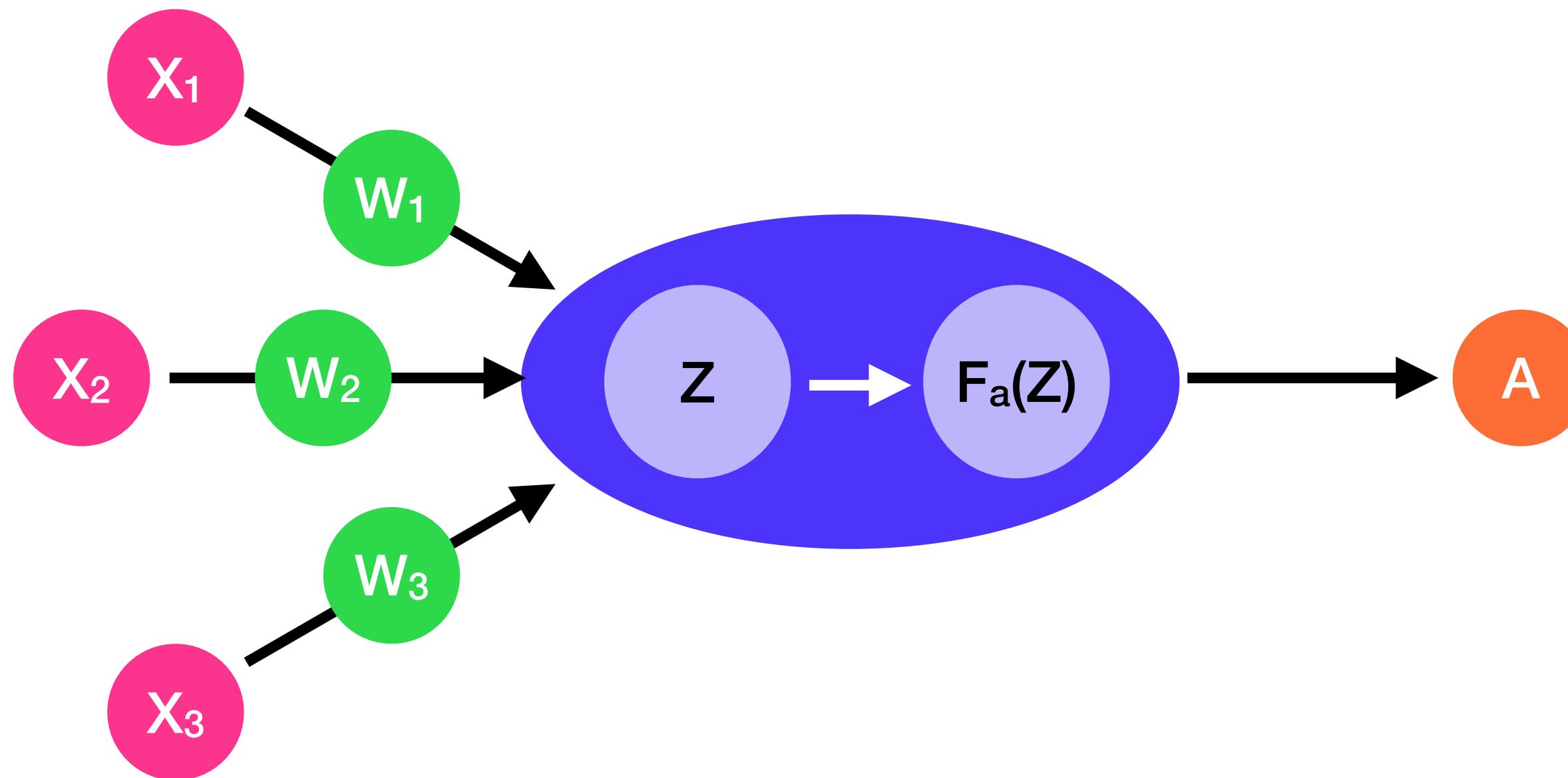
Perceptron



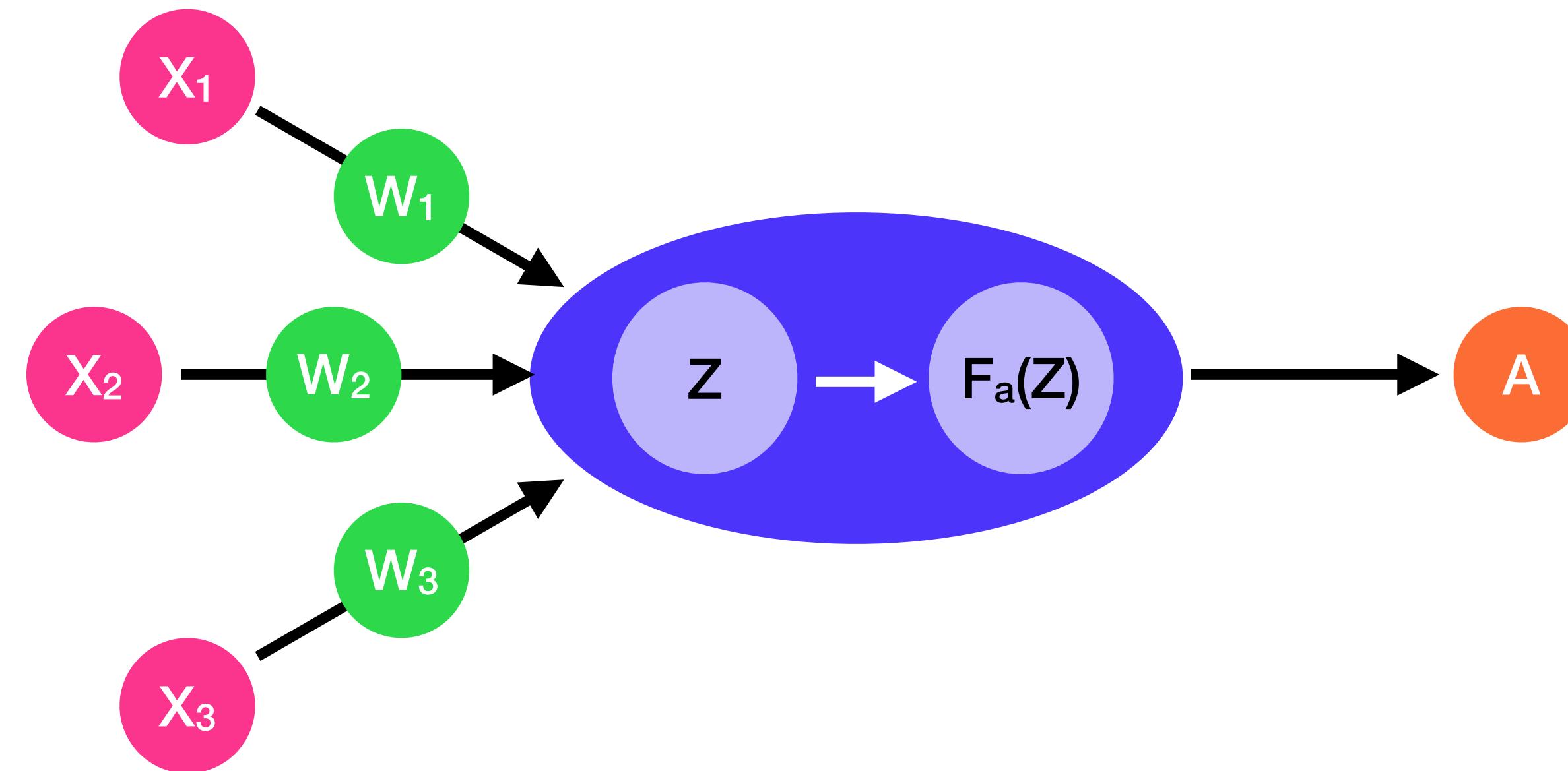
Perceptron



Perceptron

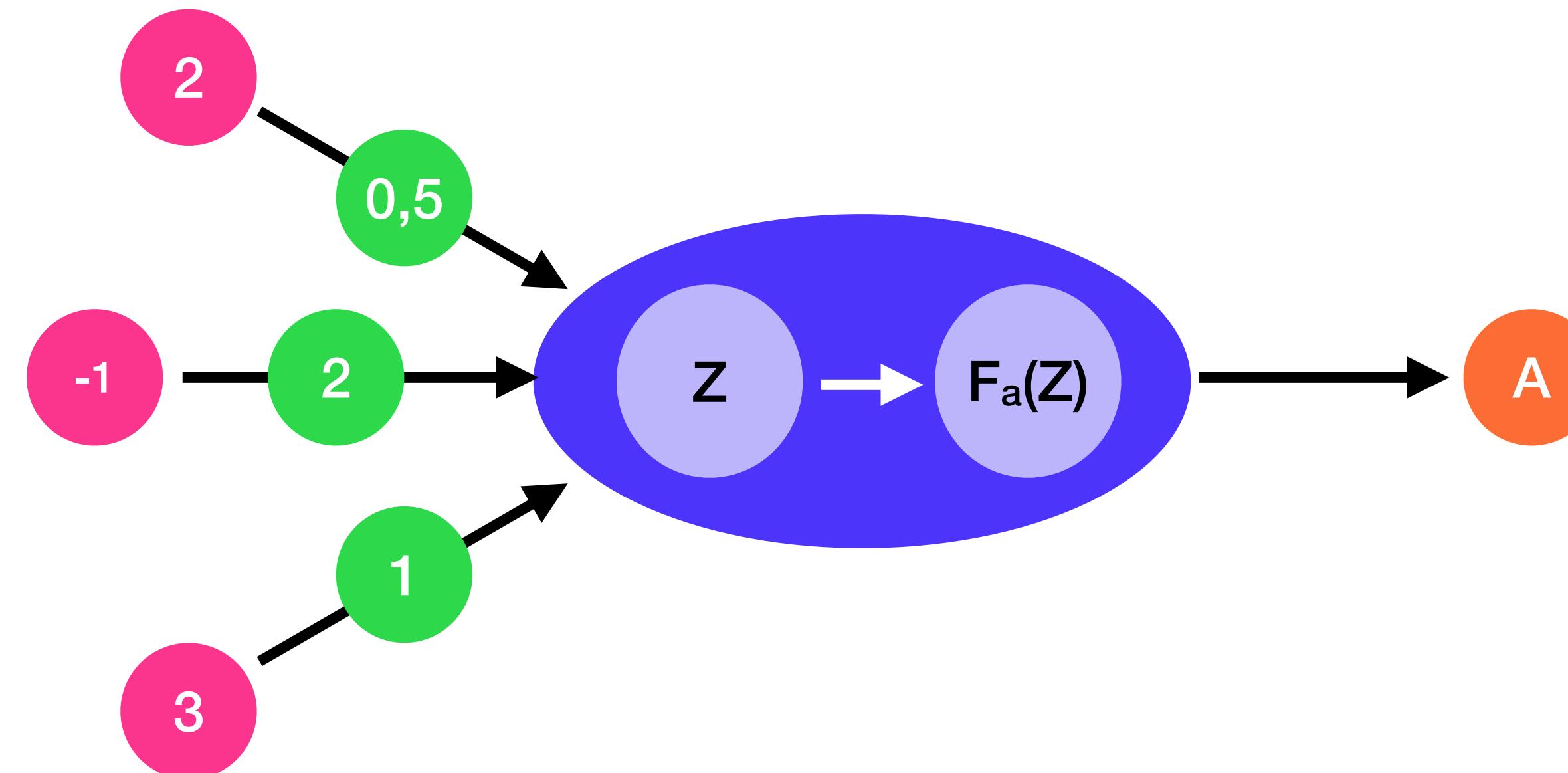


Perceptron



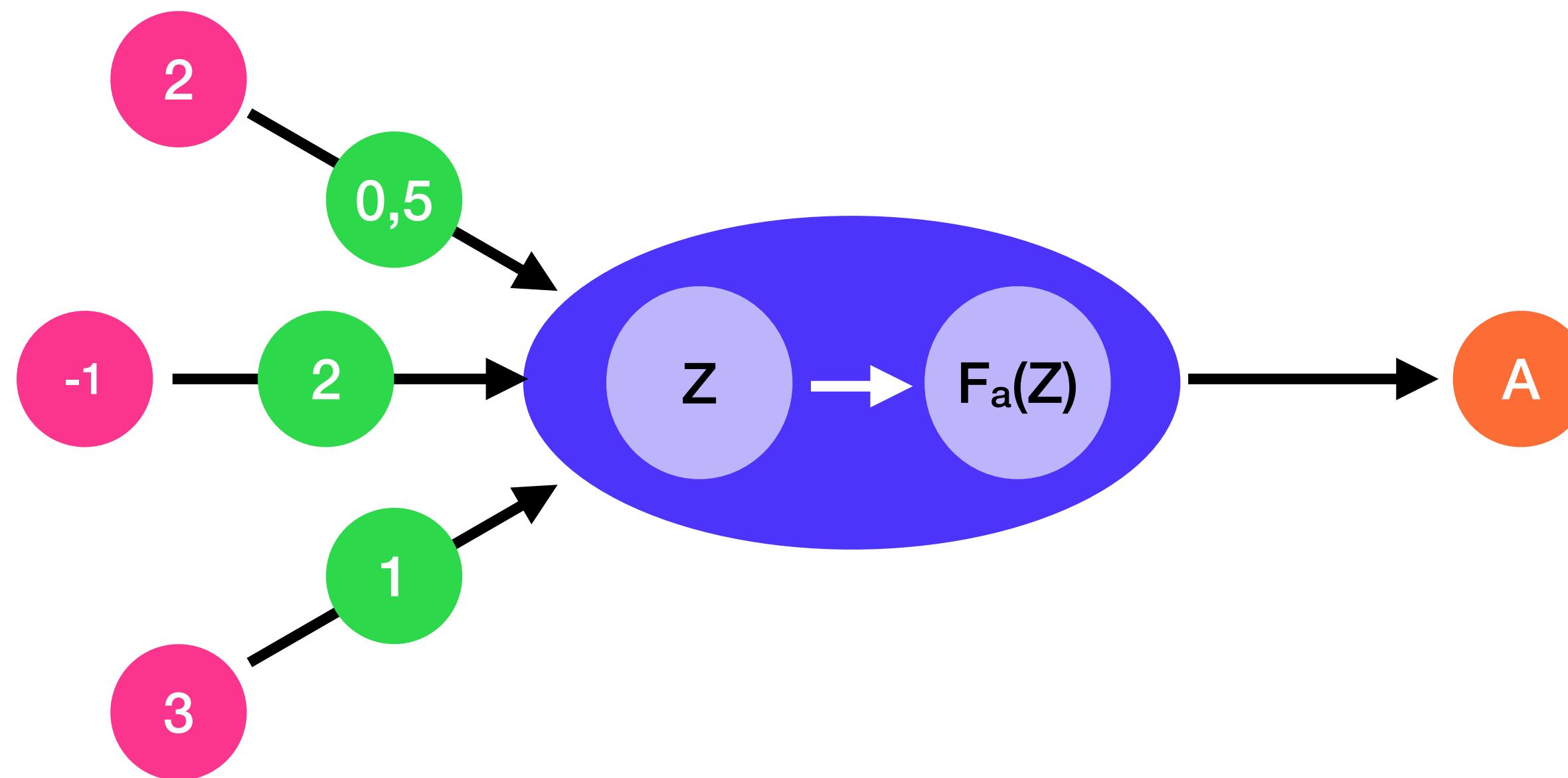
$$Z = W_1 \times X_1 + W_2 \times X_2 + W_3 \times X_3$$

Perceptron



$$Z = 0,5 \times 2 + 2 \times -1 + 1 \times 3 = 2$$

Perceptron



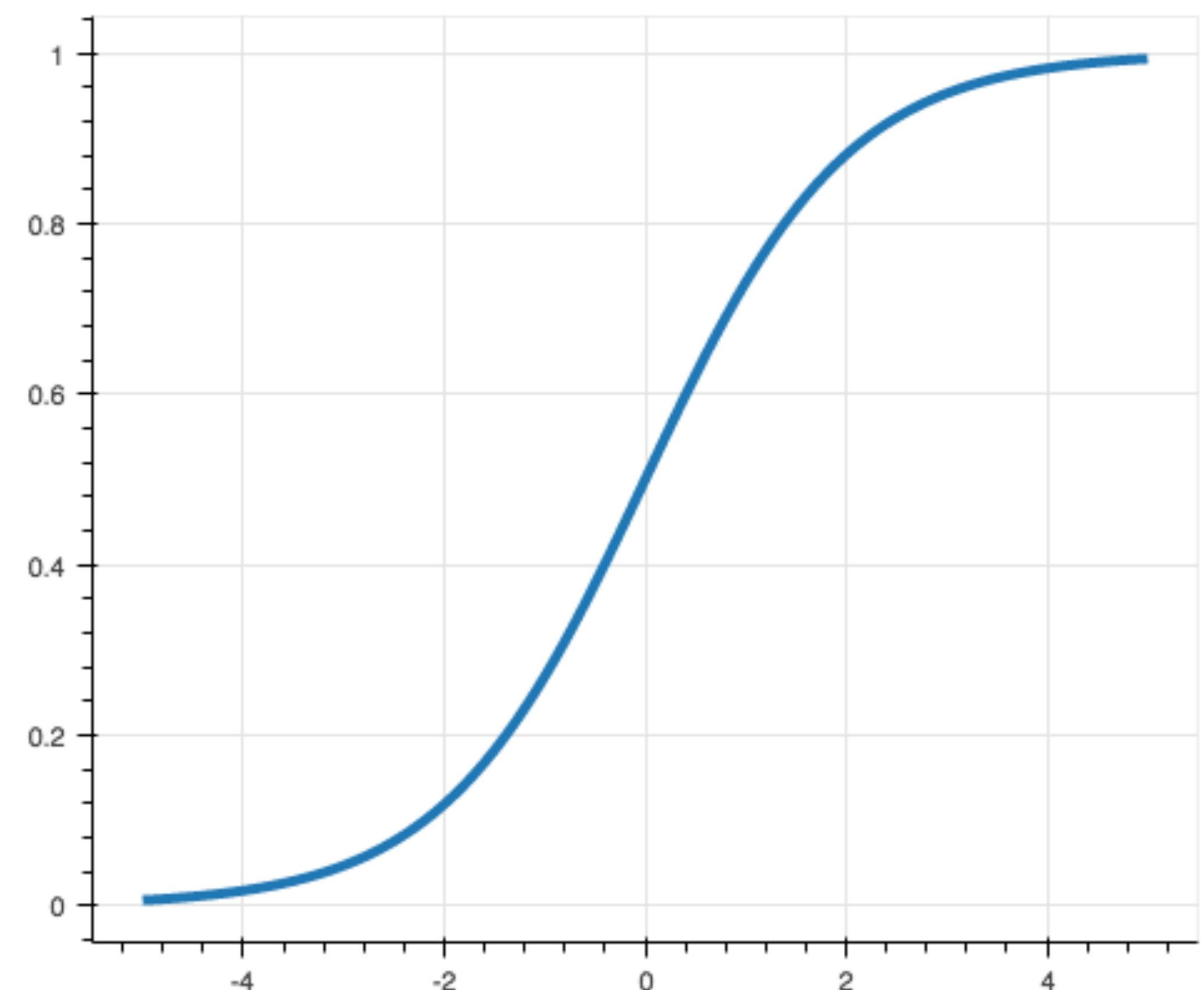
$$A = F_a(2)$$

Função de Ativação

Várias funções podem ser usadas como função de ativação (F_a)

Função Sísmoide:

$$f_a(z) = \frac{1}{1 + e^{-z}}$$

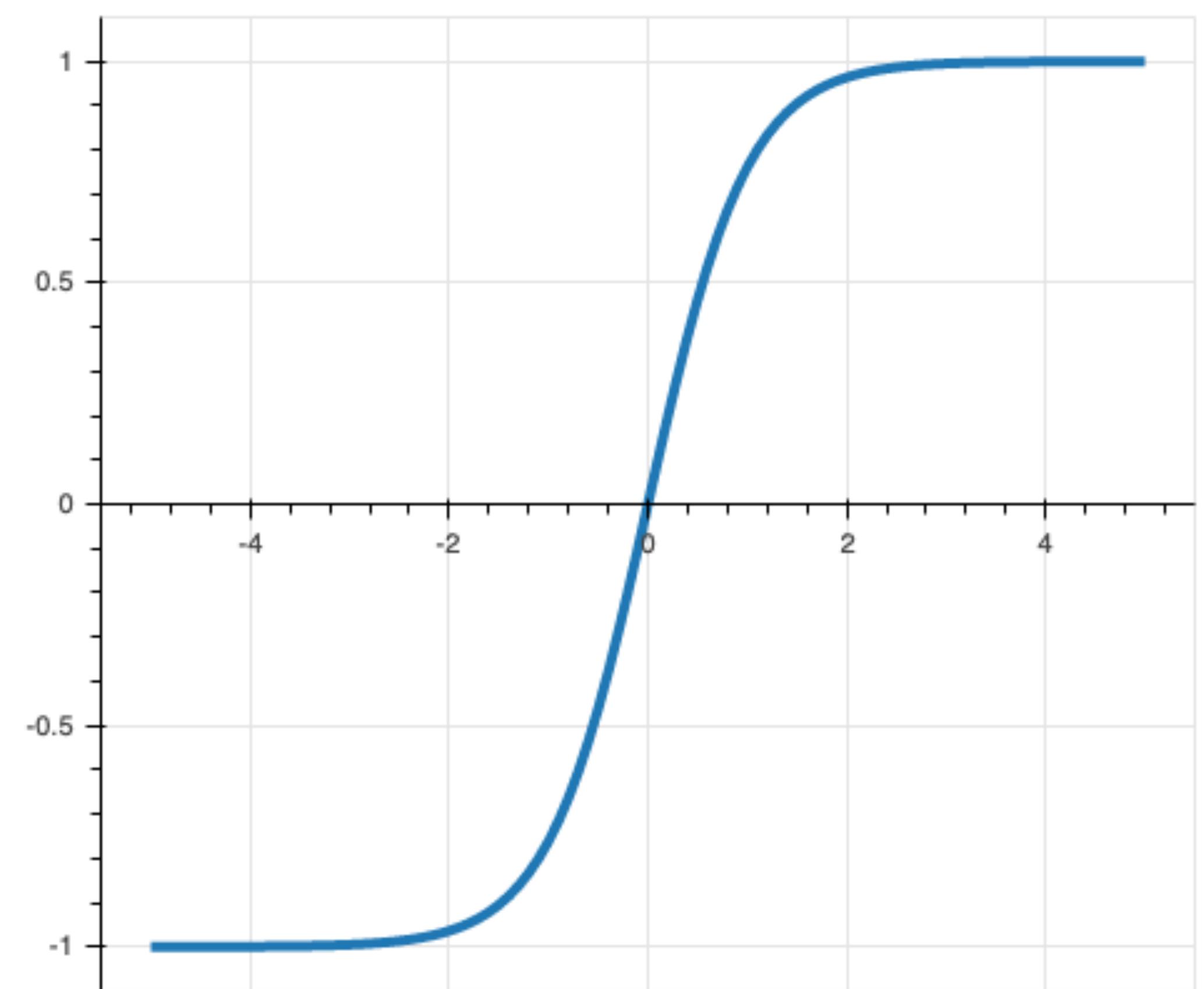


Função de Ativação

Várias funções podem ser usadas como função de ativação (F_a)

Função Tangente Hiperbólica:

$$f_a(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

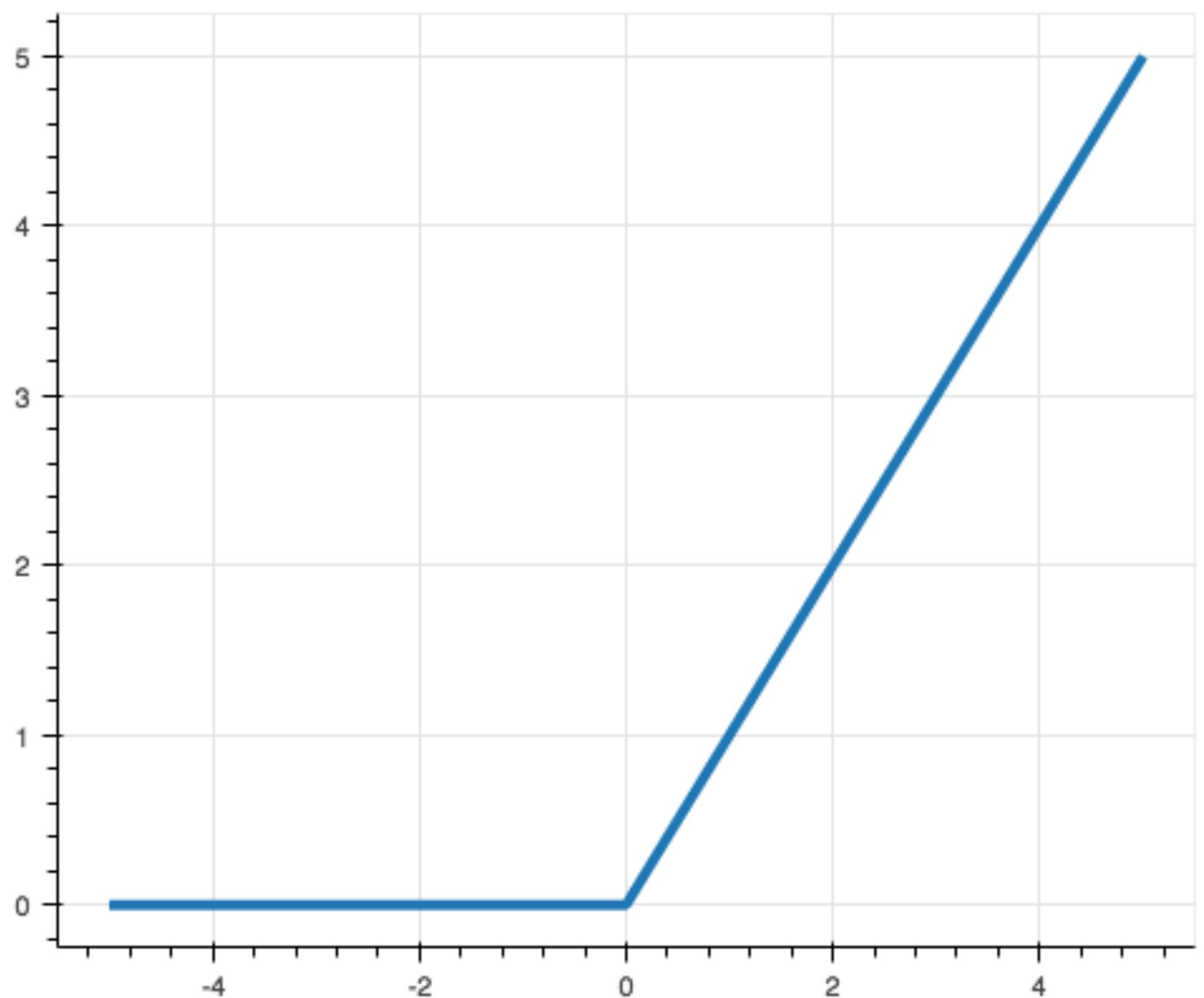


Função de Ativação

Várias funções podem ser usadas como função de ativação (F_a)

Função ReLU:

$$f_a(z) = \max(0, z)$$



Redes Neurais

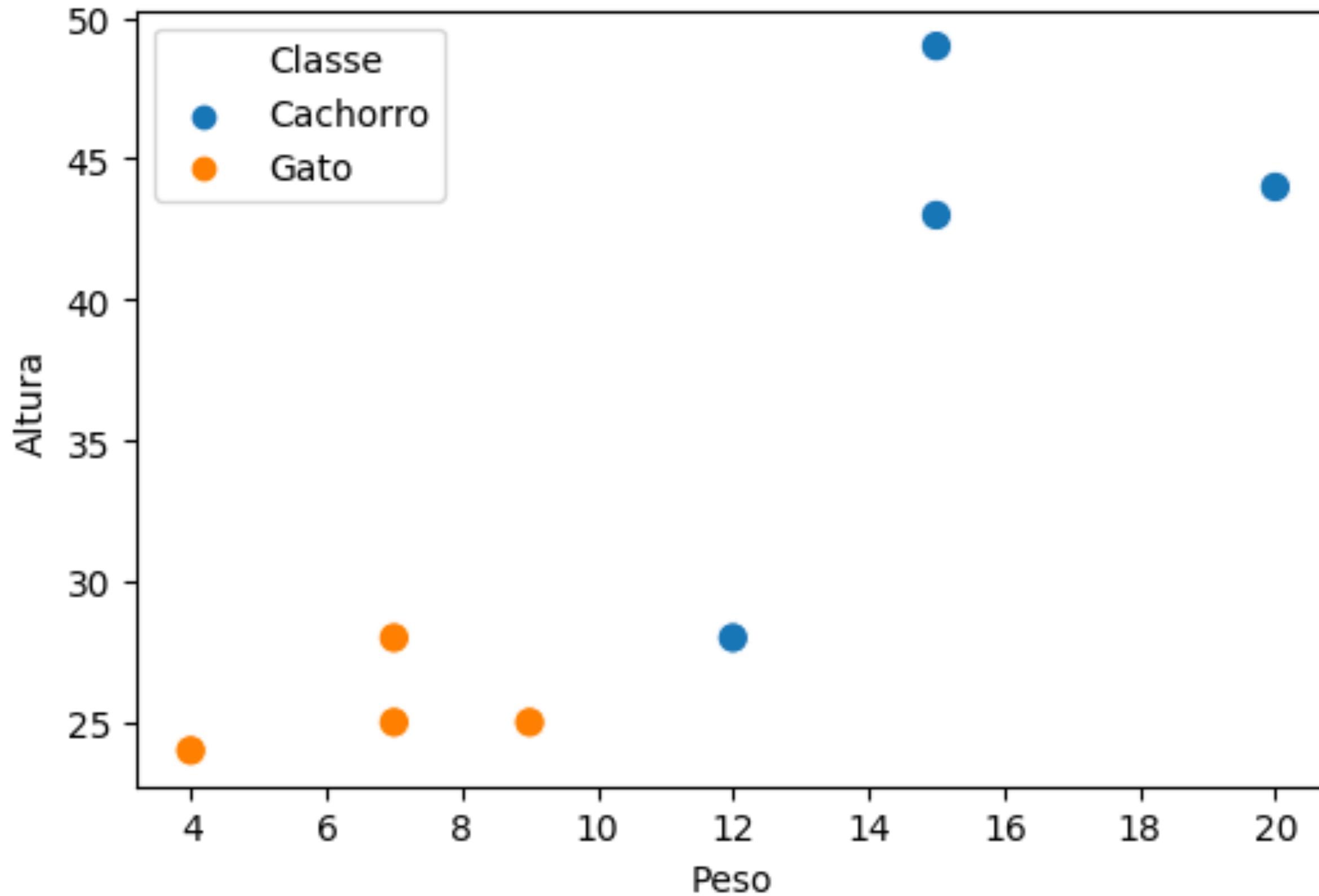
Podemos usar o neurônio artificial para diferentes problemas

Ele nos fornece uma saída de acordo com as entradas

O resultado do processamento das entradas para fornecer uma saída é determinado pela função de ativação e pelos pesos

Exemplo

Gatos e cachorros:



	Peso	Altura	Classe
0	20	44	Cachorro
1	15	43	Cachorro
2	12	28	Cachorro
3	15	49	Cachorro
4	7	28	Gato
5	7	25	Gato
6	9	25	Gato
7	4	24	Gato

Exemplo

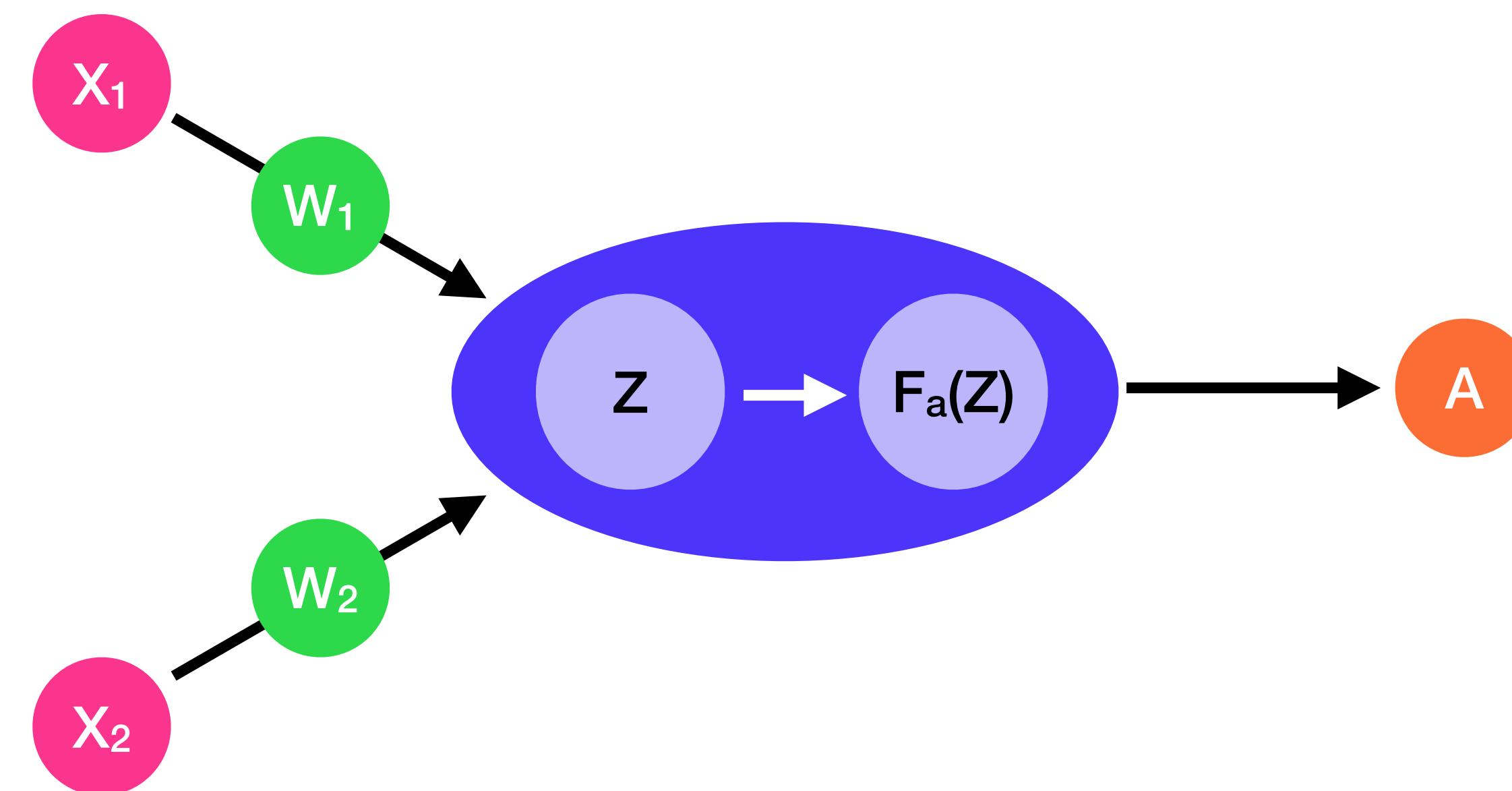
Dada a altura e o peso, vamos fazer o neurônio ter:

Saída **0** se for um gato

Saída **1** se for um cachorro

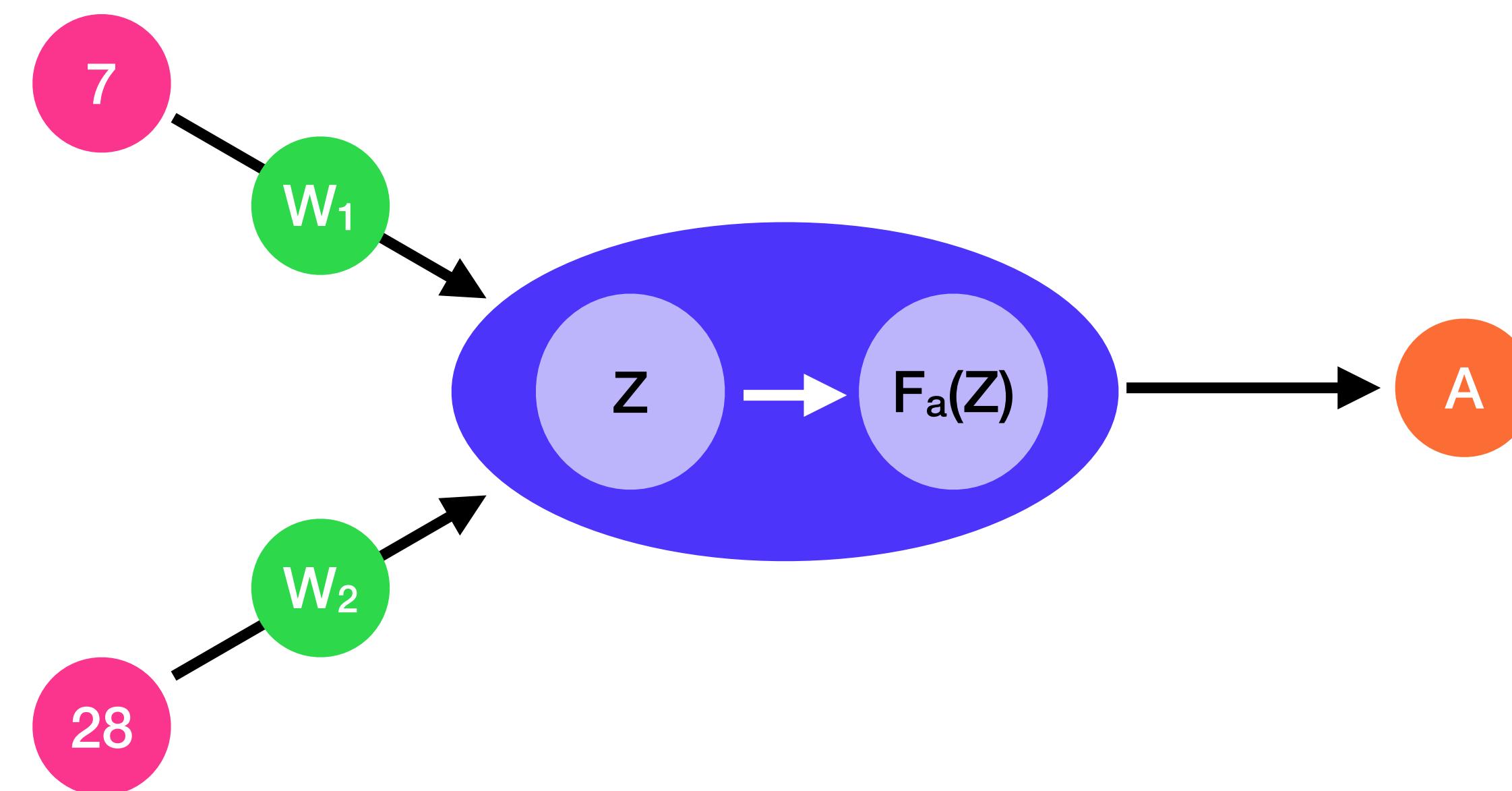
Exemplo

X_1 Peso X_2 Altura



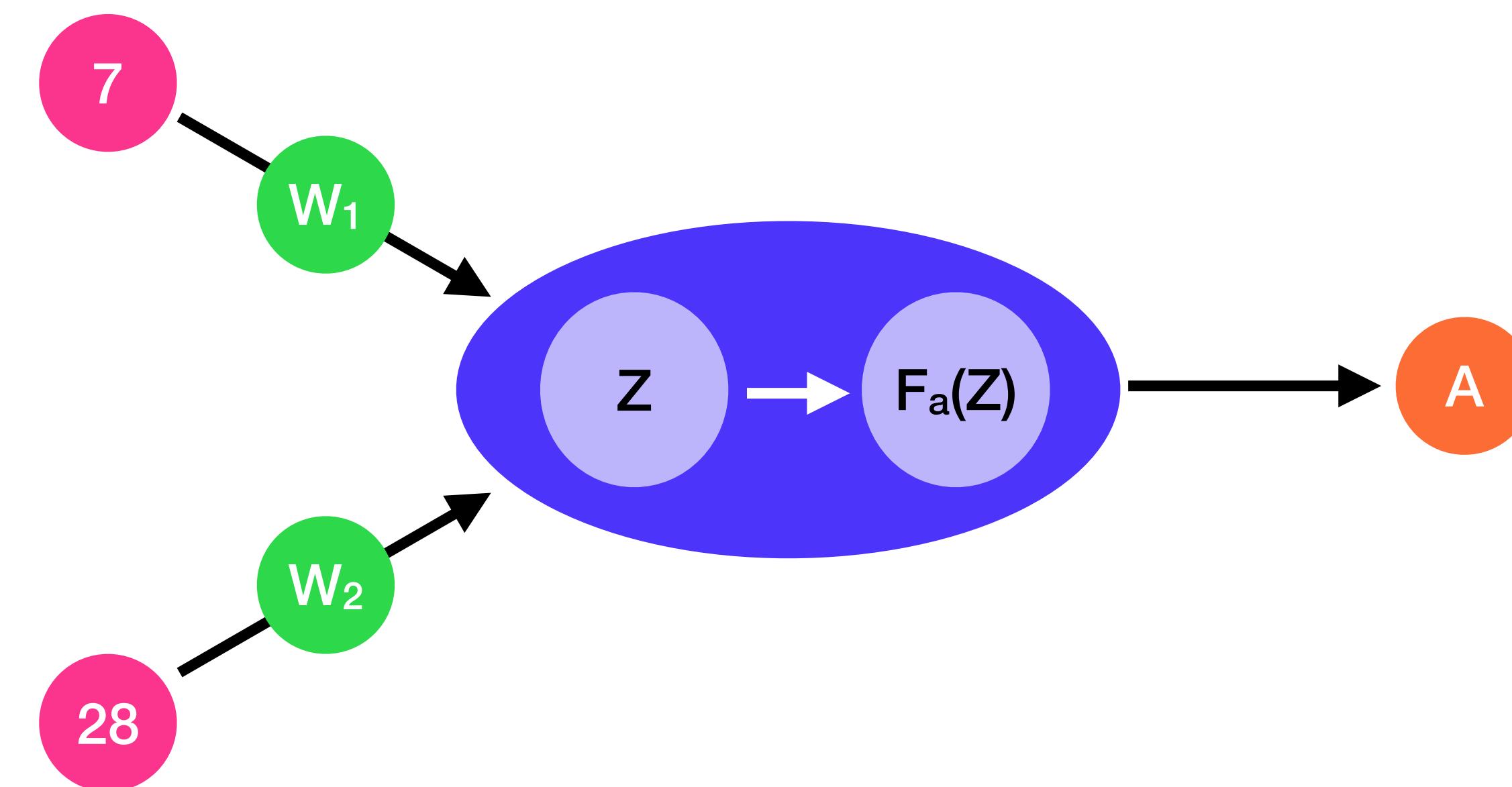
Exemplo

X_1 Peso X_2 Altura



Exemplo

X_1 Peso X_2 Altura



$$z = w_1 \times 7 + w_2 \times 28$$

Exemplo

Quais são os valores de w ?

Podemos tentar encontrá-los manualmente, porém este é um processo complexo e muitas vezes inviável

Treinando uma Rede Neural

Como não sabemos os pesos iniciais, vamos iniciá-los aleatoriamente

$$w_1 = 0,25$$

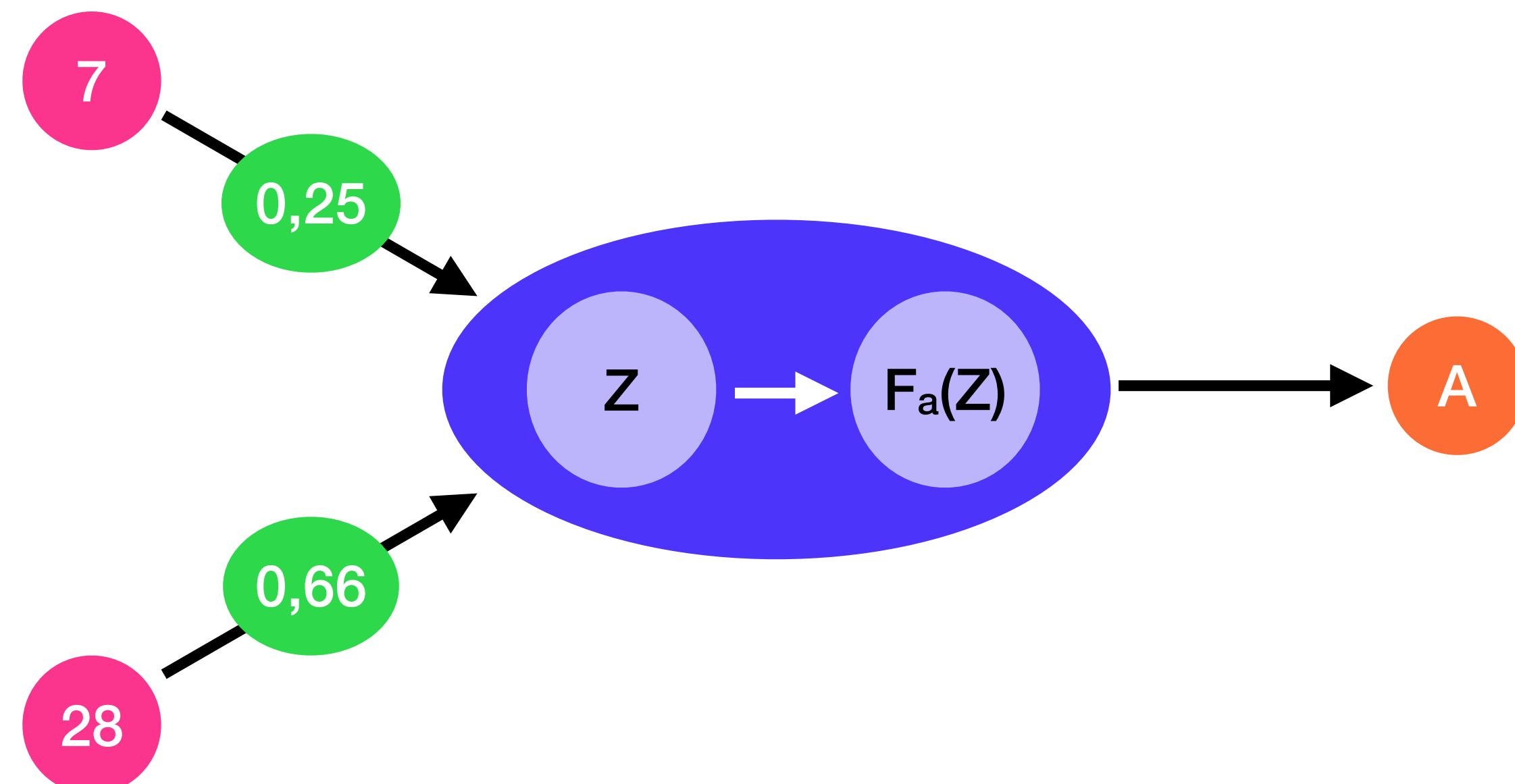
$$w_2 = 0,66$$

Em seguida, vamos calcular a saída para

$$x_1 = 7$$

$$x_2 = 28$$

Treinando uma Rede Neural



$$z = 0,25 \times 7 + 0,66 \times 28 = 20,23$$

$$A = F_a(20,23) = 0,99$$

Treinando uma Rede Neural

A saída correta para a classe gato é 0, mas a rede retornou algo muito próximo de 1

Precisamos ajustar o peso para que esse erro não aconteça ou pelo menos que ele diminua

Treinando uma Rede Neural

Esperado	Obtido	Ajuste
0	0	-
0	1	Diminuir W
1	0	Aumentar W
1	1	-

Treinando uma Rede Neural

Esperado	Obtido	Ajuste	Esperado - Obtido
0	0	-	0
0	1	Diminuir W	-1
1	0	Aumentar W	1
1	1	-	0

Treinando uma Rede Neural

Para isso, eu preciso saber o erro, ou seja a diferença entre o valor esperado e o valor obtido

Quanto maior o erro, maior precisa ser o ajuste dos valores de **w**

Então, vamos atualizar **w₁** da seguinte forma:

$$w_1 \leftarrow w_1 + (esperado - obtido)$$

Treinando uma Rede Neural

Para isso, eu preciso saber o erro, ou seja a diferença entre o valor esperado e o valor obtido

Quanto maior o erro, maior precisa ser o ajuste dos valores de **w**

Então, vamos atualizar **w₁** da seguinte forma:

$$w_1 \leftarrow w_1 + (esperado - obtido) x_1$$

Treinando uma Rede Neural

Para isso, eu preciso saber o erro, ou seja a diferença entre o valor esperado e o valor obtido

Quanto maior o erro, maior precisa ser o ajuste dos valores de **w**

Então, vamos atualizar **w₁** da seguinte forma:

$$w_1 \leftarrow w_1 + \eta (esperado - obtido) x_1$$

Treinando uma Rede Neural

Para isso, eu preciso saber o erro, ou seja a diferença entre o valor esperado e o valor obtido

Quanto maior o erro, maior precisa ser o ajuste dos valores de **w**

Então, vamos atualizar **w₁** da seguinte forma:

$$w_1 \leftarrow w_1 + \eta (esperado - obtido) x_1$$

 Taxa de aprendizagem: controla o tamanho da atualização

Treinando uma Rede Neural

Cada ajuste vai melhorando um pouco a rede neural

Vamos ajustar os pesos para cada um dos exemplos dos dados de treinamento

$$w_1 \leftarrow w_1 + \eta(\text{esperado} - \text{obtido})x_1$$

$$w_2 \leftarrow w_2 + \eta(\text{esperado} - \text{obtido})x_2$$

Treinando uma Rede Neural

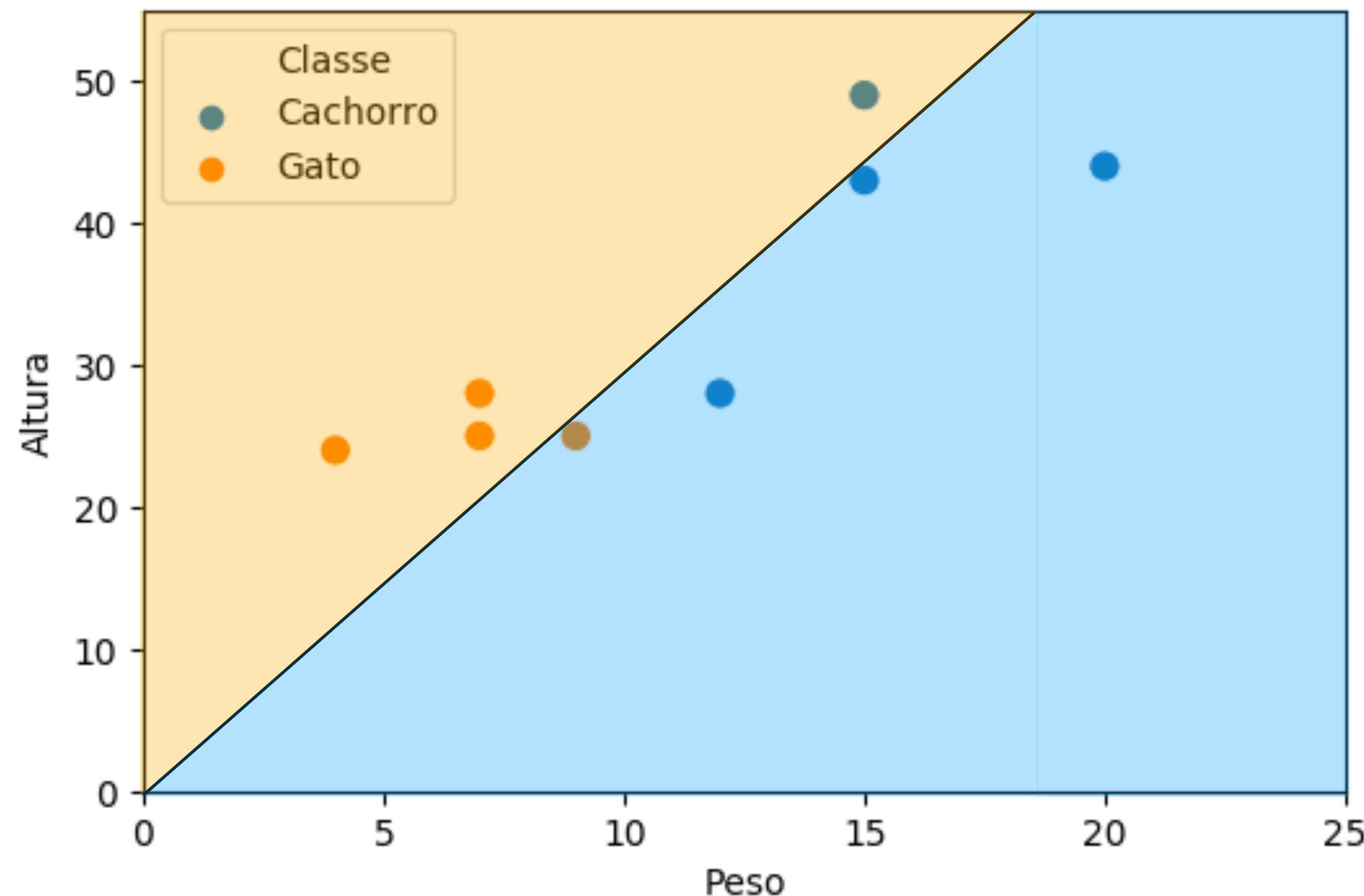
Em muitos casos, fazer uma atualização para cada exemplo pode não ser suficiente

Por isso, costumamos atualizar para cada exemplo e em seguida refazer o processo várias vezes

Esse número de vezes é chamado de **épocas**

Treinando uma Rede Neural

Temos como resultado:

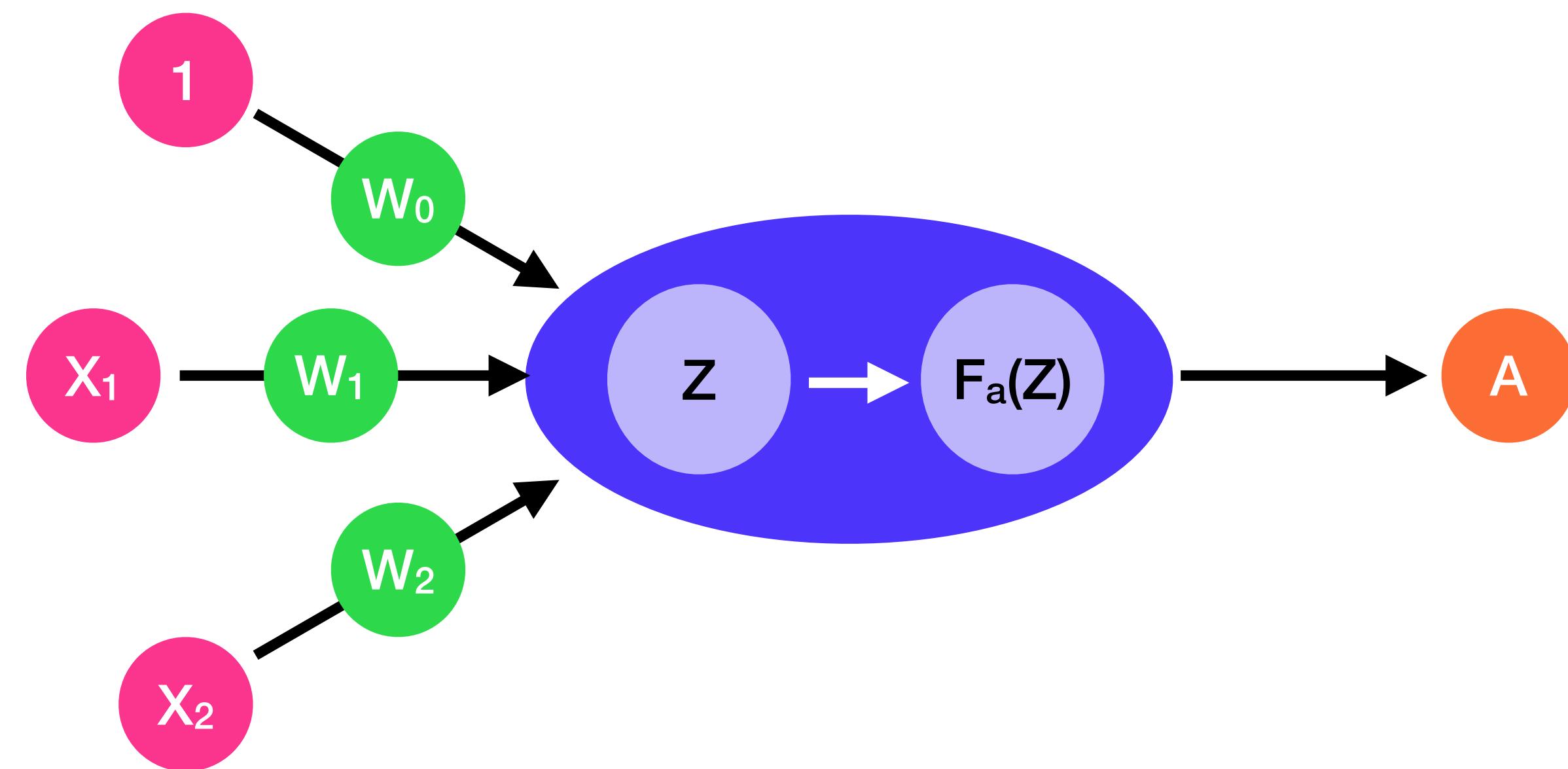


Treinando uma Rede Neural

Usando o neurônio com a estrutura que fizemos, sempre teremos uma fronteira linear partindo da origem

Para permitir que a fronteira possa se deslocar no eixo X, vamos adicionar uma entrada extra no neurônio chamada de **viés**

Treinando uma Rede Neural



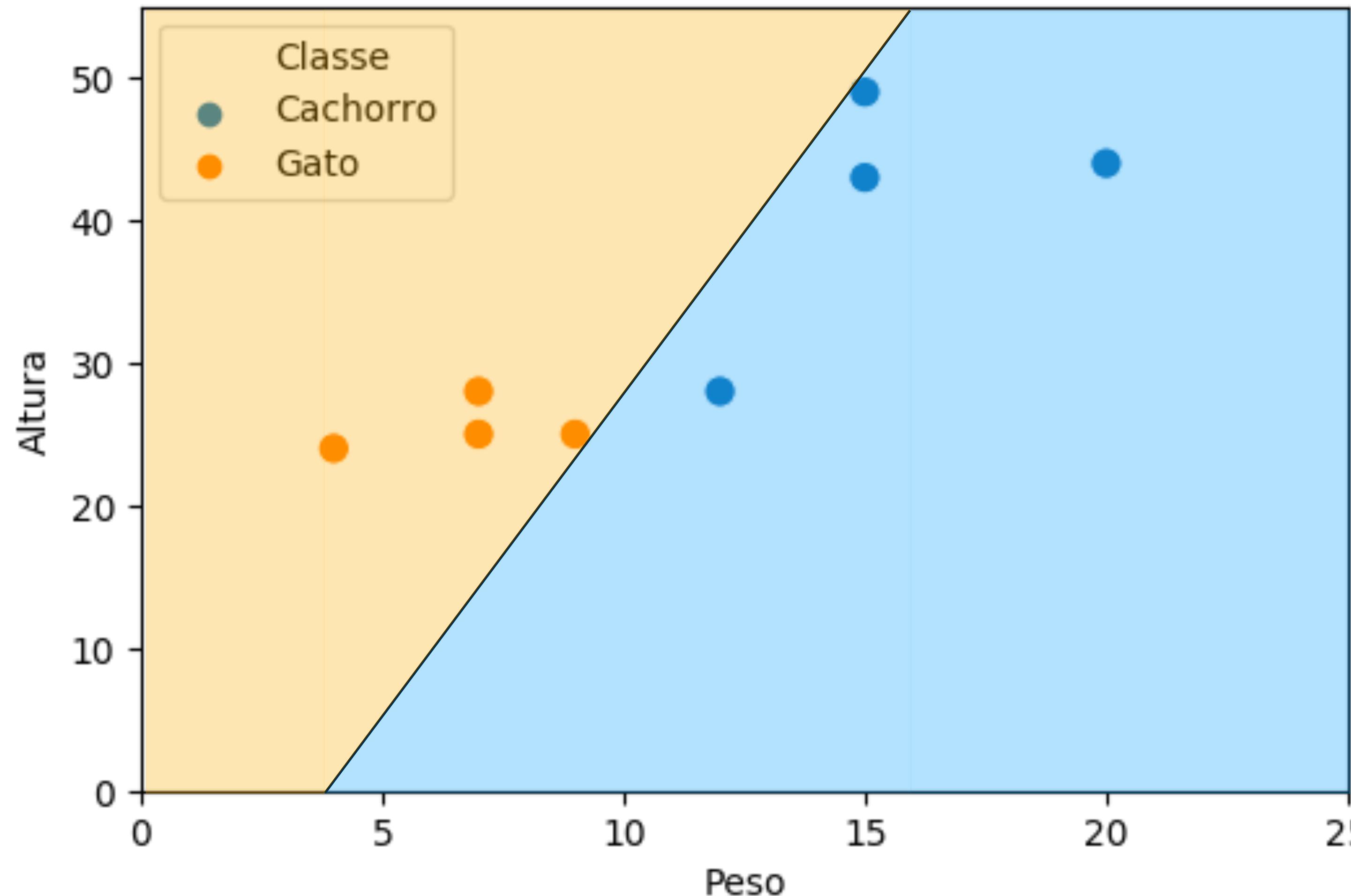
Treinando uma Rede Neural

O viés sempre recebe 1 como entrada

O peso dessa entrada segue o comportamento das outras

Realizando o treinamento dessa forma, temos o seguinte resultado

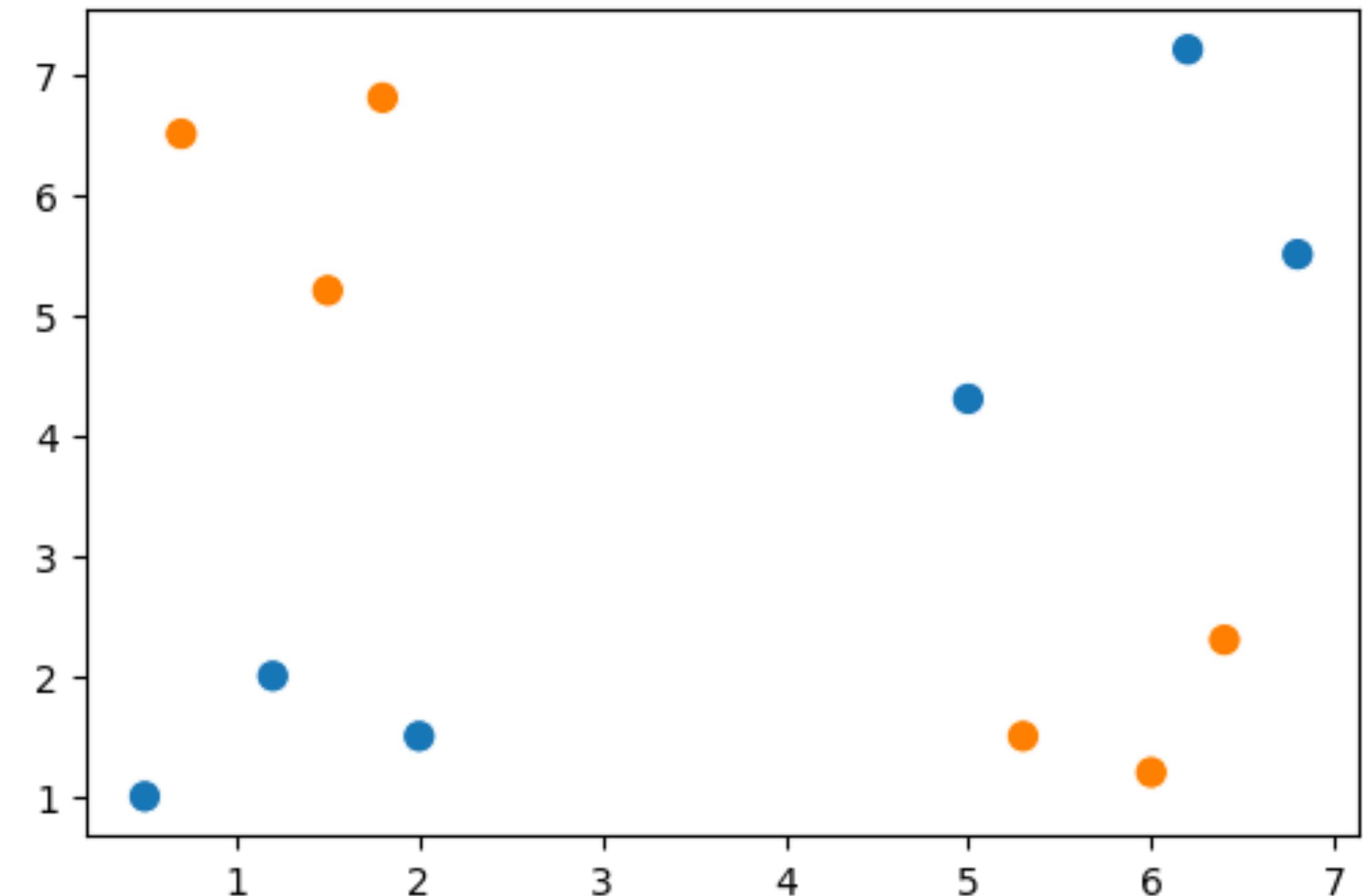
Treinando uma Rede Neural



Perceptron

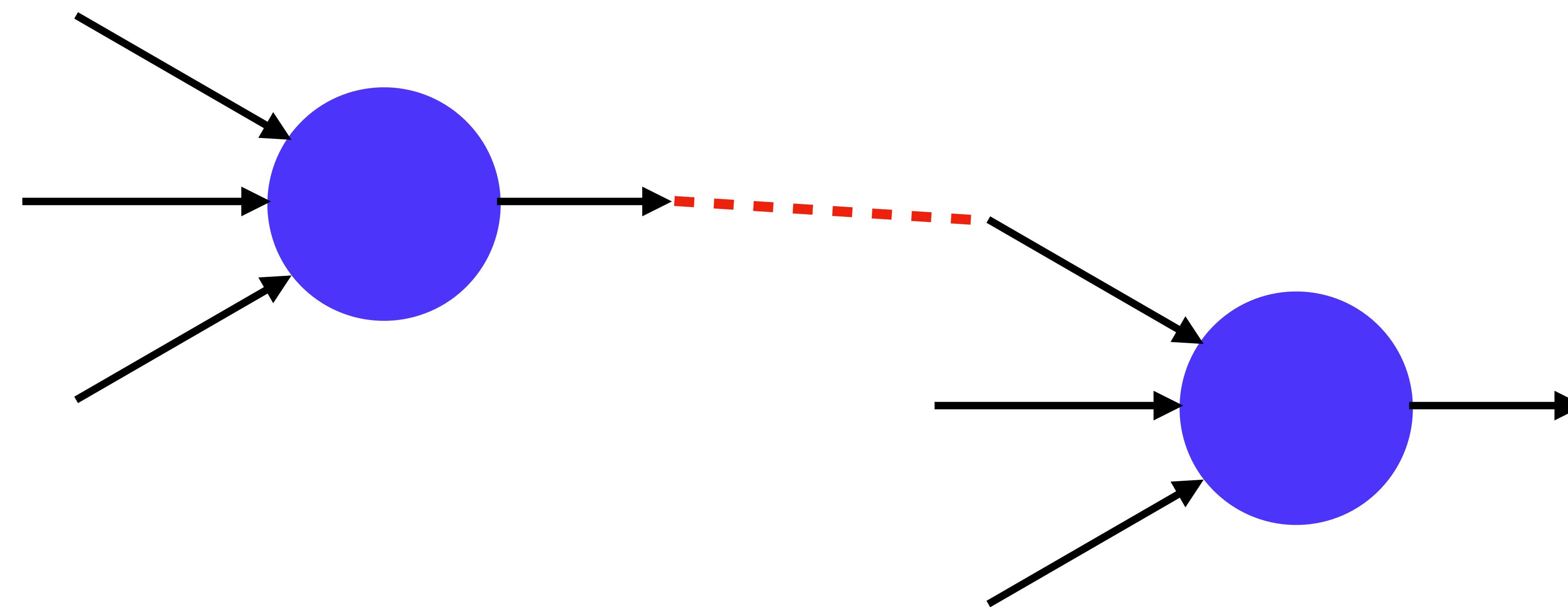
Como vimos, o perceptron possui a limitação de criar apenas fronteiras lineares

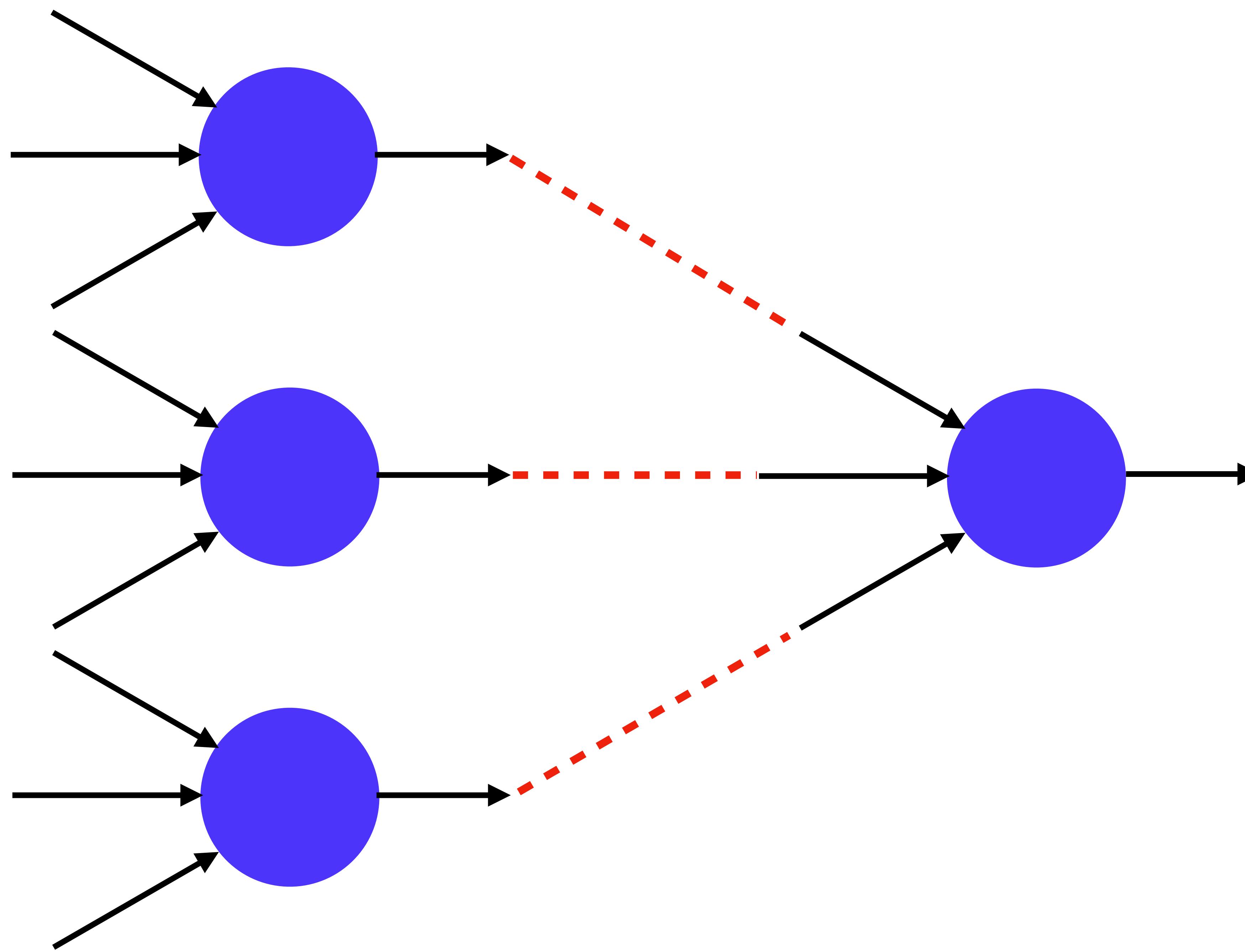
Ele não consegue classificar bem um conjunto de dados dessa forma:

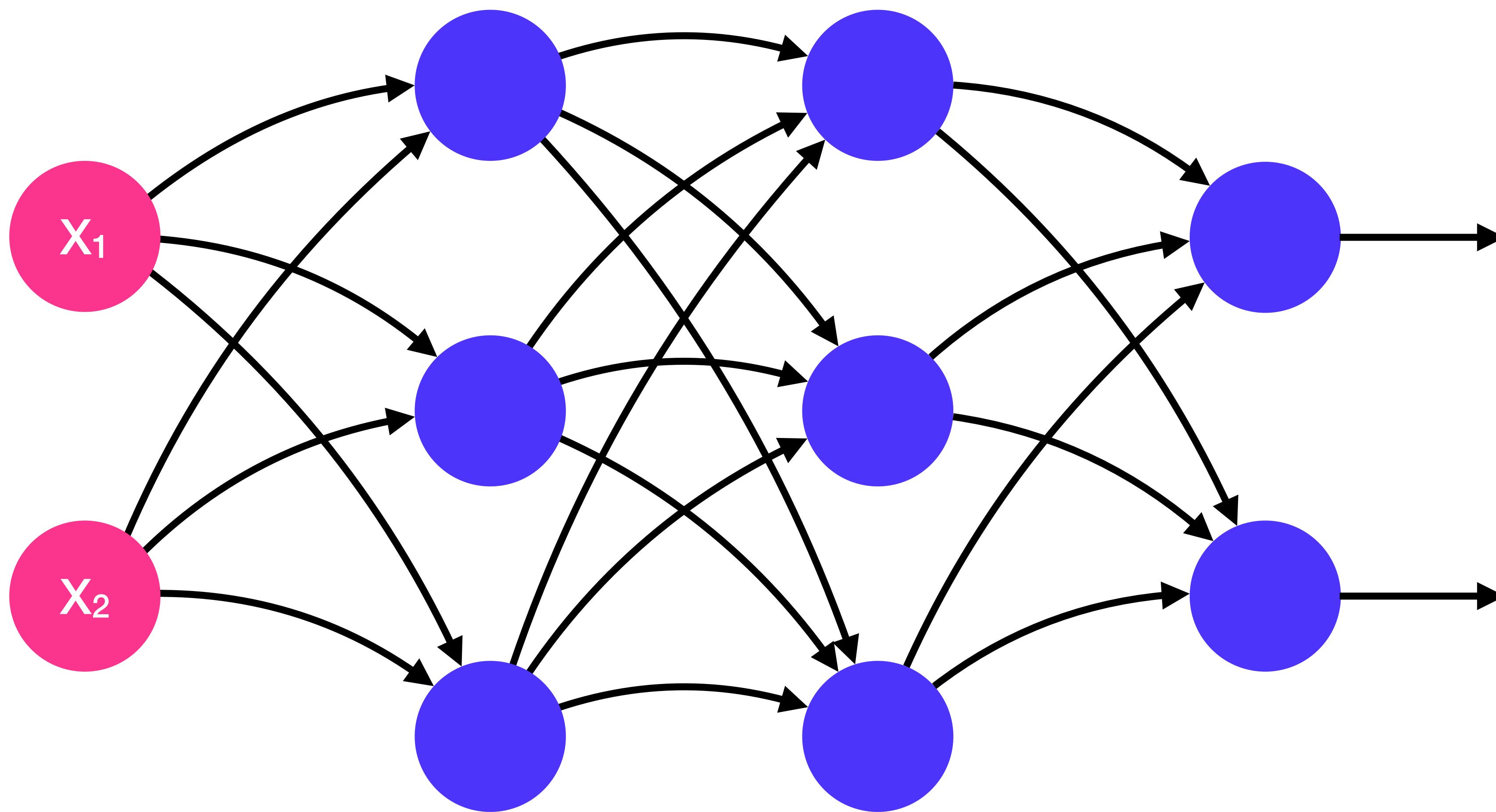


Redes Neurais

Para resolver isso, podemos agrupar os neurônios



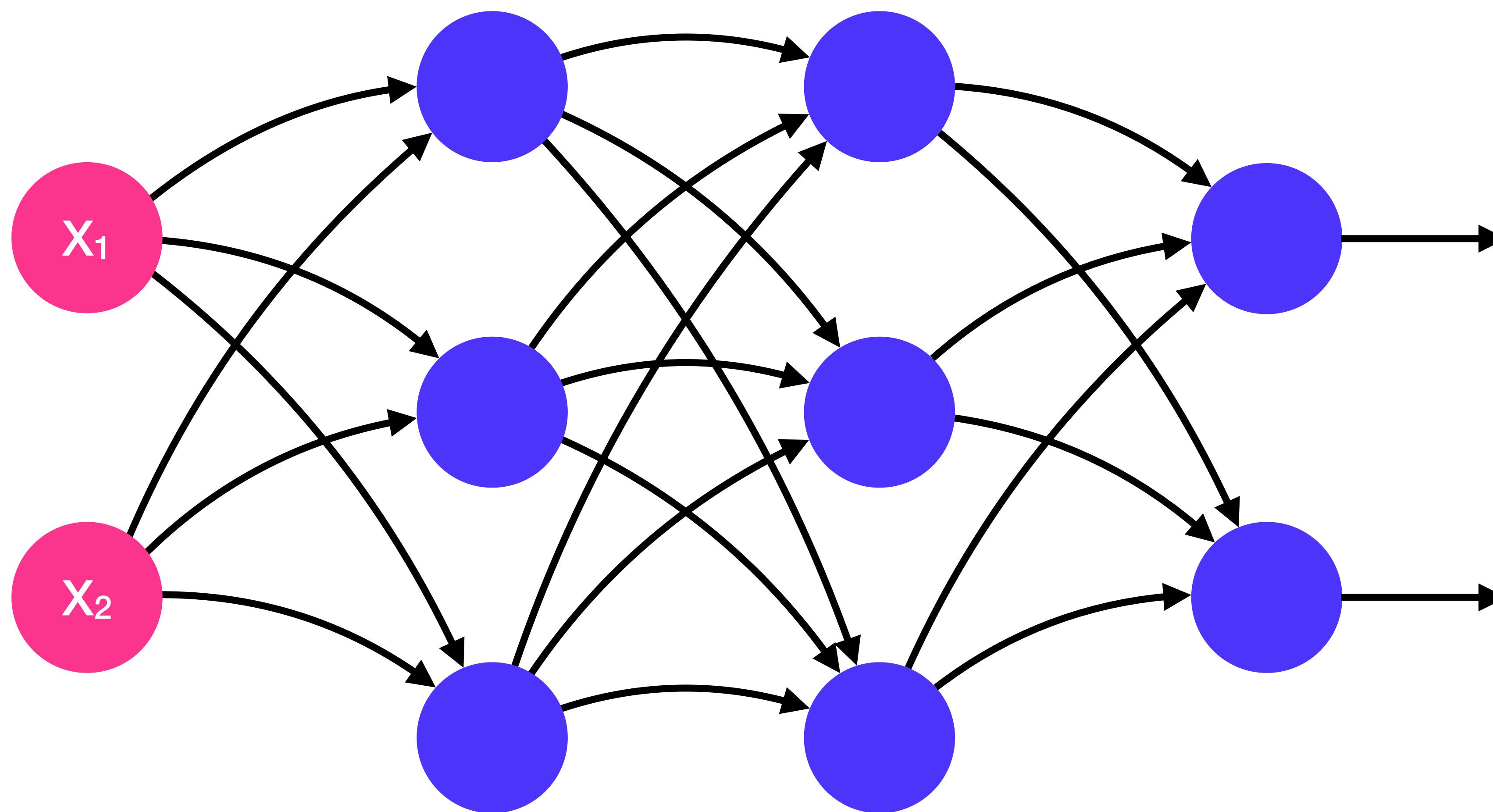


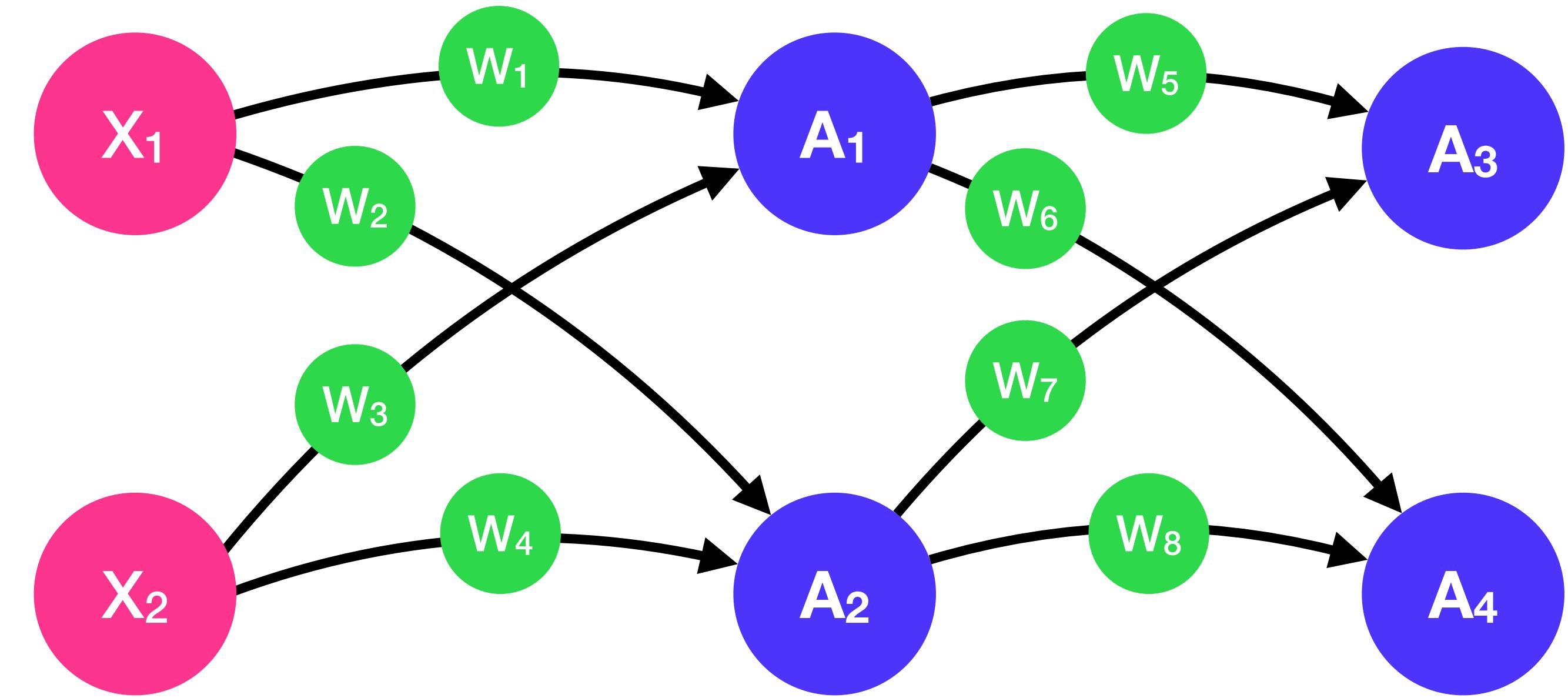


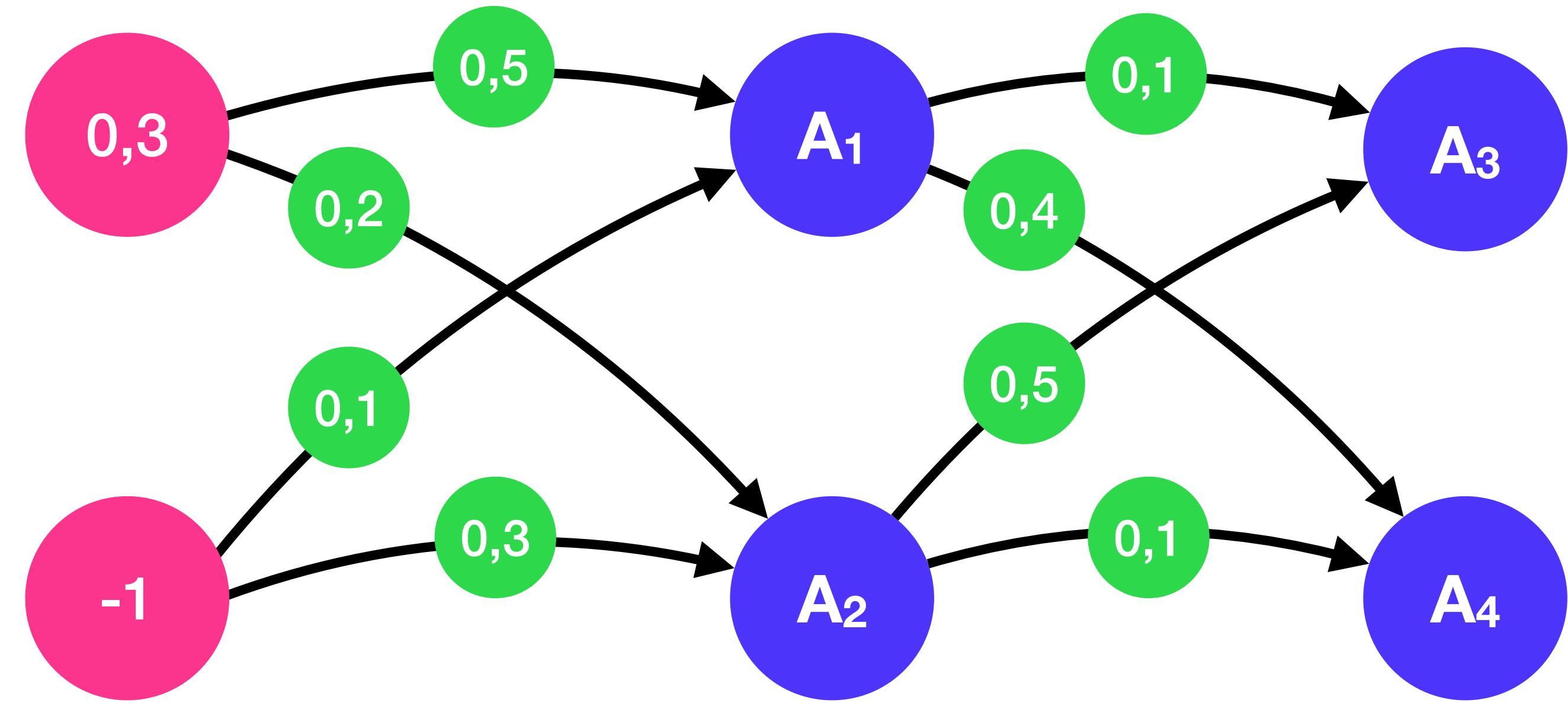
**Camada de
entrada**

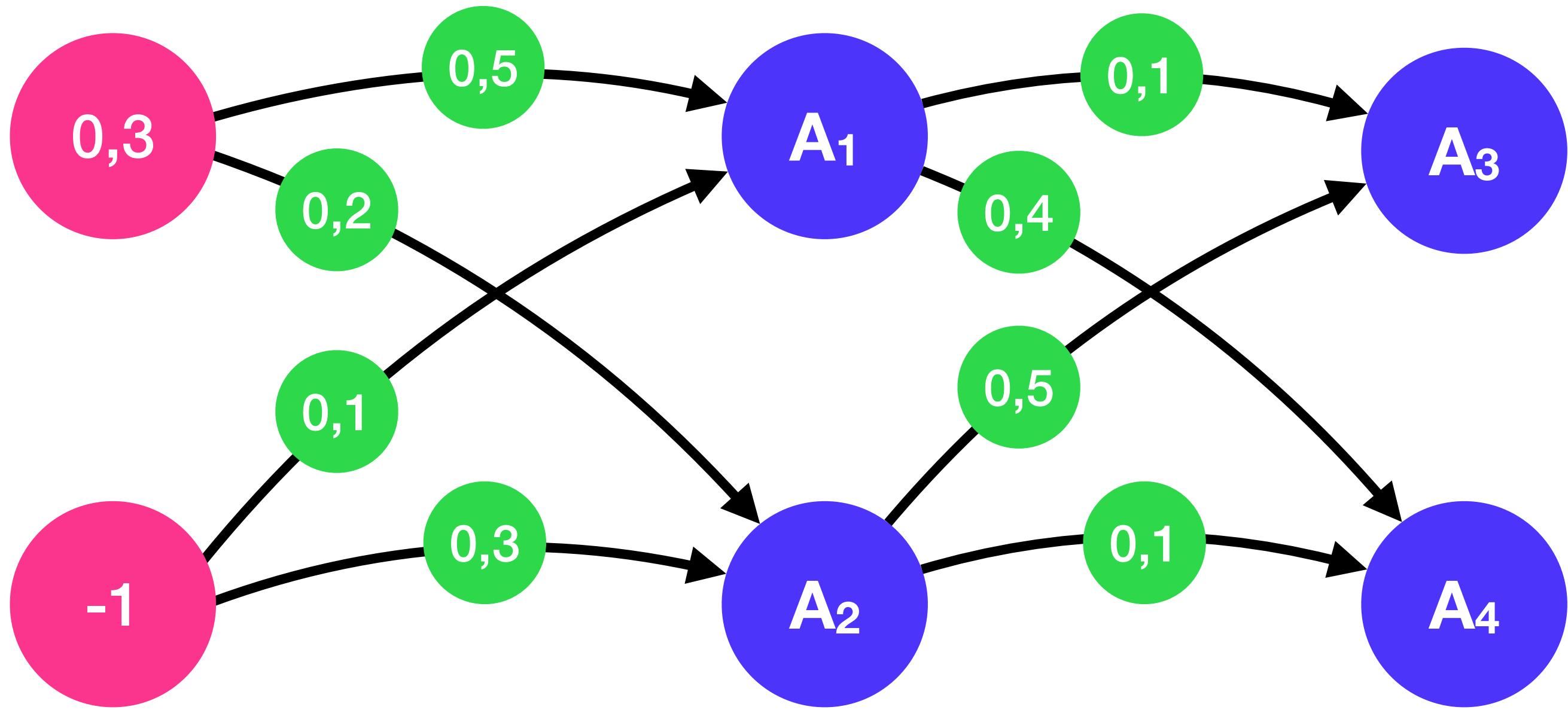
**Camada
intermediária**

**Camada
de saída**



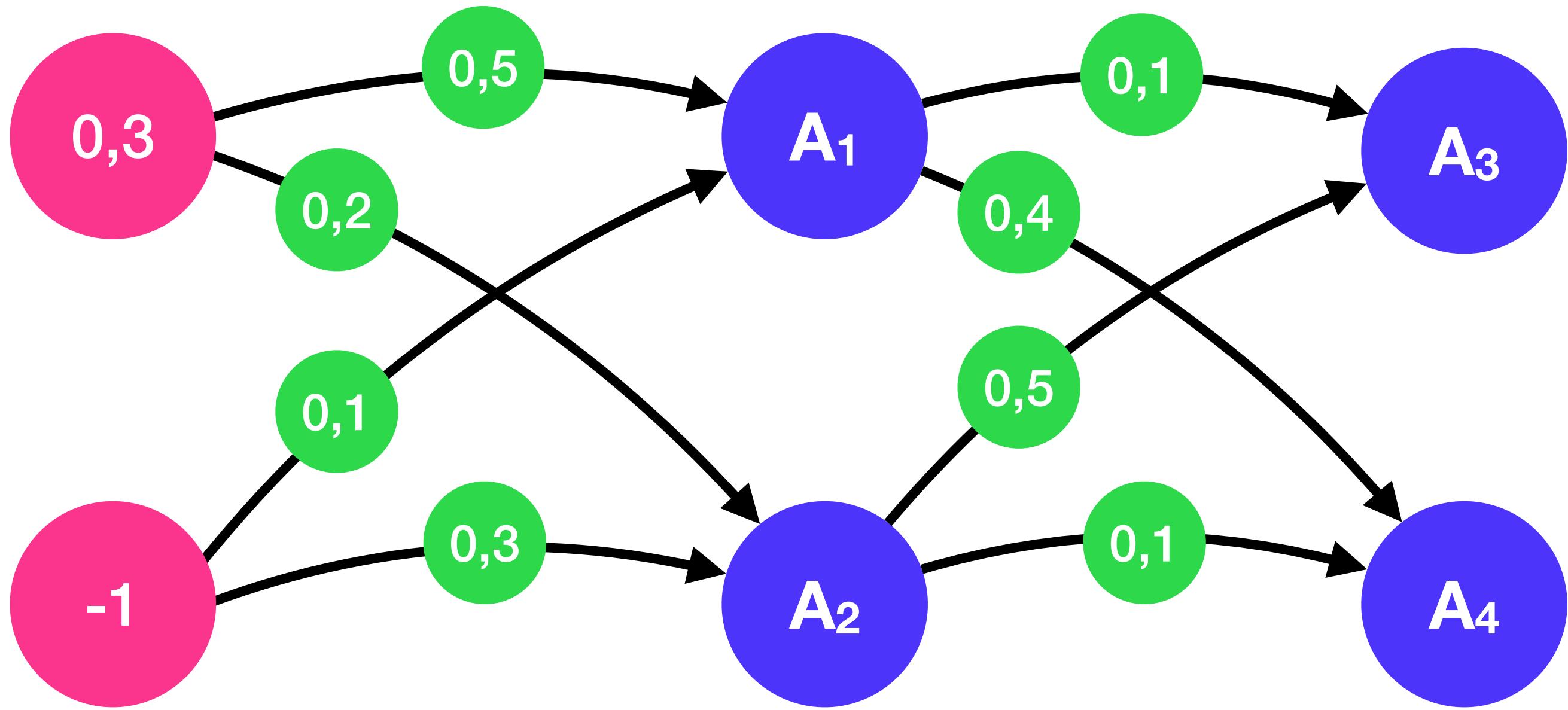






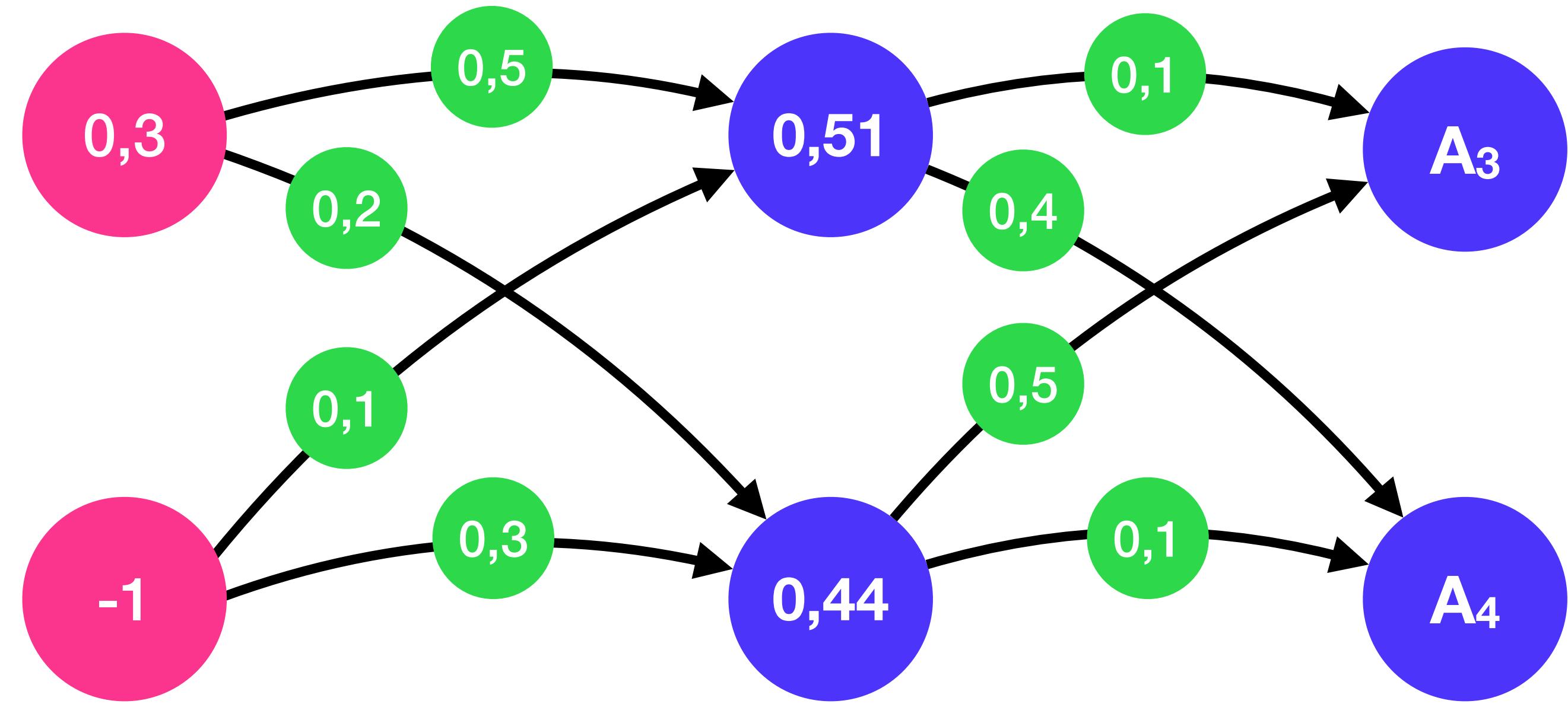
$$z_1 = 0,5 \times 0,3 + 0,1 \times -1 = 0,049$$

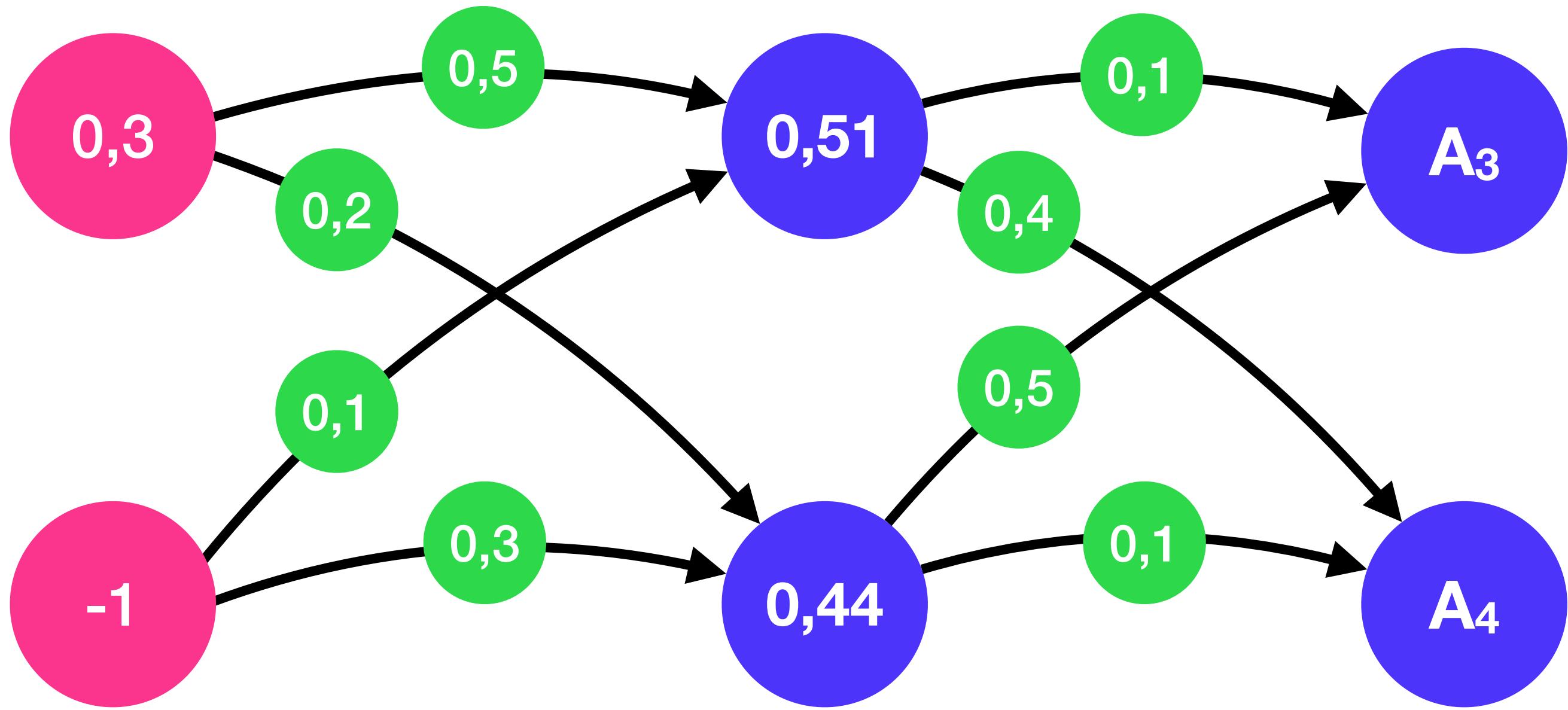
$$A_1 = F_a(0,049) = 0,51$$



$$z_2 = 0,2 \times 0,3 + 0,3 \times -1 = -0,24$$

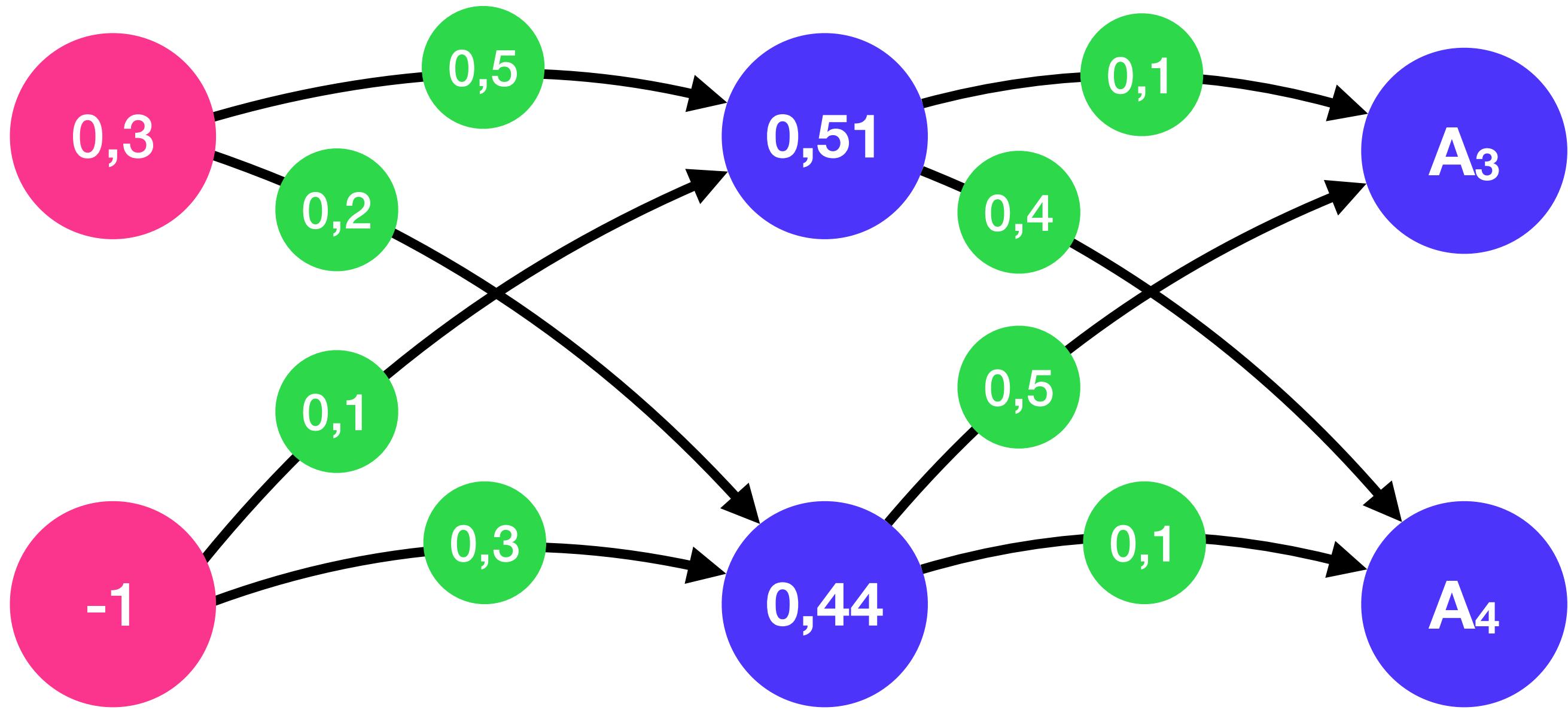
$$A_2 = F_a(-0,24) = 0,44$$





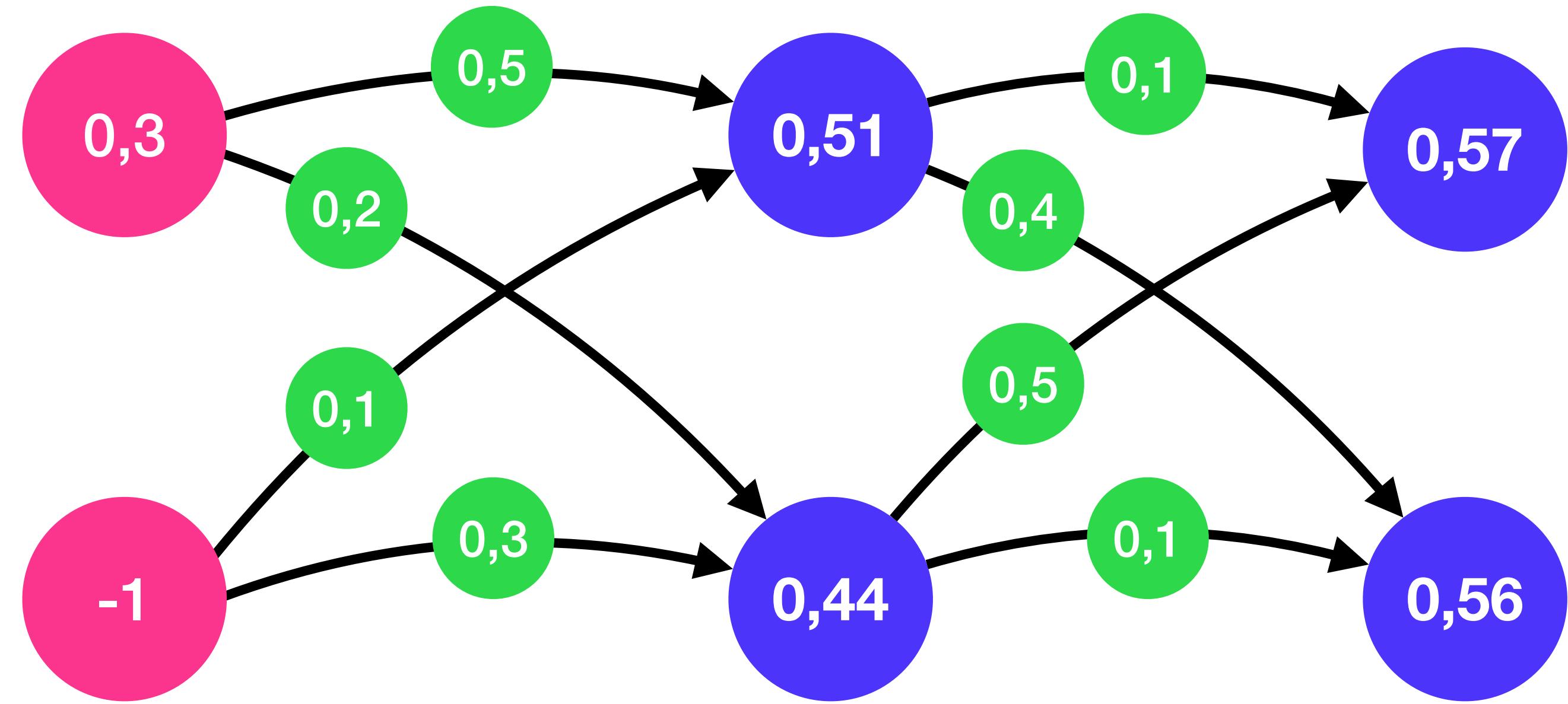
$$Z_3 = 0,1 \times 0,51 + 0,5 \times 0,44 = 0,271$$

$$A_3 = F_a(0,271) = 0,57$$



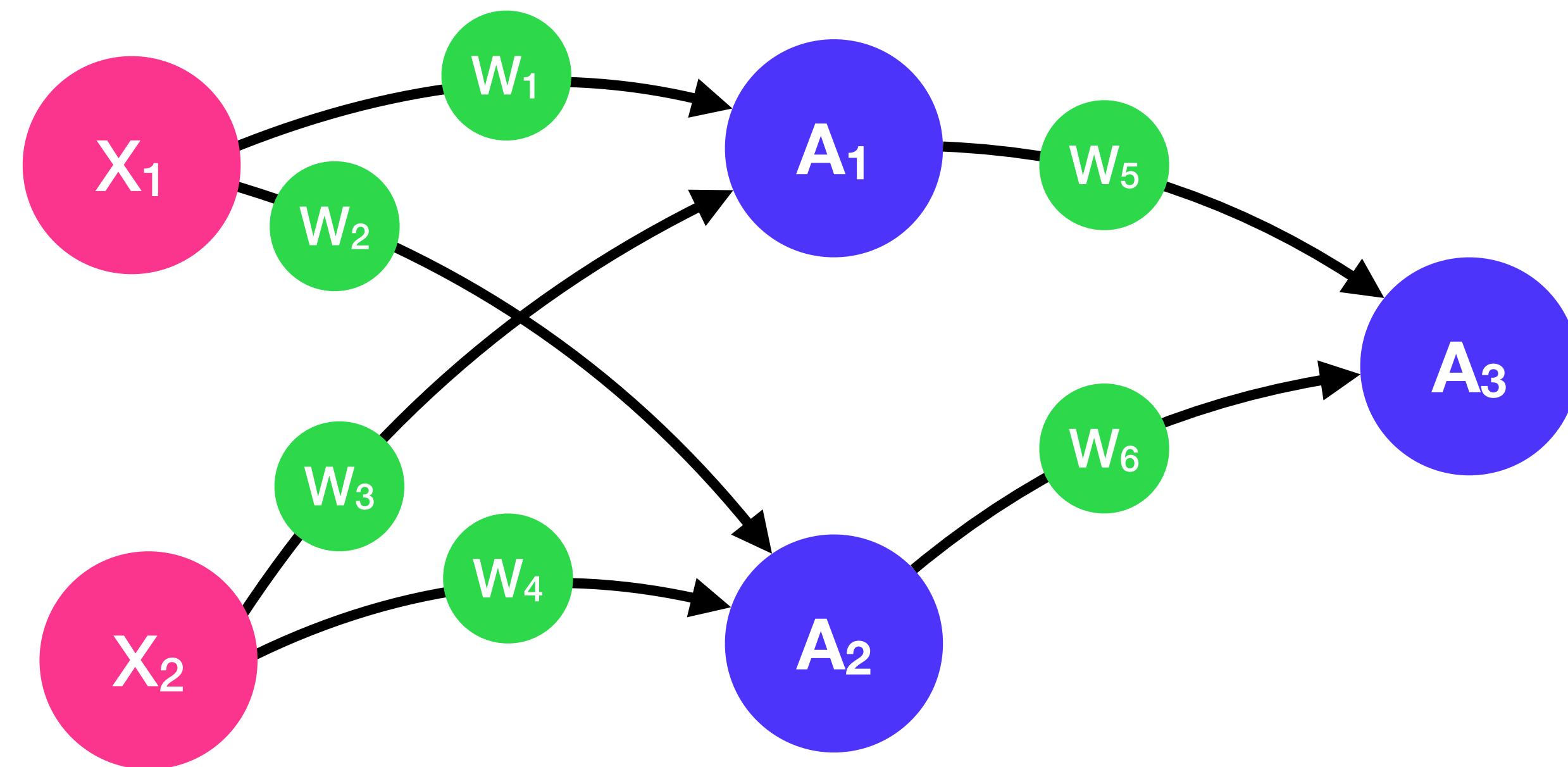
$$Z_4 = 0,4 \times 0,51 + 0,1 \times 0,44 = 0,248$$

$$A_4 = F_a(0,271) = 0,56$$



Treinando uma Rede Neural

Como treinar uma rede com múltiplas camadas?



Treinando uma Rede Neural

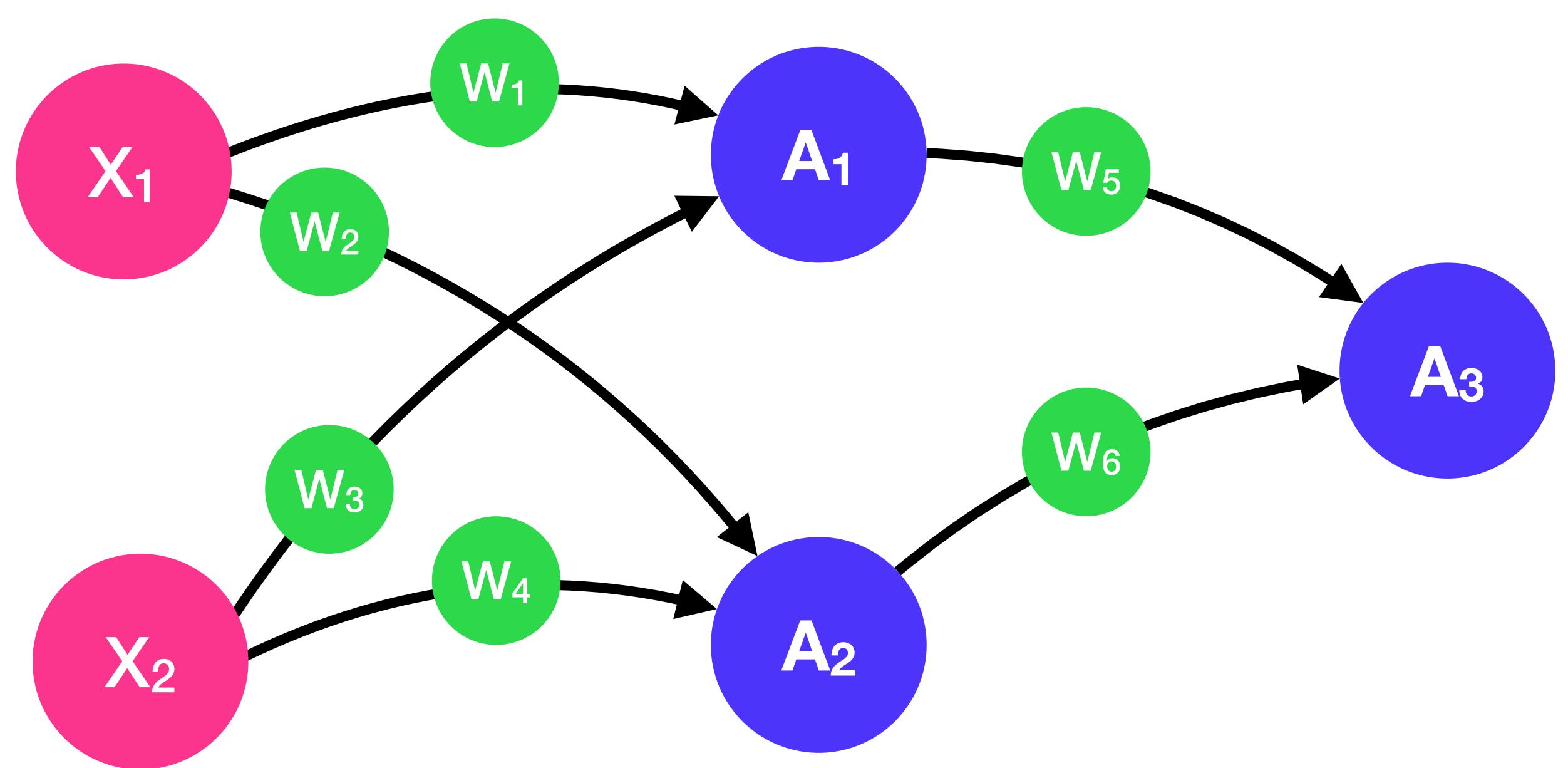
O algoritmo é semelhante ao que fizemos para apenas um neurônio

Iniciamos os pesos de forma aleatória

Entramos com os dados de treinamento e comparamos o resultado da rede com o resultado esperado

Ajustamos os pesos de acordo com o erro

Treinando uma Rede Neural



O valor de A_3 é a saída da rede.
Podemos compará-lo diretamente
com o resultado dos dados de
treinamento

E para A_1 e A_2 ? Quais são os
valores esperados? Não temos
esses dados

Treinando uma Rede Neural

Para treinar uma rede com multcamadas usamos o algoritmo backpropagation

Nele, vamos definir uma função que mede o erro da saída da rede

Também conseguimos calcular através do gradiente da função de erro, em qual sentido devemos ajustar os pesos para que o erro diminua

Função de Erro

Calculamos o erro para um exemplo através da função:

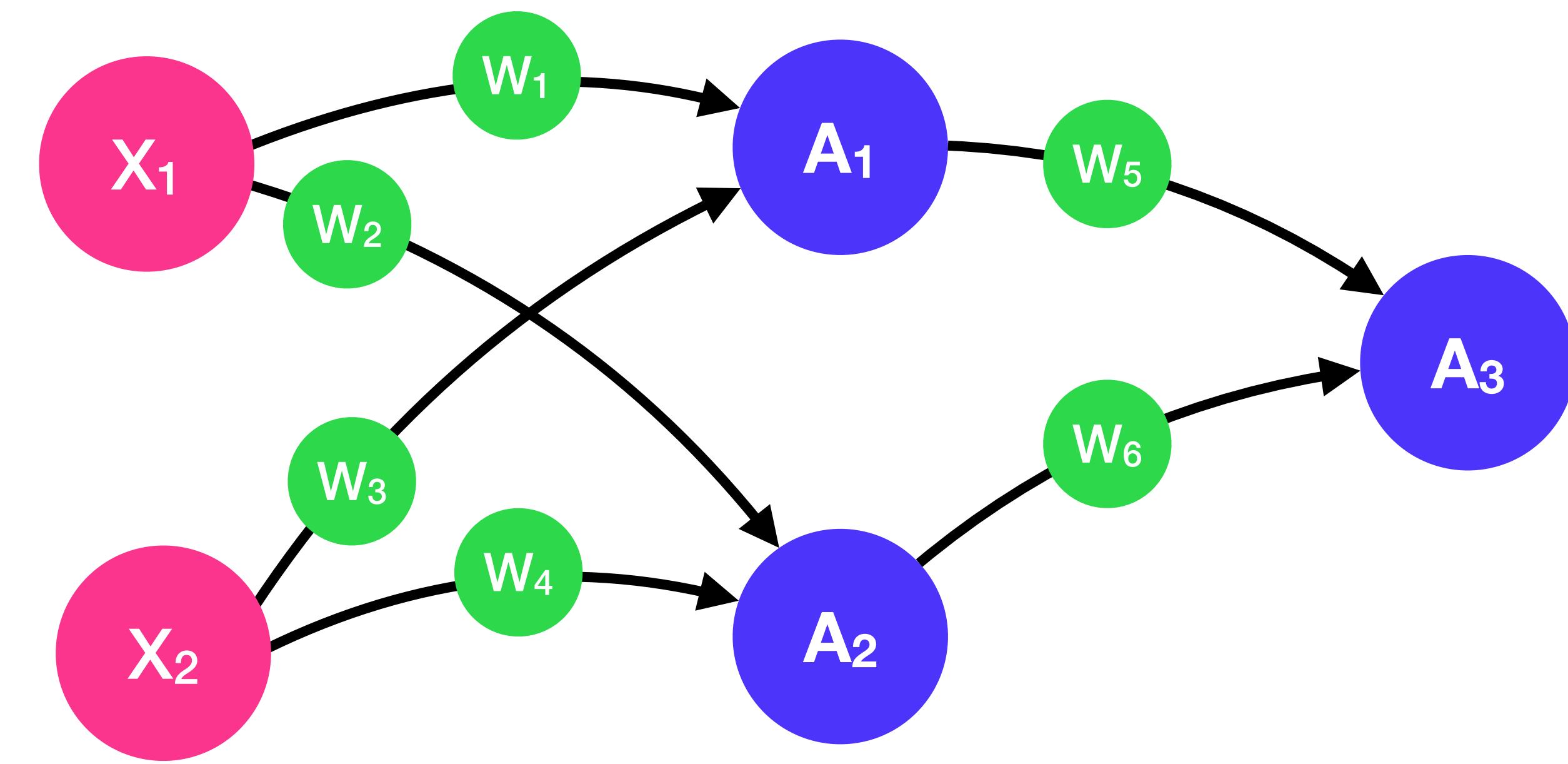
$$J(w) = \frac{1}{2}(a_{obtido} - a_{esperado})^2$$

Ou podemos também fazer um média para vários exemplos

Treinando uma Rede Neural

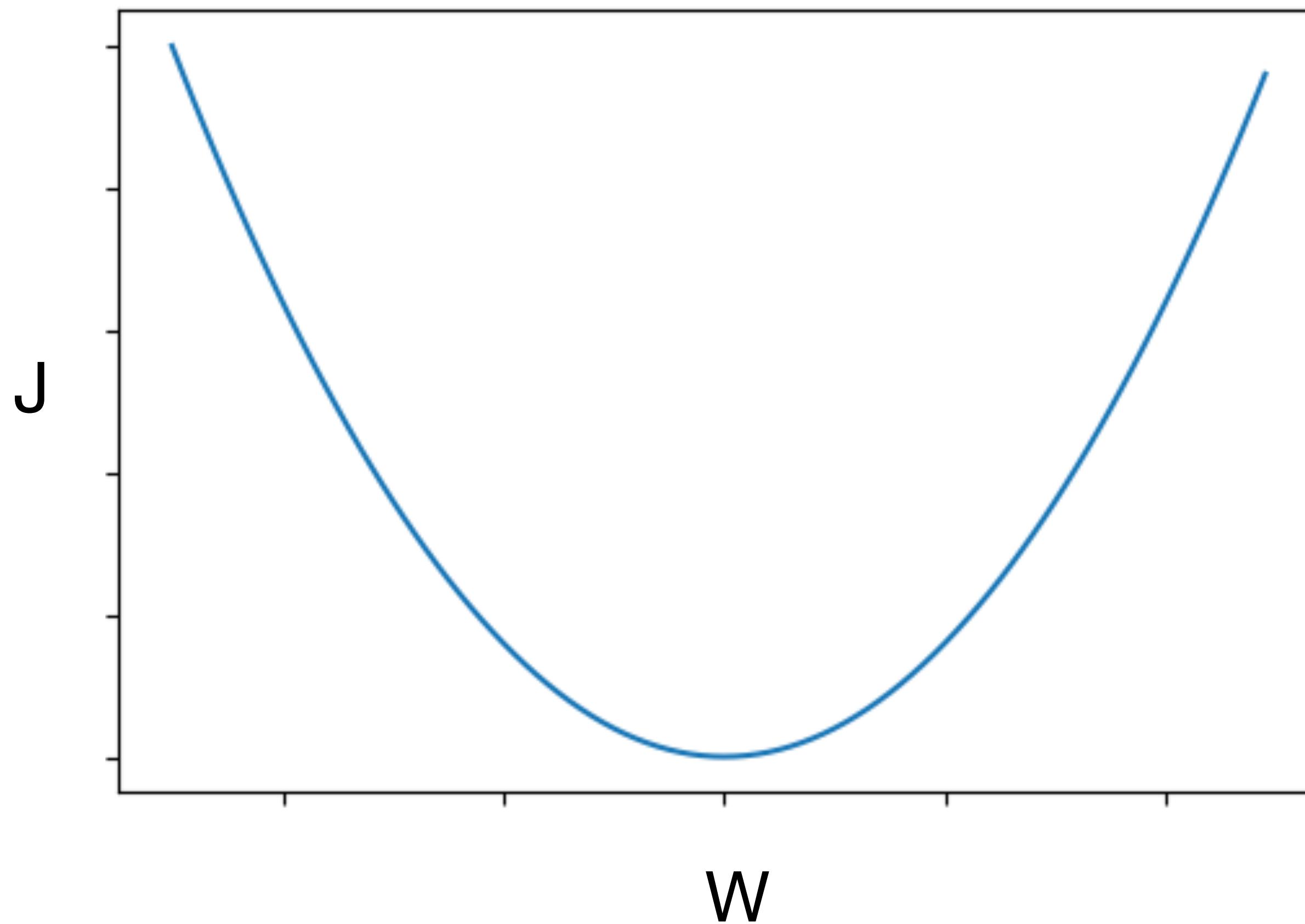
Com isso, a regra de atualização dos pesos é:

$$w_i \leftarrow w_i - \eta \frac{\partial J}{\partial w_i}$$



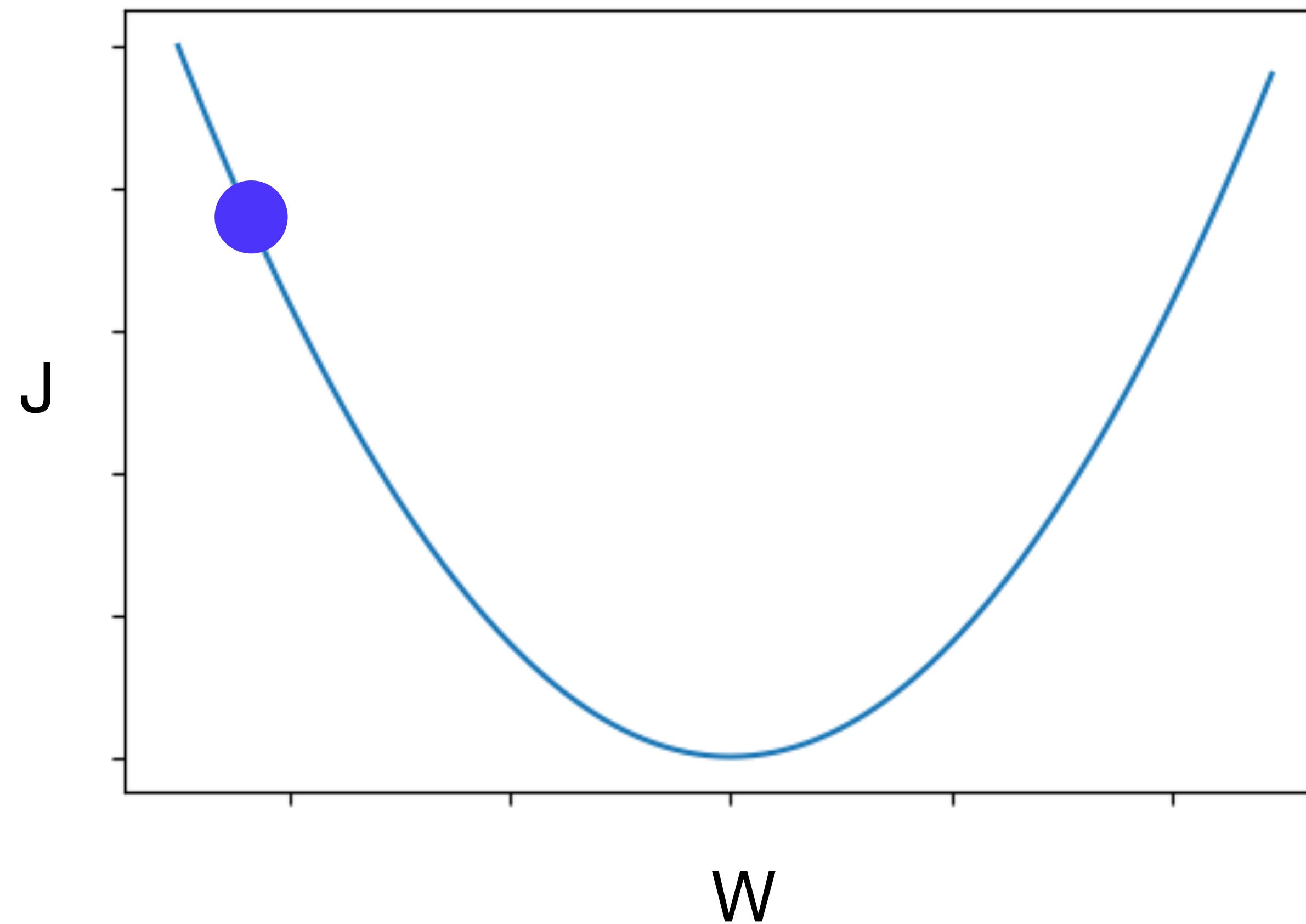
Treinando uma Rede Neural

A função de erro tem uma forma como essa:



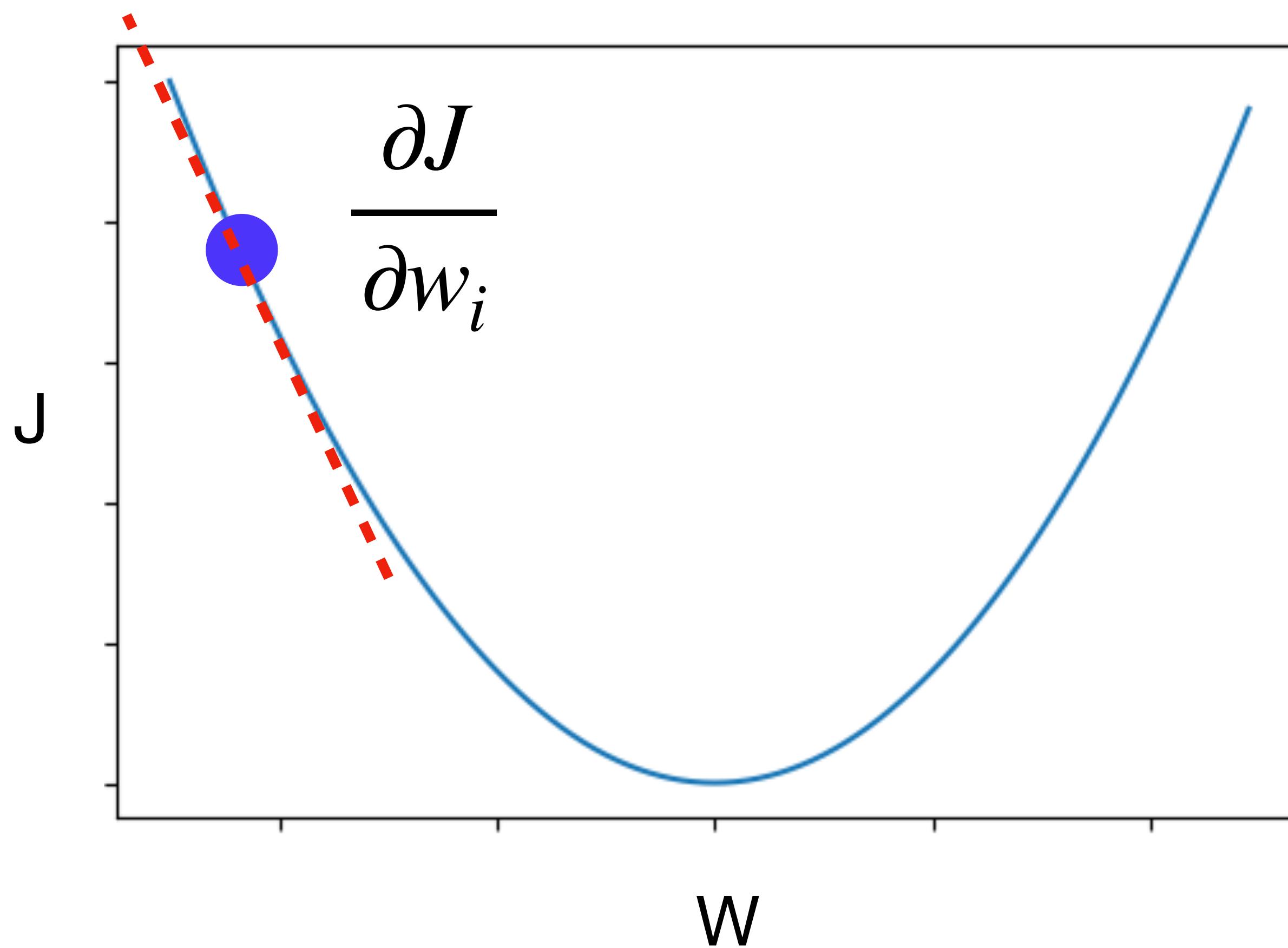
Treinando uma Rede Neural

Inicializamos o valor de W aleatoriamente



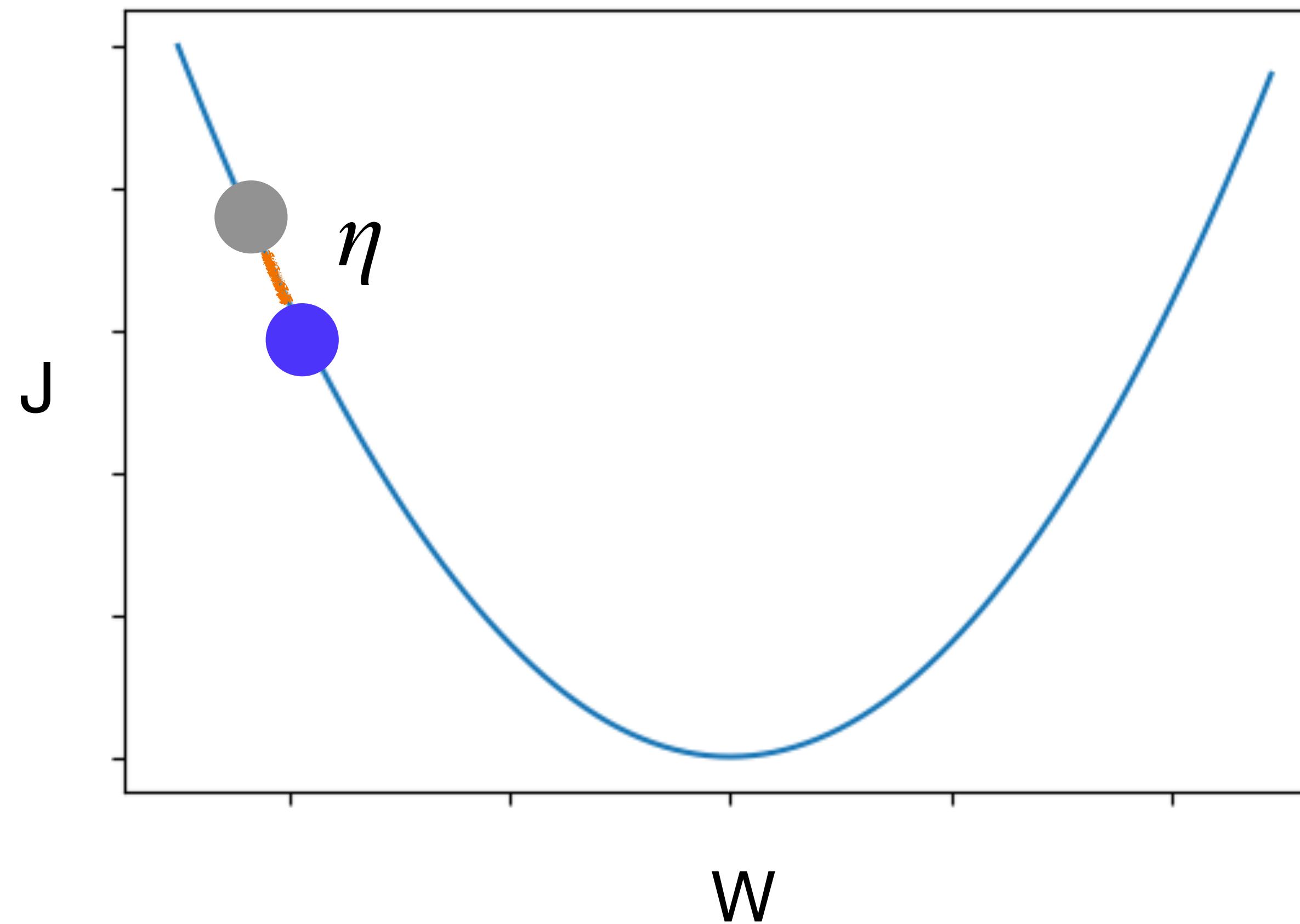
Treinando uma Rede Neural

A derivada nos fornece a inclinação do ponto



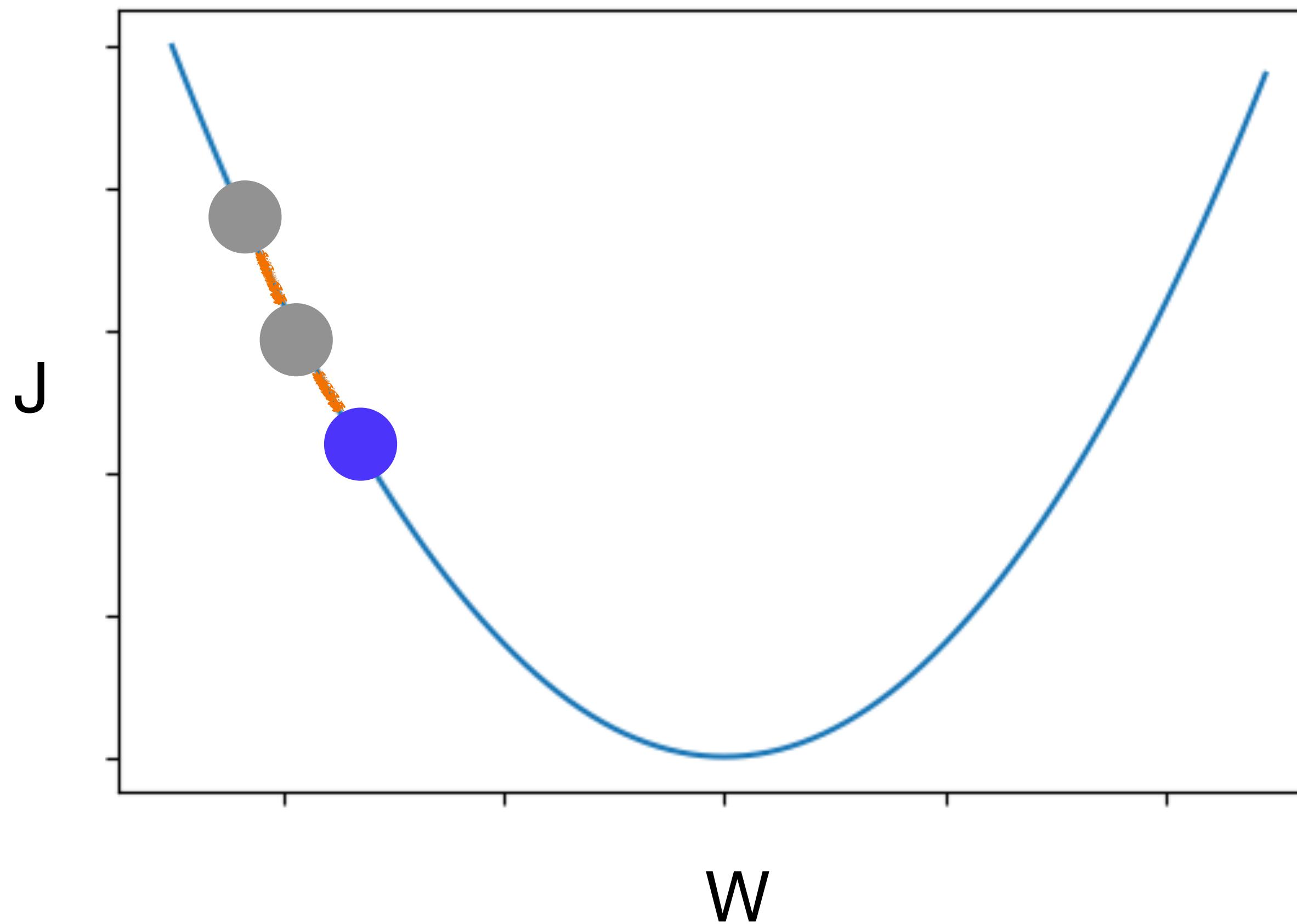
Treinando uma Rede Neural

A taxa de aprendizagem determina o tamanho do passo de atualização



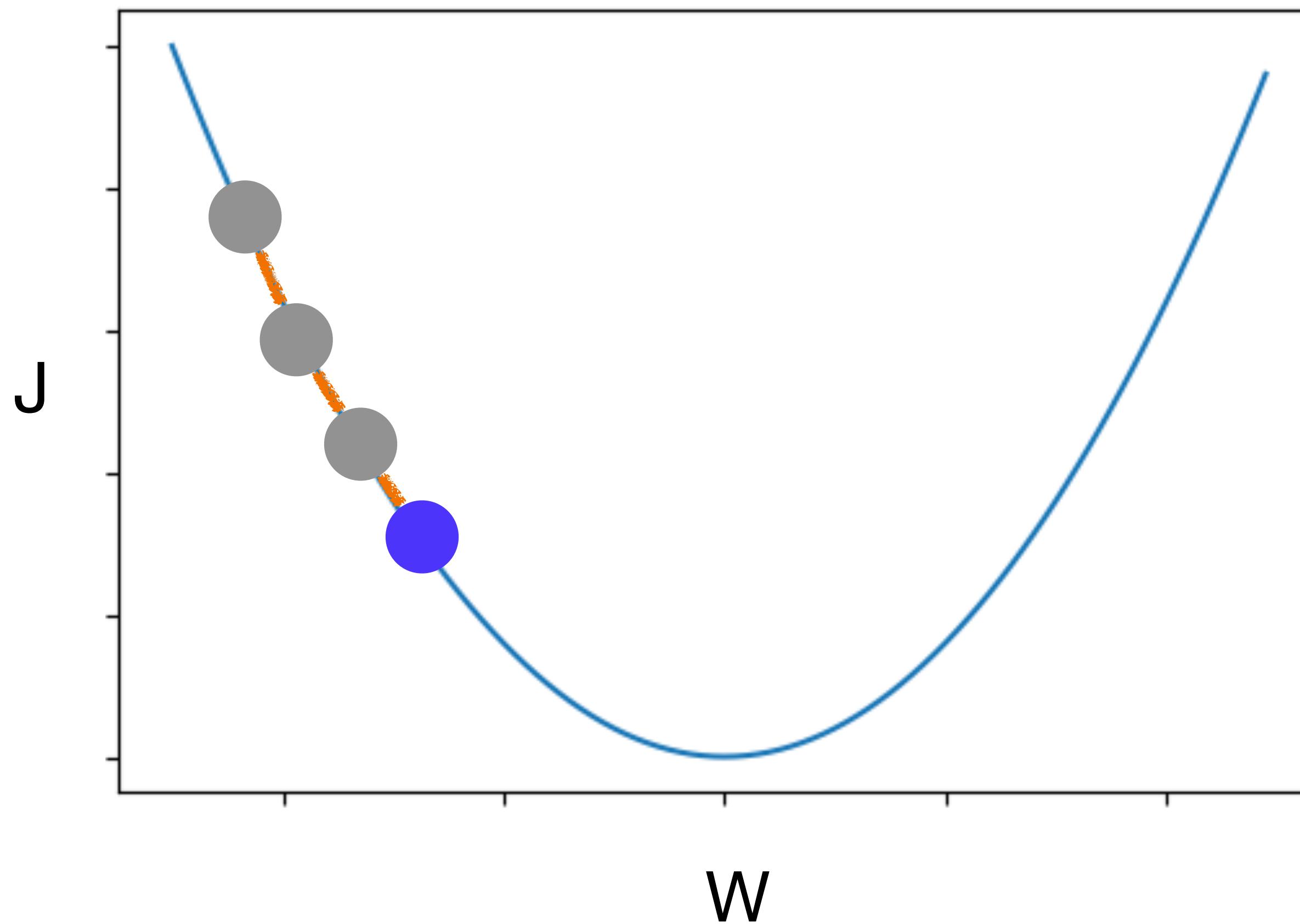
Treinando uma Rede Neural

Repetimos atualização de W múltiplas vezes



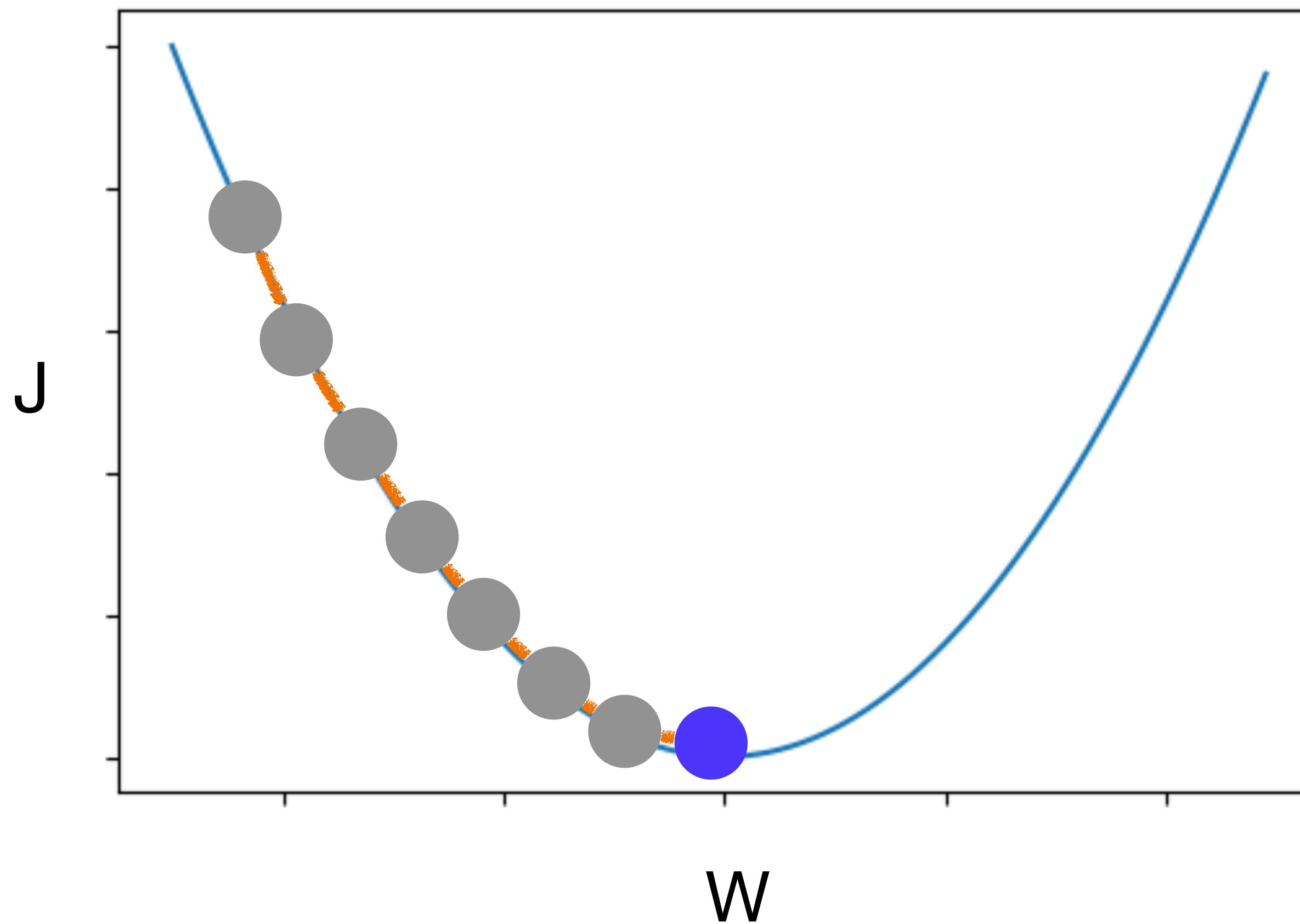
Treinando uma Rede Neural

Repetimos atualização de W múltiplas vezes



Treinando uma Rede Neural

Repetimos atualização de W múltiplas vezes



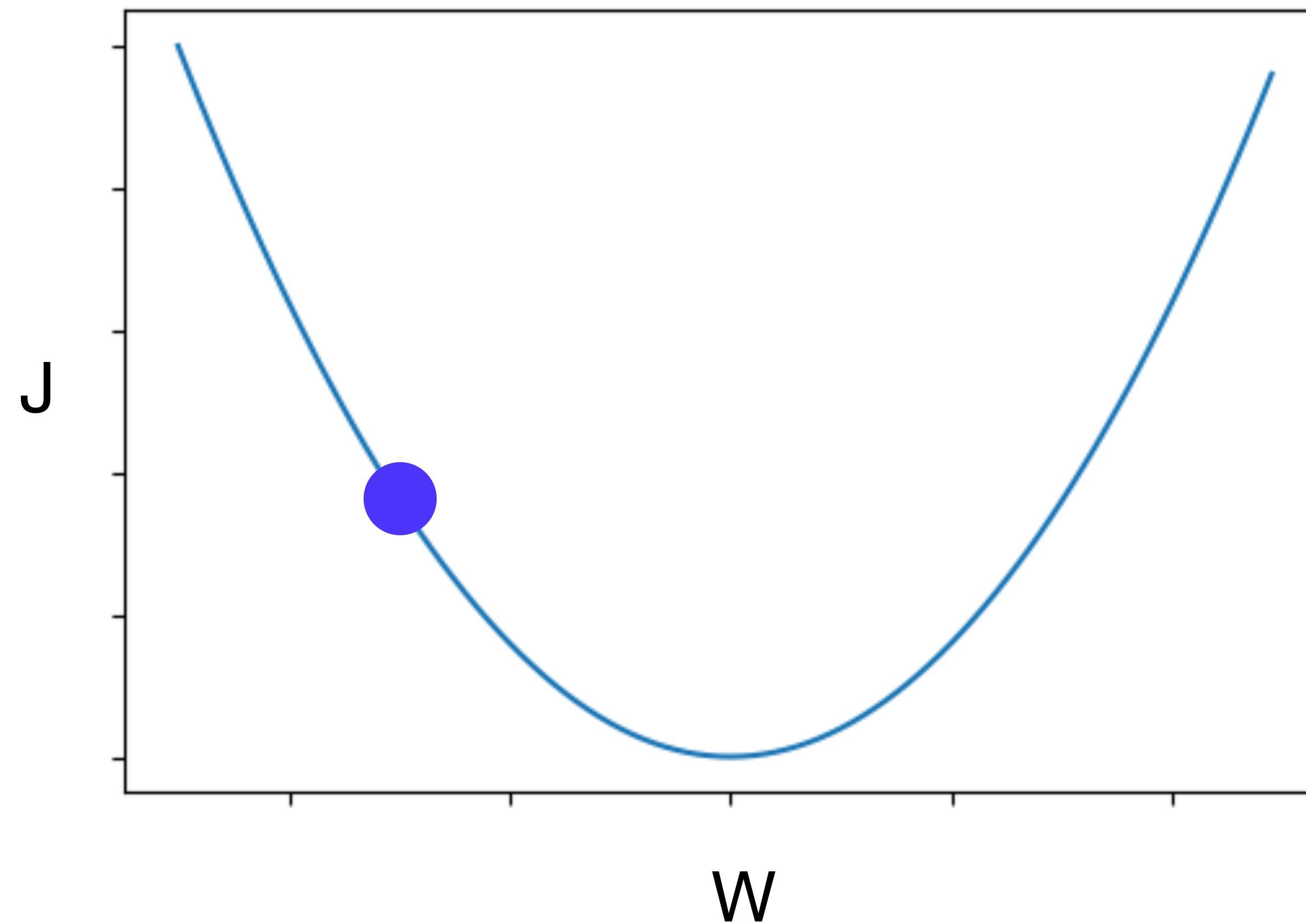
Treinando uma Rede Neural

Valores pequenos para a taxa de aprendizagem podem fazer o treinamento demorar muito a convergir

Valores muito grandes podem fazer o treinamento divergir

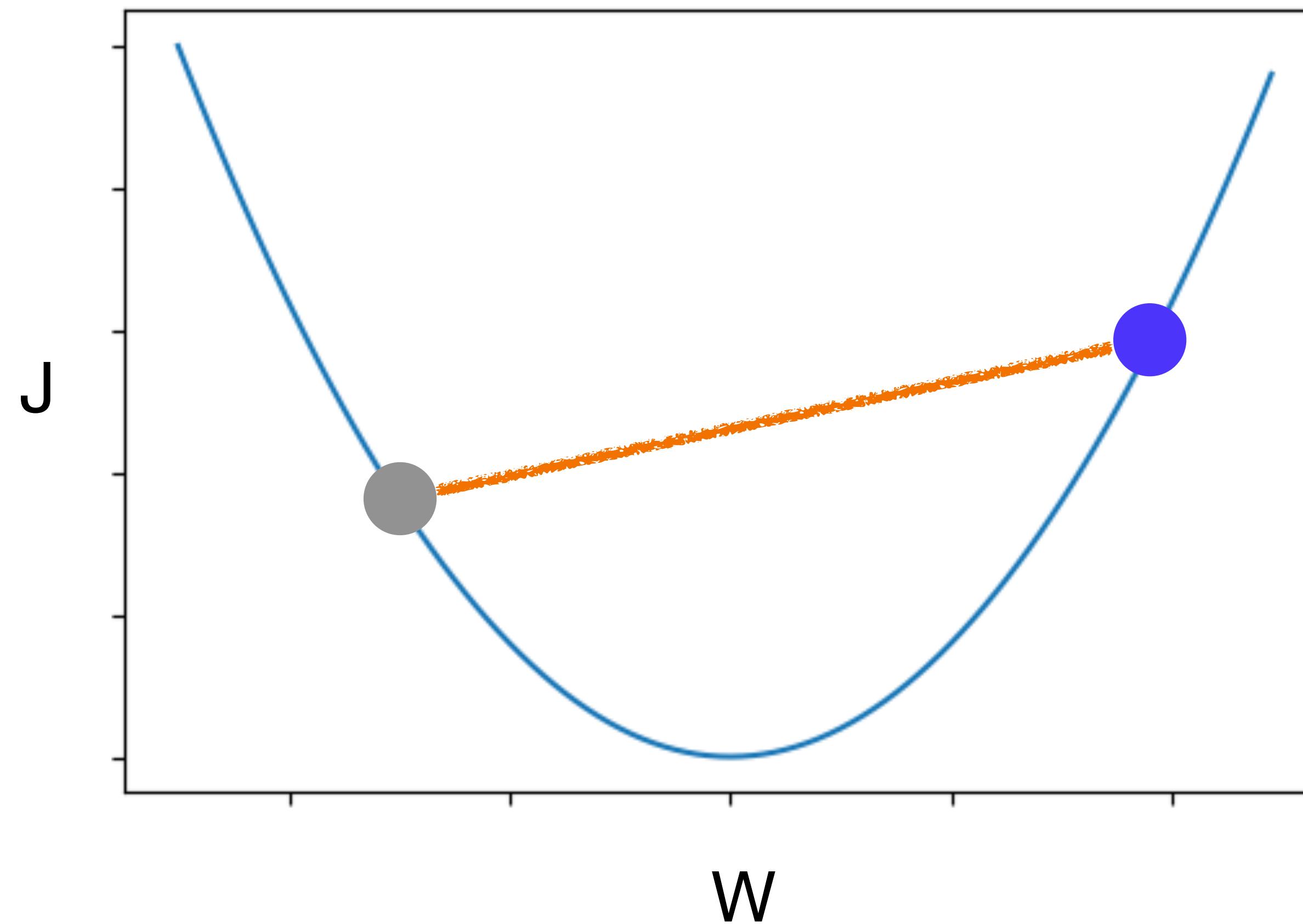
Treinando uma Rede Neural

Inicializamos o valor de W aleatoriamente



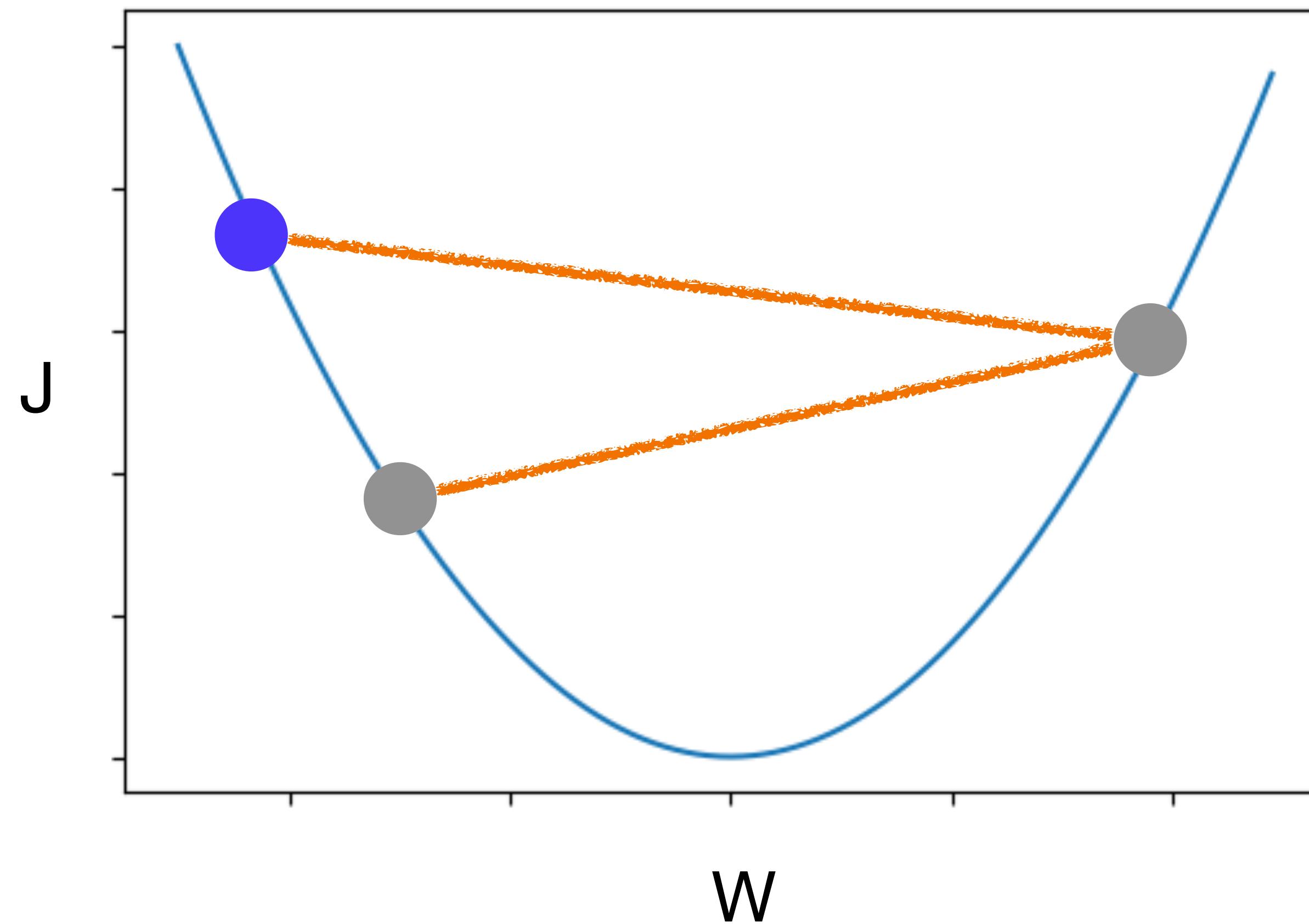
Treinando uma Rede Neural

O passo de atualização foi tão grande que o erro aumentou



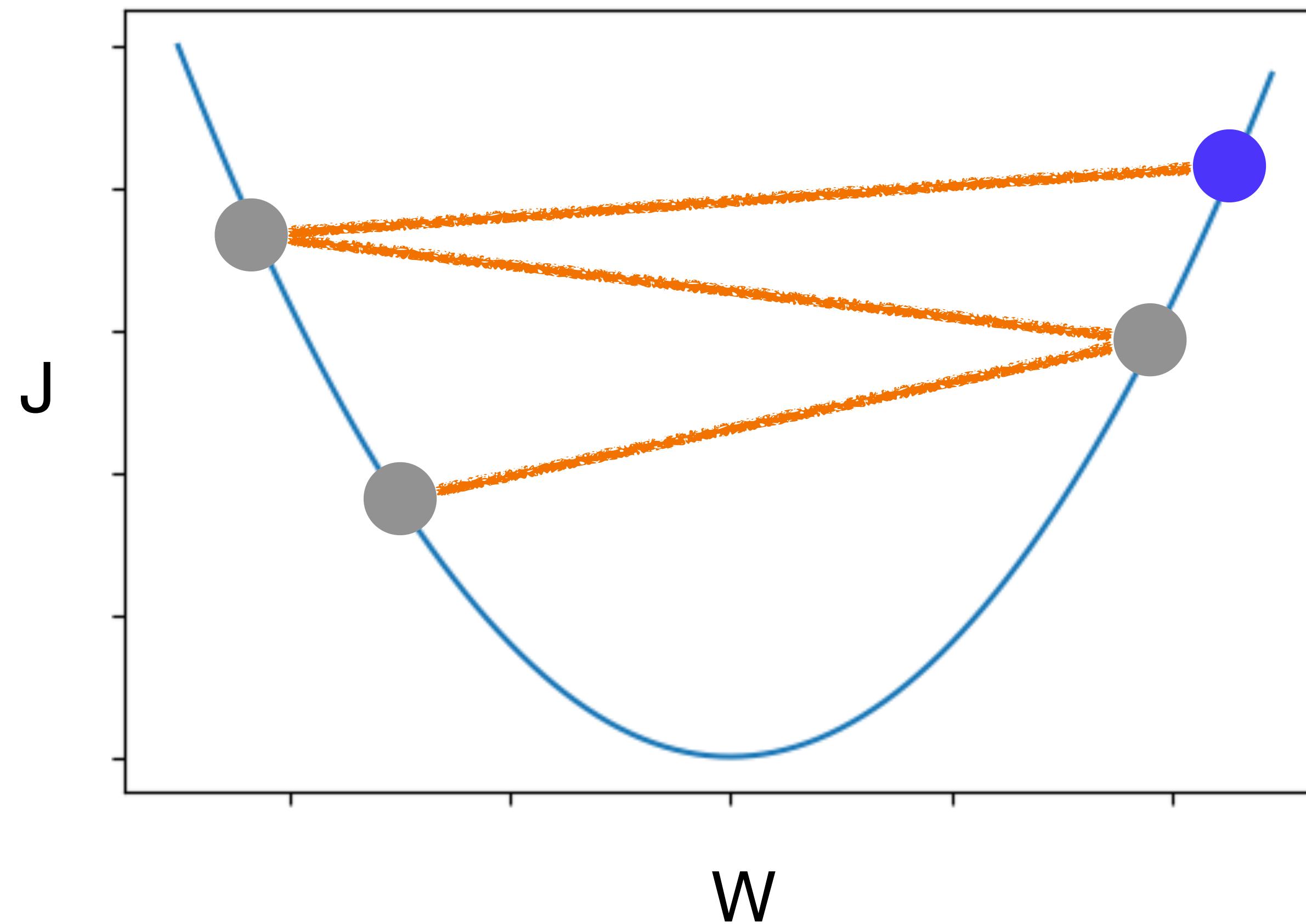
Treinando uma Rede Neural

O erro continua aumentando



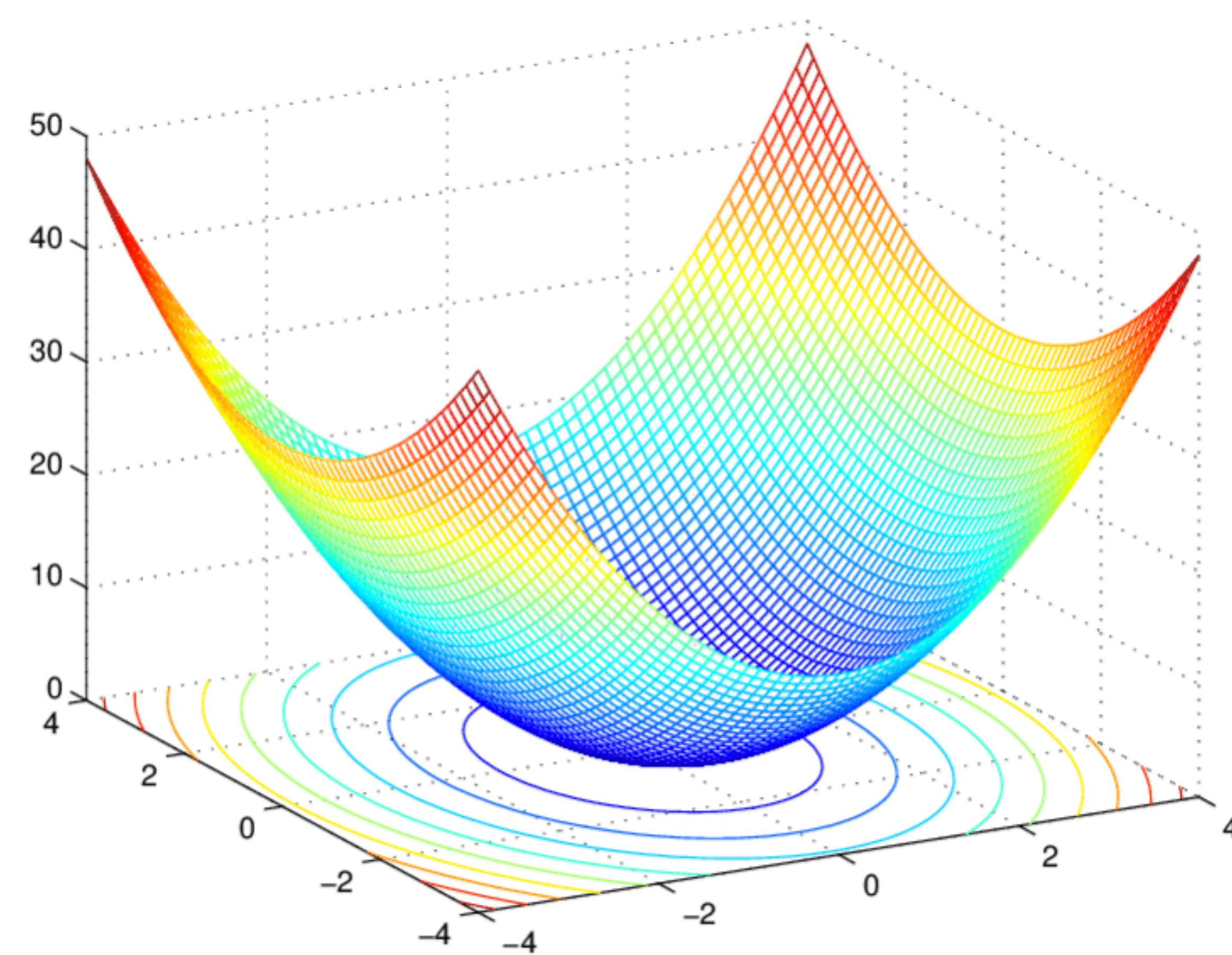
Treinando uma Rede Neural

O erro continua aumentando



Treinando uma Rede Neural

Na prática a função de J é calculada usando vários W s



Bibliotecas

Existem bibliotecas que facilitam a criação de redes neurais

Por exemplo:

- TensorFlow
- PyTorch

Bibliotecas

Rede neural com 4 entradas, duas camadas intermediárias com 32 neurônios e a camada de saída com 3 neurônios:

```
● ● ●  
  
model = Sequential()  
model.add(Dense(32, activation='relu', input_dim=4))  
model.add(Dense(32, activation='relu'))  
model.add(Dense(3, activation='softmax'))  
  
sgd = SGD(lr=0.01)  
  
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=[  
    'accuracy'])  
  
model.fit(x_train, y_train, epochs=100, batch_size=32)  
score = model.evaluate(x_test, y_test, batch_size=32)
```

Redes Neurais

TensorFlow Playground é uma excelente ferramenta para entendermos visualmente o funcionamento de uma rede neural

playground.tensorflow.org

Conclusão

Ainda estamos longe de alcançar a capacidade de um cérebro

Um computador processa informações mais rapidamente que um neurônio, mas essa diferença é compensada pelo paralelismo do cérebro

É difícil de entender o funcionamento de uma rede

Conclusão

Possui bom poder de generalização

Deep Learning (redes neurais com muitas camadas) tem se destacado como a melhor técnica para diversos problemas de classificação

Bate-papo com Geraldo Ramos

EMPREENDEDOR SERIAL,
GRADUADO EM MARKETING
PELO IESP. FUNDADOR DA
GIGAHOST EM 2003, 6PS
CORPORATION EM 2007 E DA
PLATAFORMA DE MENTORIA
PARA PROGRAMADORES
HACKHANDS EM 2013, ESTA
ÚLTIMA ADQUIRIDA PELA
PLURALSIGHT. EM 2019
FUNDOU A MOISES.AI,
STARTUP QUE USA
INTELIGÊNCIA ARTIFICIAL
PARA PRODUÇÃO MUSICAL.



22/07
19:00 - Ao vivo
YouTube
[/arialab](https://www.youtube.com/c/Larialab)