UFG

RNP-VC

Instituto de Informática
UFG

Aula 3: Desenvolvendo Redes Neurais Profundas

Lucas Pereira, Rafael Teixeira, Lucas Assis, Anderson Soares
Instituto de Informática
Universidade Federal de Goiás (UFG)

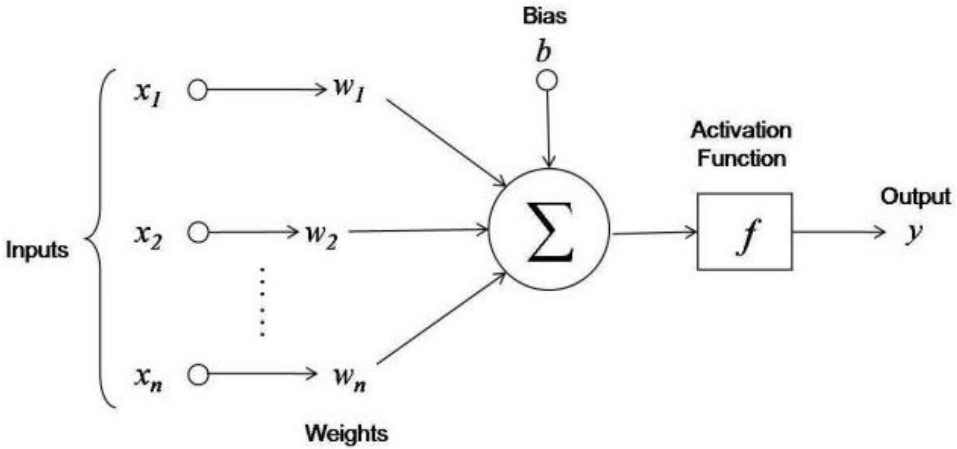
DEEP LEARNING
BRASIL

Sumário

- No último episódio...
- Hardware para DL
- Software para DL
- Ciclo vicioso de ML
- No próximo episódio...

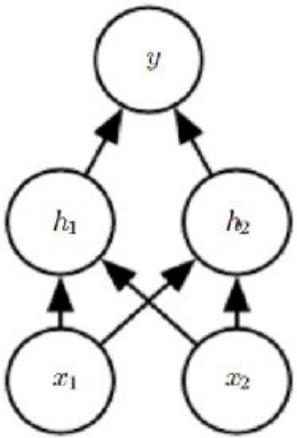
No último episódio...

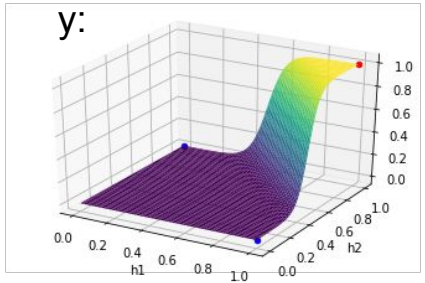
- Perceptron

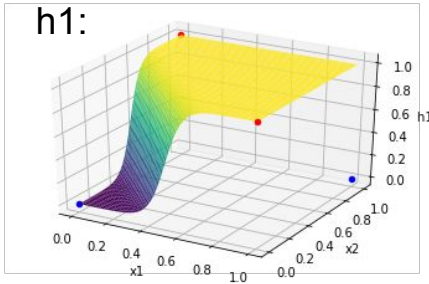


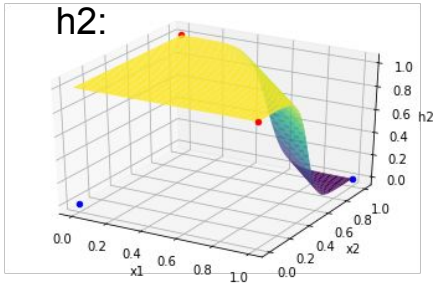
No último episódio...

- MLP como aproximador universal de funções



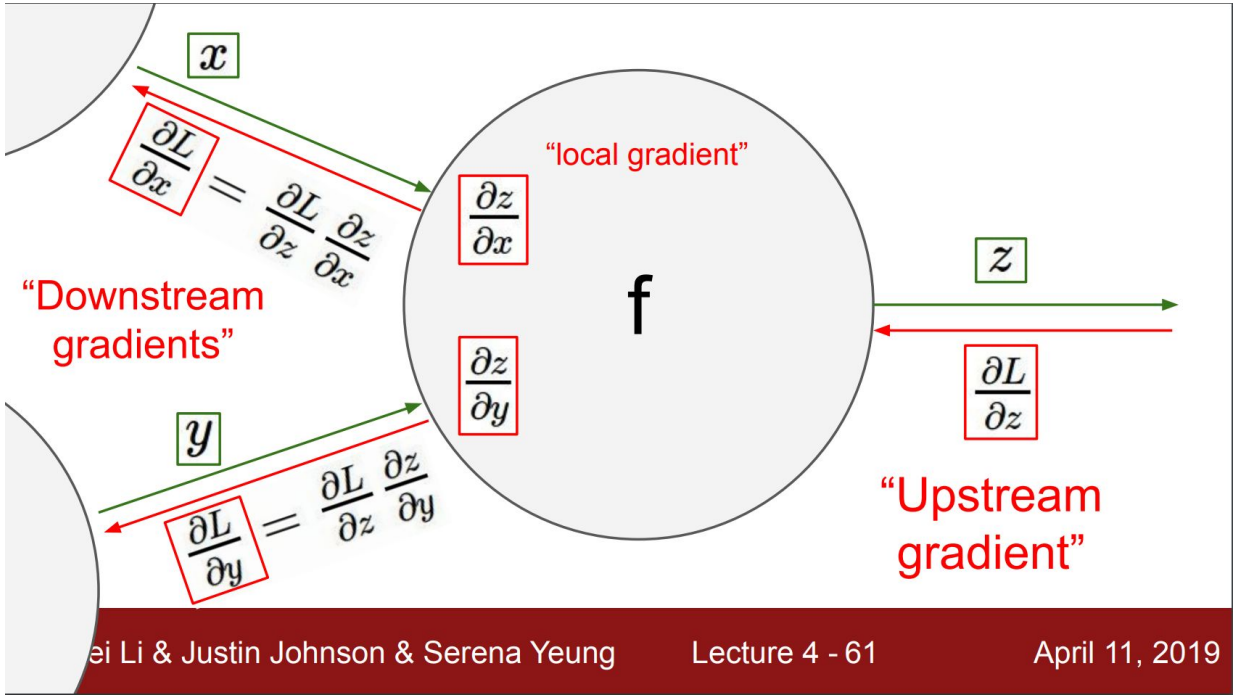






No último episódio...

- Backprop



No último episódio...

- Redes convolucionais (CNN)

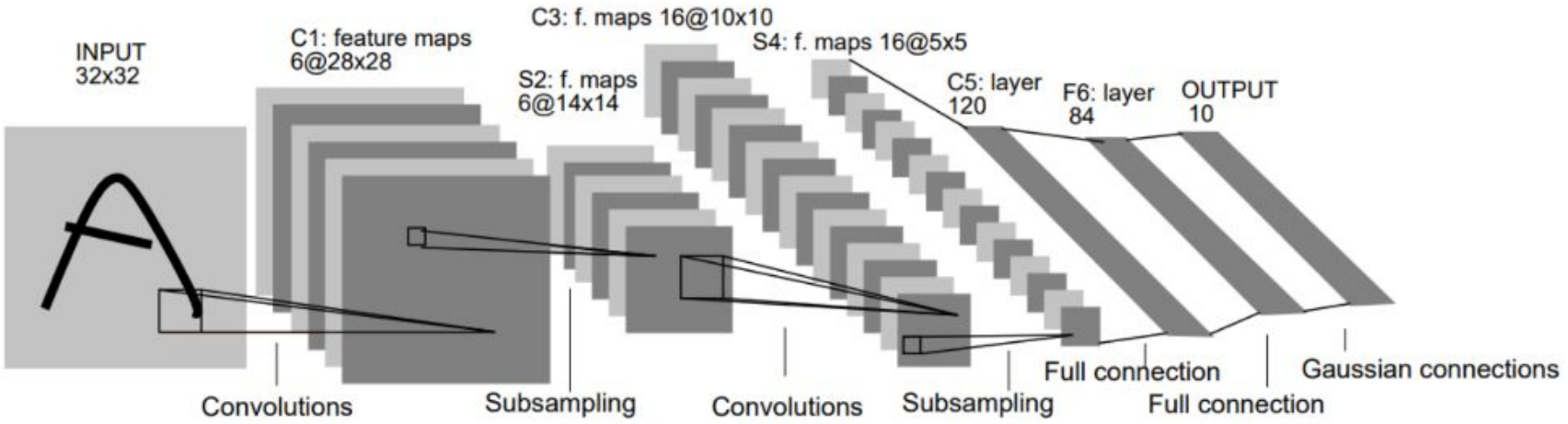


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Hardware para DL

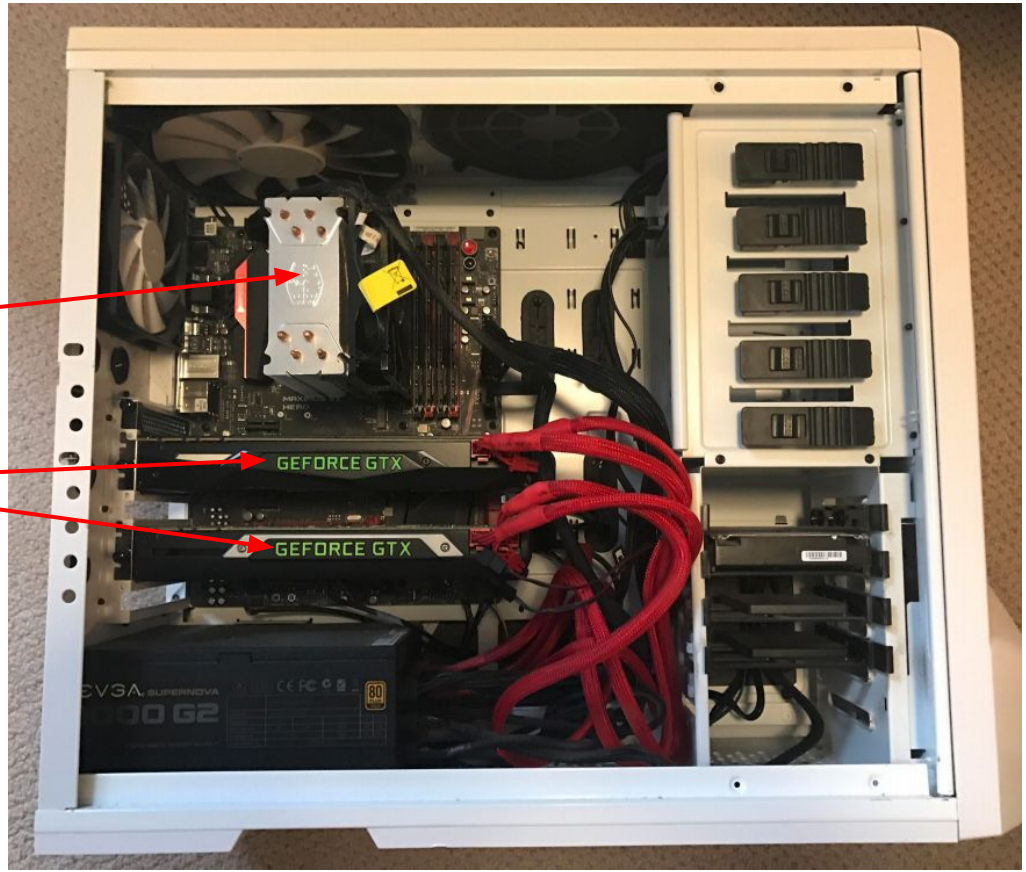
- CPU vs GPU

Hardware para DL

- CPU vs GPU

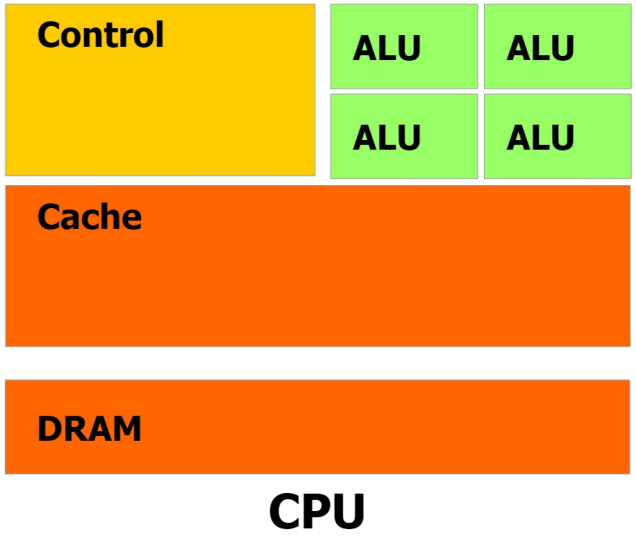
CPU
(Central Processing Unit)

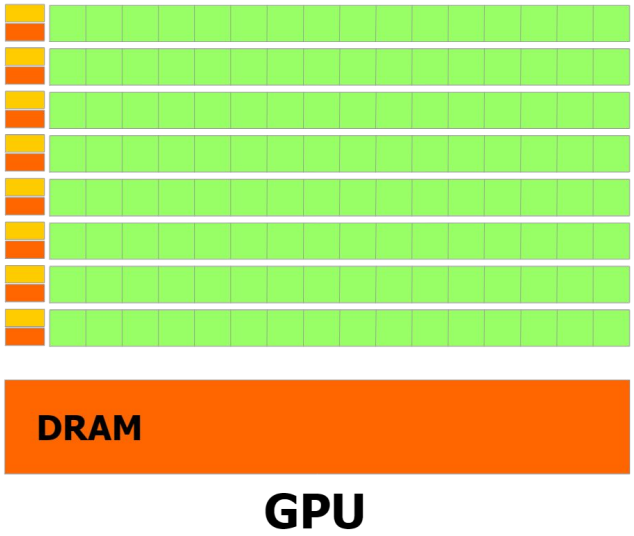
GPU
(Graphics Processing Unit)



Hardware para DL

- CPU vs GPU





ALU = Arithmetic Logic Unit (Unidade de Lógica Aritmética)

Hardware para DL

- CPU vs GPU

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$385	~540 GFLOPs FP32
GPU (NVIDIA RTX 2080 Ti)	3584	1.6 GHz	11 GB GDDR6	$1199	~13.4 TFLOPs FP32

Hardware para DL

- CPU vs GPU

  - CPU: excelente para lidar com tarefas sequenciais e diversas

  - GPU: excelente para lidar com tarefas paralelas e específicas

CPU:

GPU:

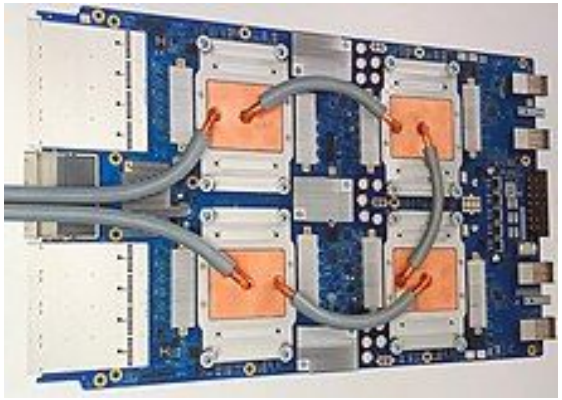
Hardware para DL

- TPU?

Hardware para DL

- TPU (Tensor Processing Unit)

    - ASIC (Application Specific Integrated Circuit): excelente para lidar com tarefas paralelas de aplicações bastante específicas



Hardware para DL

- CPU vs GPU vs TPU

	Cores	Clock Speed	Memory	Price	Speed
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	System RAM	$385	~540 GFLOPs FP32
GPU (NVIDIA RTX 2080 Ti)	3584	1.6 GHz	11 GB GDDR6	$1199	~13.4 TFLOPs FP32
TPU NVIDIA TITAN V	5120 CUDA, 640 Tensor	1.5 GHz	12GB HBM2	$2999	~14 TFLOPs FP32 ~112 TFLOP FP16
TPU Google Cloud TPU	?	?	64 GB HBM	$4.50 per hour	~180 TFLOP

Hardware para DL

- CPU vs GPU vs TPU

CPU:

GPU:

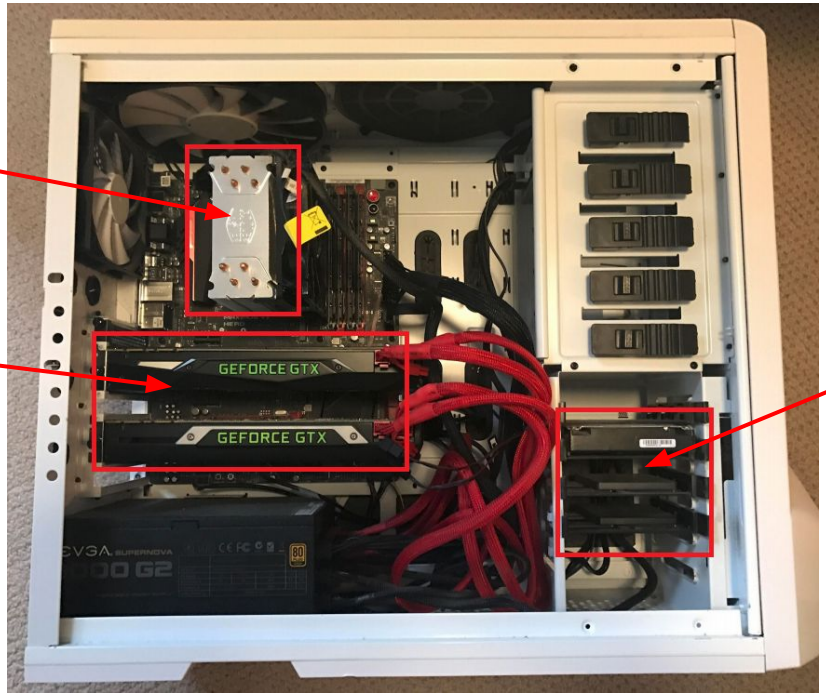
TPU:

Hardware para DL

- Gargalo da comunicação

Pré-processamento pode acontecer aqui

Modelo está aqui



Dados estão aqui

Sincronia é extremamente importante!

Software para DL

- Frameworks

  - Tensorflow (Google)
  - Pytorch (Facebook)
  - Caffe2 (Facebook)
  - MXNet (Amazon)
  - CNTK (Microsoft)
  - PaddlePaddle (Baidu)
  - … … …

Library	Rank	Overall	Github	Stack Overflow	Google Results
tensorflow	1	10.87	4.25	4.37	2.24
keras	2	1.93	0.61	0.83	0.48
caffe	3	1.86	1.00	0.30	0.55
theano	4	0.76	-0.16	0.36	0.55
pytorch	5	0.48	-0.20	-0.30	0.98
sonnet	6	0.43	-0.33	-0.36	1.12
mxnet	7	0.10	0.12	-0.31	0.28
torch	8	0.01	-0.15	-0.01	0.17
cntk	9	-0.02	0.10	-0.28	0.17
dlib	10	-0.60	-0.40	-0.22	0.02
caffe2	11	-0.67	-0.27	-0.36	-0.04
chainer	12	-0.70	-0.40	-0.23	-0.07
paddlepaddle	13	-0.83	-0.27	-0.37	-0.20
deeplearning4j	14	-0.89	-0.06	-0.32	-0.51
lasagne	15	-1.11	-0.38	-0.29	-0.44
bigdl	16	-1.13	-0.46	-0.37	-0.30
dynet	17	-1.25	-0.47	-0.37	-0.42
apache singa	18	-1.34	-0.50	-0.37	-0.47
nvidia digits	19	-1.39	-0.41	-0.35	-0.64
matconvnet	20	-1.41	-0.49	-0.35	-0.58
tflearn	21	-1.45	-0.23	-0.28	-0.94
nervana neon	22	-1.65	-0.39	-0.37	-0.89
opennn	23	-1.97	-0.53	-0.37	-1.07

Software para DL

- Frameworks

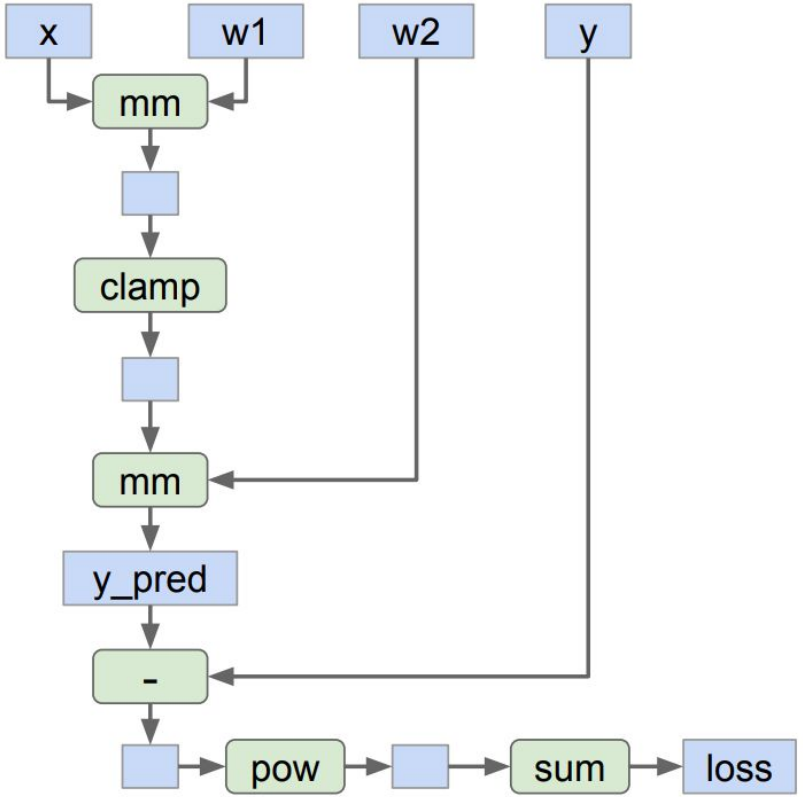
- Tensorflow (Google)
- Pytorch (Facebook)
- Caffe2 (Facebook)
- MXNet (Amazon)
- CNTK (Microsoft)
- PaddlePaddle (Baidu)
- … … …

Grafos computacionais dinâmicos vs estáticos

tf.keras vs torch.nn

Software para DL

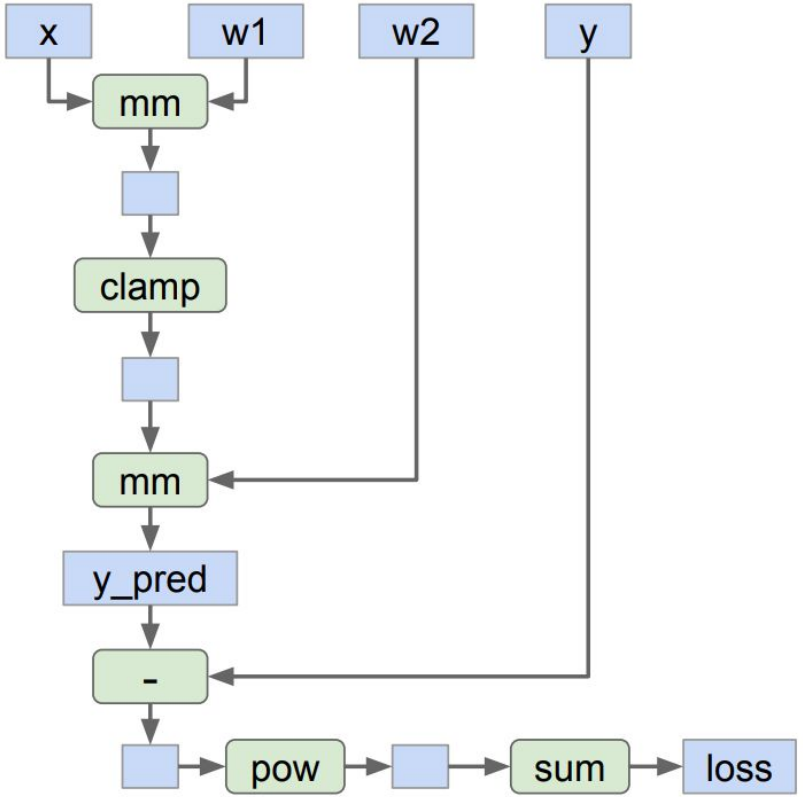
- Grafos computacionais



Software para DL

- Grafos computacionais

$$y_{pred} = clamp(xW_1)W_2$$
$$Loss = \sum (y_{pred} - y)^2$$



Software para DL

- Pytorch (grafos dinâmicos)

```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Software para DL

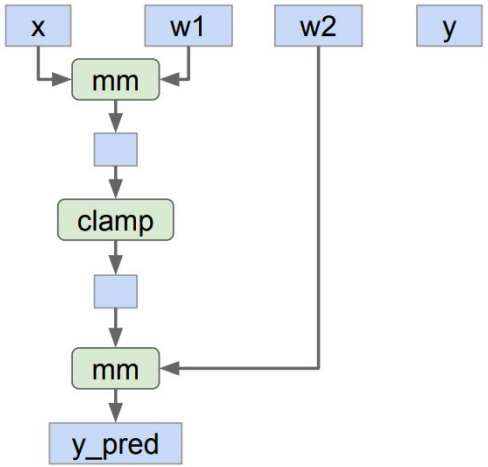
- Pytorch (grafos dinâmicos)


```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Create Tensor objects

Software para DL

- Pytorch (grafos dinâmicos)

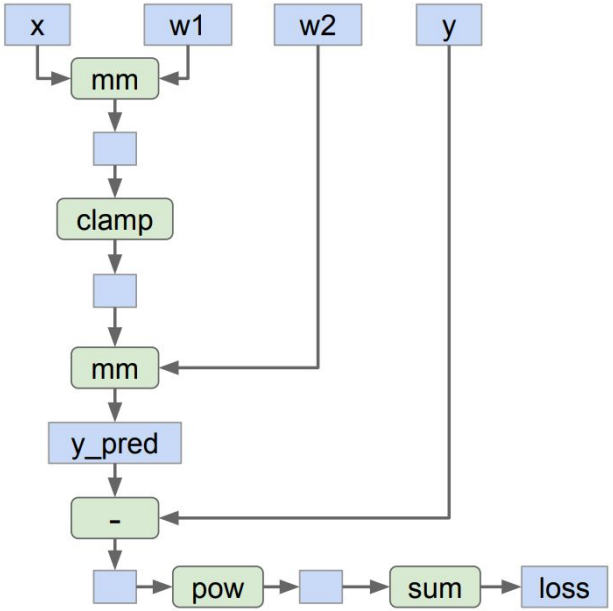



```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Build graph data structure AND perform computation

Software para DL

- Pytorch (grafos dinâmicos)

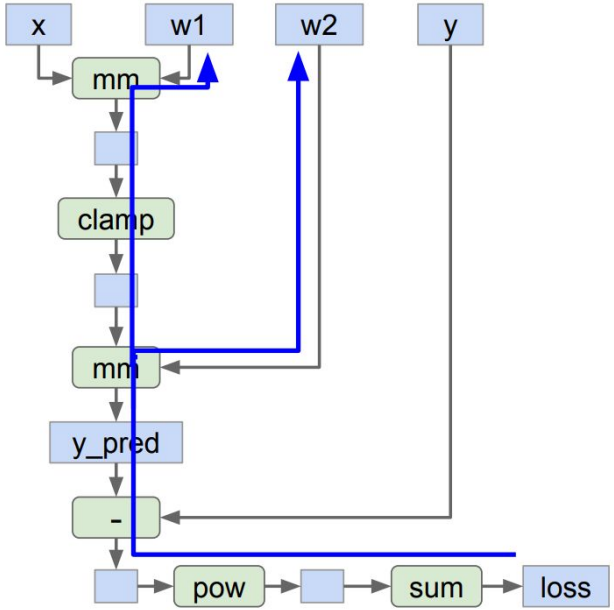



```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Build graph data structure AND perform computation

Software para DL

- Pytorch (grafos dinâmicos)




```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Search for path between loss and w1, w2
(for backprop) AND perform computation

Software para DL

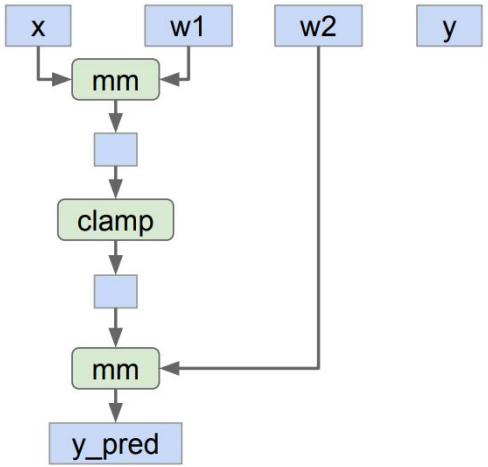
- Pytorch (grafos dinâmicos)


```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Throw away the graph, backprop path, and rebuild it from scratch on every iteration

Software para DL

- Pytorch (grafos dinâmicos)



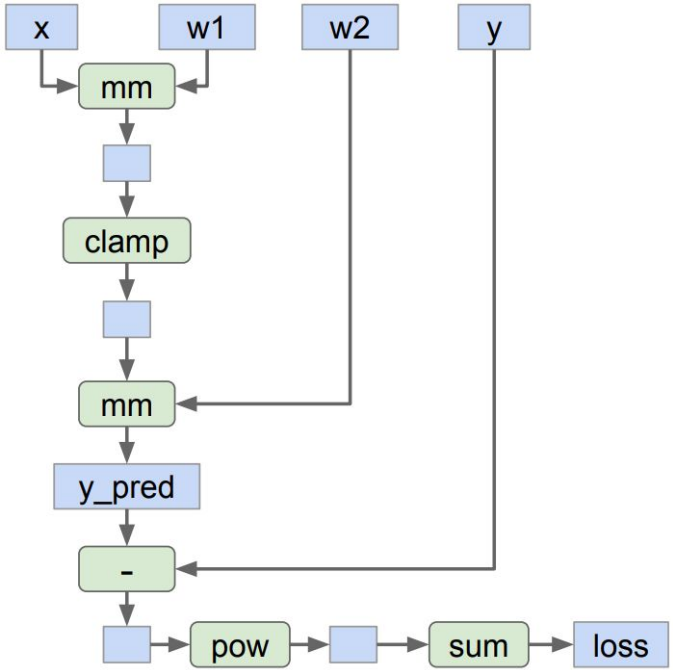

```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)
w1 = torch.randn(D_in, H, requires_grad=True)
w2 = torch.randn(H, D_out, requires_grad=True)

learning_rate = 1e-6
for t in range(500):
    y_pred = x.mm(w1).clamp(min=0).mm(w2)
    loss = (y_pred - y).pow(2).sum()

    loss.backward()
```

Build graph data structure AND perform computation

Software para DL

- Tensorflow 1.xx (grafos estáticos)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```



Software para DL

- Tensorflow 1.xx (grafos estáticos)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])
```

construção do grafo

```
with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

execução do grafo

Software para DL

- Tensorflow 2.0 (grafos dinâmicos)

```
N, D, H = 64, 1000, 100

x = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
y = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
w1 = tf.Variable(tf.random.uniform((D, H)))  # weights
w2 = tf.Variable(tf.random.uniform((H, D)))  # weights

with tf.GradientTape() as tape:
  h = tf.maximum(tf.matmul(x, w1), 0)
  y_pred = tf.matmul(h, w2)
  diff = y_pred - y
  loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
gradients = tape.gradient(loss, [w1, w2])
```

Software para DL

- Tensorflow 2.0 vs 1.xx

```
N, D, H = 64, 1000, 100

x = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
y = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
w1 = tf.Variable(tf.random.uniform((D, H)))  # weights
w2 = tf.Variable(tf.random.uniform((H, D)))  # weights
```

```
with tf.GradientTape() as tape:
  h = tf.maximum(tf.matmul(x, w1), 0)
  y_pred = tf.matmul(h, w2)
  diff = y_pred - y
  loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
gradients = tape.gradient(loss, [w1, w2])
```

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(D, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))
```

```
h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))

grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])
```

```
with tf.Session() as sess:
    values = {x: np.random.randn(N, D),
              w1: np.random.randn(D, H),
              w2: np.random.randn(H, D),
              y: np.random.randn(N, D),}
    out = sess.run([loss, grad_w1, grad_w2],
                   feed_dict=values)
    loss_val, grad_w1_val, grad_w2_val = out
```

Software para DL

- Estático
  - Vantagens:
    - velocidade
    - serialização
    - distribuição

- Dinâmico:
  - Vantagens:
    - flexibilidade

Software para DL

- Módulos de nível mais alto

  - torch.nn

```
import torch

N, D_in, H, D_out = 64, 1000, 100, 10
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)

model = torch.nn.Sequential(
          torch.nn.Linear(D_in, H),
          torch.nn.ReLU(),
          torch.nn.Linear(H, D_out))

learning_rate = 1e-2
for t in range(500):
    y_pred = model(x)
    loss = torch.nn.functional.mse_loss(y_pred, y)

    loss.backward()

    with torch.no_grad():
        for param in model.parameters():
            param -= learning_rate * param.grad
    model.zero_grad()
```

Software para DL

- Módulos de nível mais alto

  - tf.keras

```
N, D, H = 64, 1000, 100

x = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
y = tf.convert_to_tensor(np.random.randn(N, D), np.float32)
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(H, input_shape=(D,),
                                activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(D))
optimizer = tf.optimizers.SGD(1e-1)
model.compile(loss=tf.keras.losses.MeanSquaredError(),
              optimizer=optimizer)

history = model.fit(x, y, epochs=50, batch_size=N)
```

Software para DL

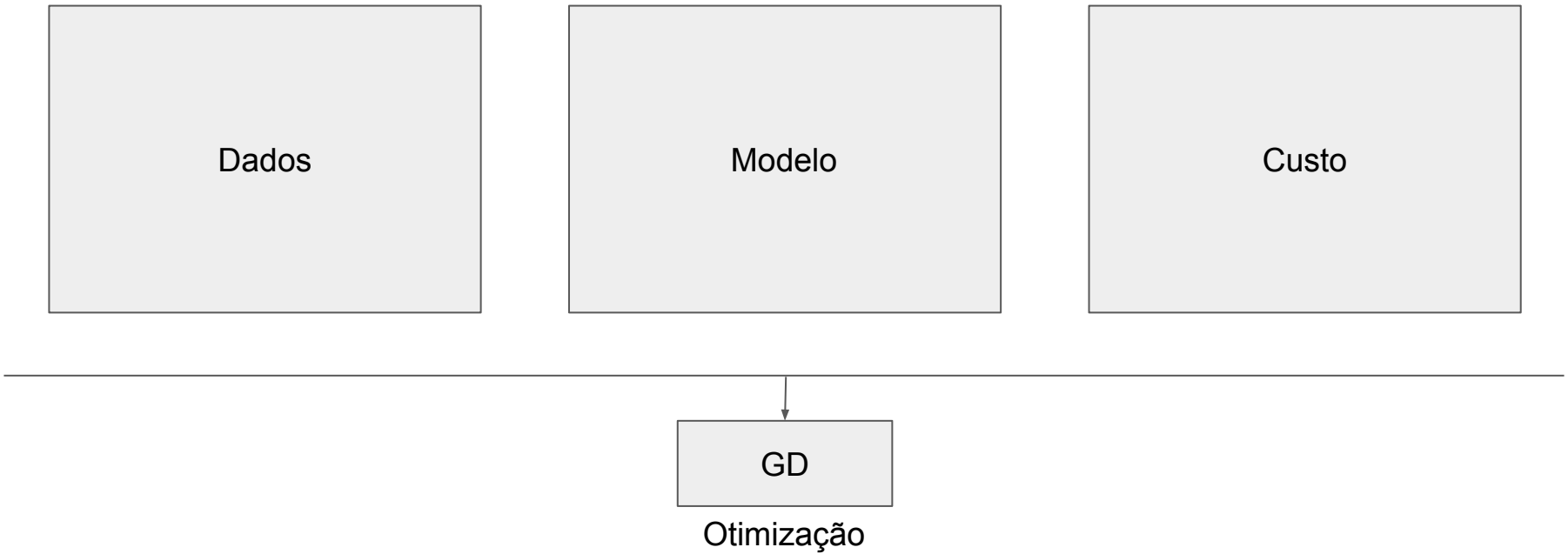
- Aulas de hands-on serão em **tf.keras**

    - Documentação:

        - https://www.tensorflow.org/guide/keras
        - https://keras.io/

Ciclo vicioso de ML

- Onde estamos...



Ciclo vicioso de ML

- Onde estamos...

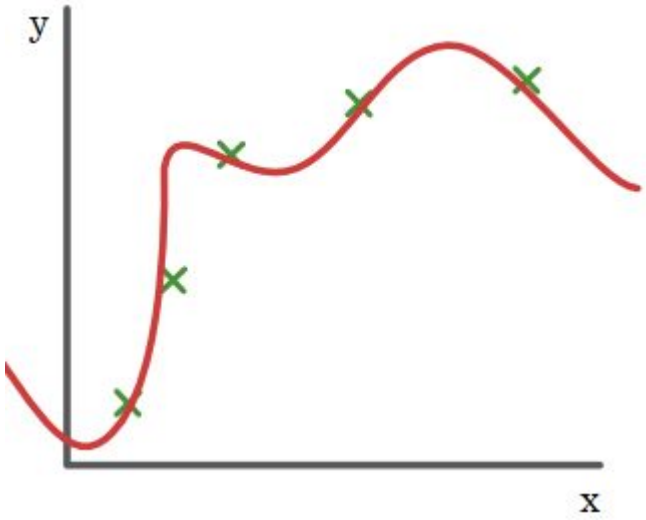
$$while\ \mathcal{L}(\Theta) > threshold:$$
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta)$$

Ciclo vicioso de ML

- Problemas

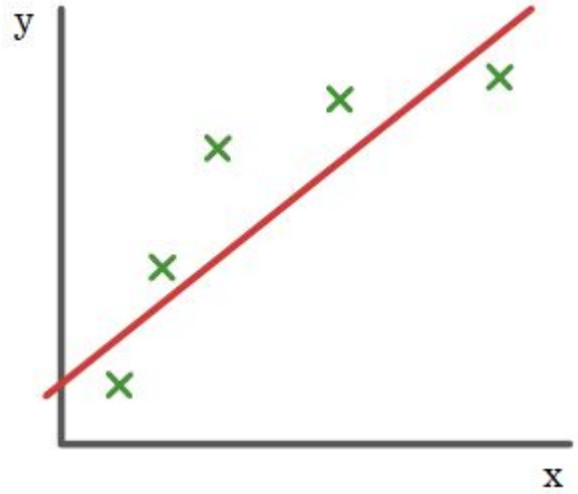
Ciclo vicioso de ML

- Threshold muito pequeno: overfitting



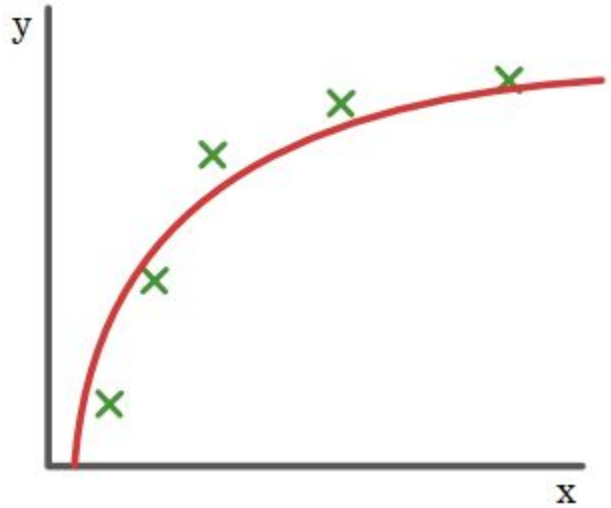
Ciclo vicioso de ML

- Threshold muito grande: underfitting



Ciclo vicioso de ML

- Threshold ideal



Ciclo vicioso de ML

- Como achar esse threshold ideal?

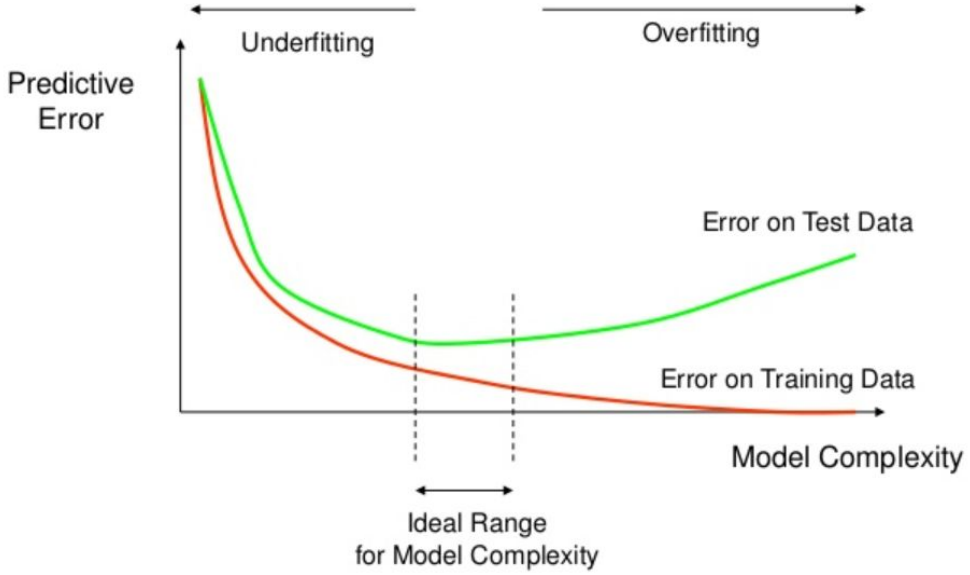
Ciclo vicioso de ML

- Como achar esse threshold ideal?

    - Divisão do dataset em treino/teste (train/test split)!

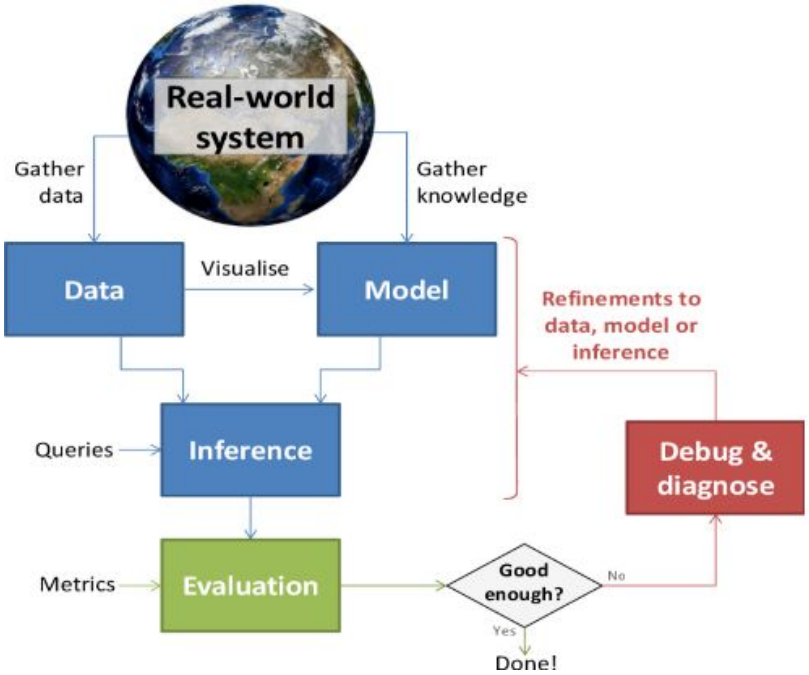
Ciclo vicioso de ML

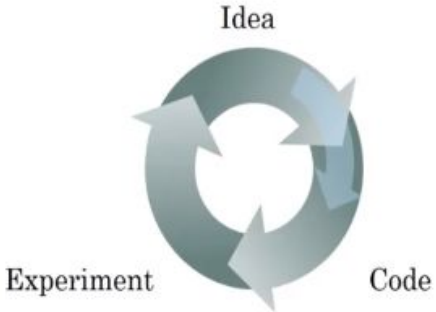
- Como achar esse threshold ideal?
  - Divisão do dataset em treino/teste (train/test split)!



Ciclo vicioso de ML

- O ciclo tentador...





Ciclo vicioso de ML

- Pronto?

Ciclo vicioso de ML

- Pronto? Não! -> Leakage

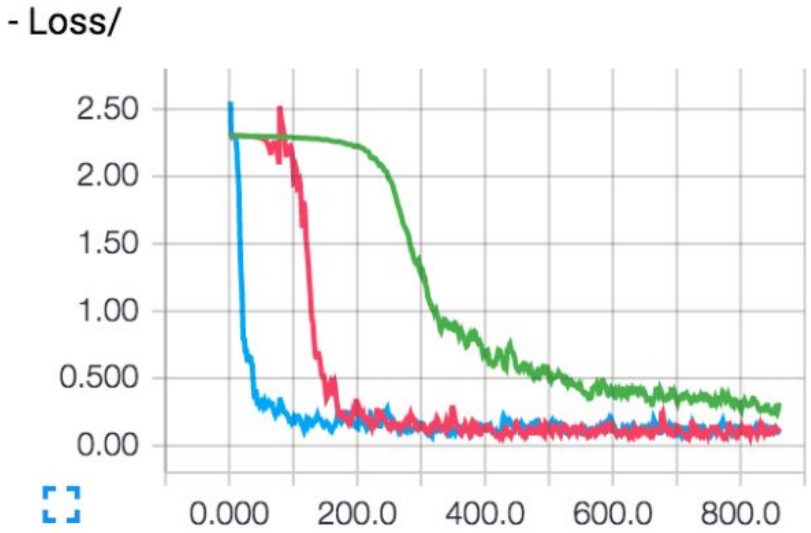
    - Se o professor sempre repete a mesma prova, o aluno acaba decorando a prova!

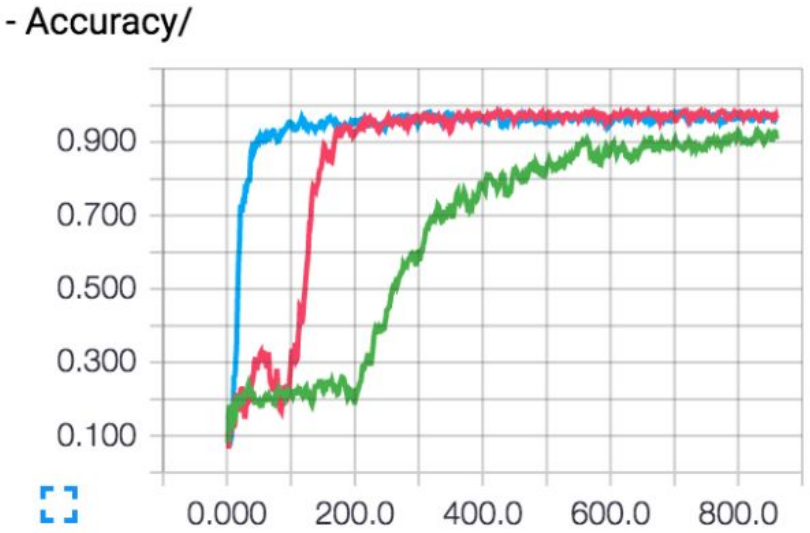
Ciclo vicioso de ML

- Divisão treino/validação/teste

Ciclo vicioso de ML

- Diferença entre custo e métrica

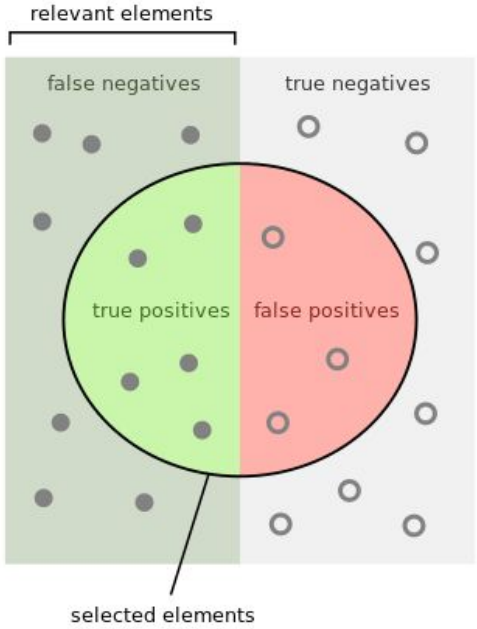


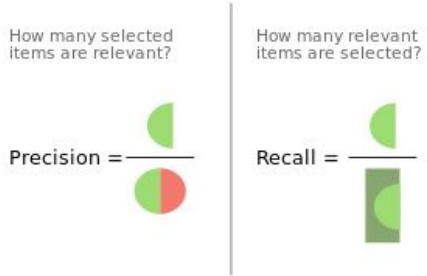


Ciclo vicioso de ML

- Diferentes tipos de métricas

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)





Ciclo vicioso de ML

- Análises necessárias:

    - Preciso treinar mais?
    - Preciso de um novo modelo?
    - Preciso de mais dados?

No próximo episódio...

- Aula hands-on