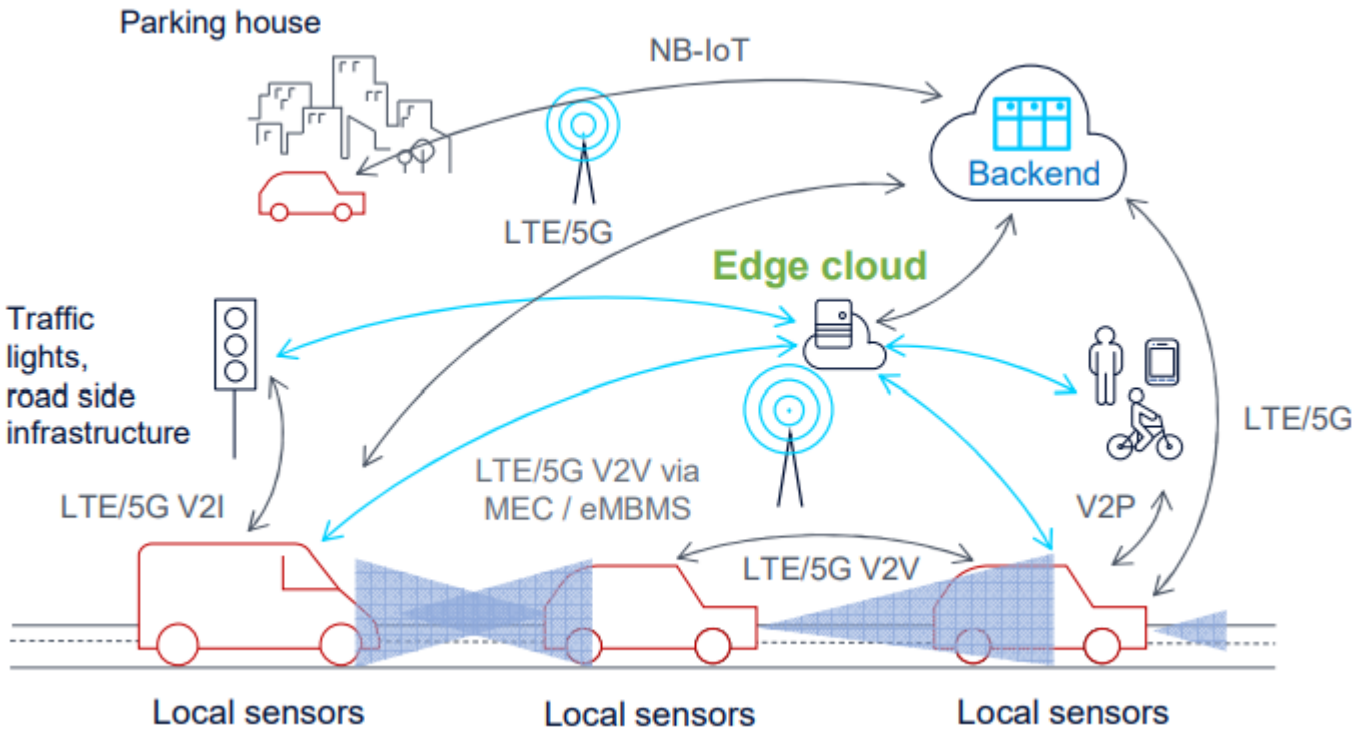
Sistemas auto-organizados: dados, aprendizagem e decisões

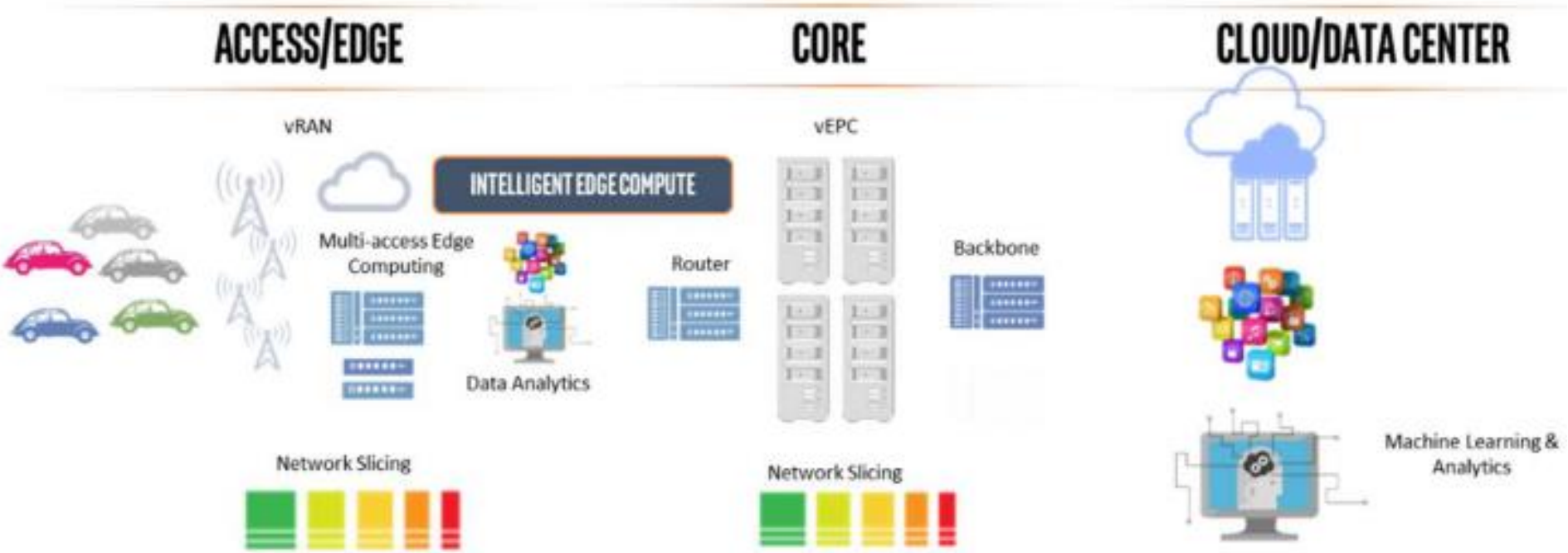
Mestrado em Engenharia de Computadores e Telemática

2023/2024

Casos de uso e dados



Onde processar os dados?

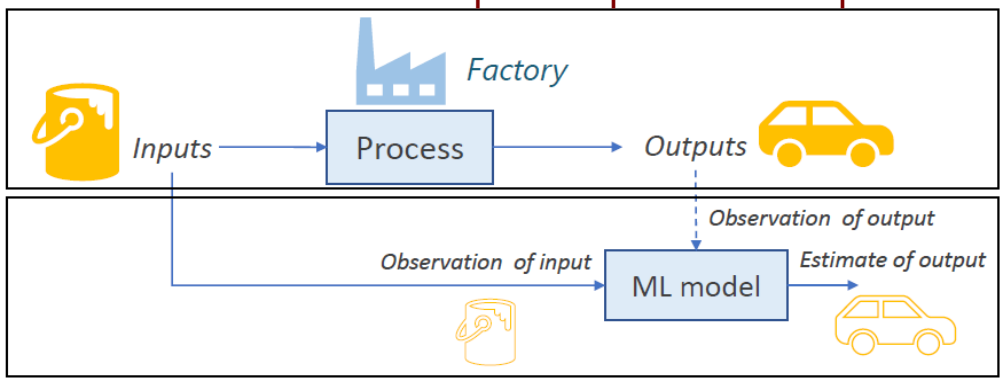


Análise de dados

- Processando dados
- Tome alguma decisão com os dados
- Decisões de rede com dados de rede e serviços
  - Dar mais largura de banda?
  - Atribuir alguma fila especial para reduzir o atraso?
- Decisões do usuário ou decisões de elementos
  - Obstáculo no lugar?
  - Robô chuta a bola para a direita?
- Decisões de rede com dados dos usuários
  - Preveja transferências com localização e velocidade
  - Mova o conteúdo da CDN para o ponto de acesso mais próximo dos usuários
  - A ambulância está a caminho com requisitos de rede
- Decisões do usuário com dados de rede
  - Escolha um caminho com ótima conectividade para jogos ou vídeo
  - Escolha um local para realidade virtual e aumentada remota

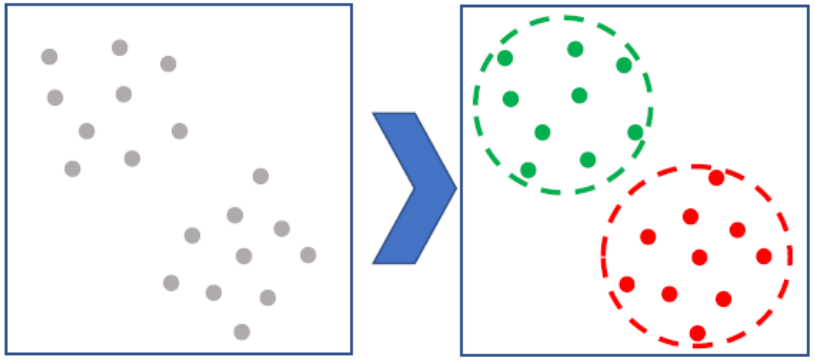
Aprendizado de máquina

- Uma definição pragmática:
  - Coleção de algoritmos e modelos estatísticos (métodos) para máquinas realizarem tarefas automatizadas ***baseado na observação de entradas e/ou saídas de um processo***
- O objetivo do Machine Learning é produzir uma estimativa ou classificação dado um conjunto de valores de entrada.
- Muitas vezes distinguimos:
  - Método ML: o mecanismo para treinar um modelo (rede neural, máquina de vetores de suporte, etc.)
  - Modelo de ML: uma instância do método treinado para replicar o comportamento do processo alvo



Tipos de aprendizagem

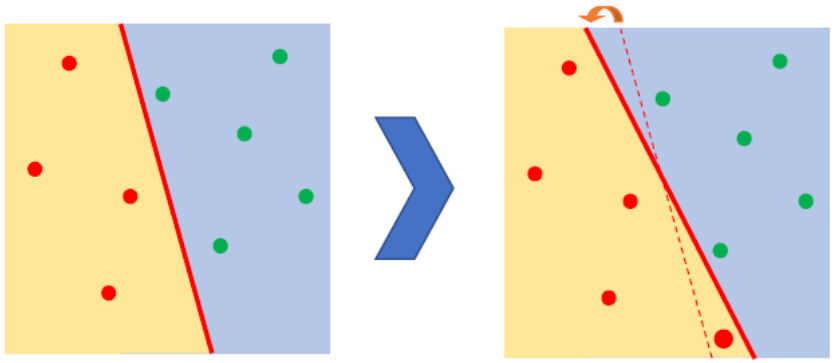
- Supervisionado: o modelo é treinado com um conjunto de dados do processo alvo
  - Ao treinar para uma tarefa de classificação, o conjunto de dados históricos deve conter a verdade básica - a classe real de uma determinada amostra
- Não supervisionado: classificação ou regressão independe de conhecimento prévio



Unsupervised (classes are created by, e.g., finding clusters of similar data points)

Ground Truth
necessary for training

Historical Dataset		
Feature 1	**Feature 2**	**Class**
A	1	X
B	1	X
C	2	Y



Supervised (classification depends on historical inputs)

Aprendizagem Supervisionada

- Estágio de treinamento

  Um conjunto de dados com dados de entrada e saída correspondente

  Os dados de entrada são pré-processados para identificar e/ou extrair recursos relevantes

  Os dados do recurso são inseridos no método ML, normalmente um conjunto de recursos (por exemplo, média, mediana, desenvolvimento padrão)

  Às vezes, um conjunto cego de recursos é produzido e, então, apenas os mais relevantes são selecionados (por exemplo, árvores de decisão).

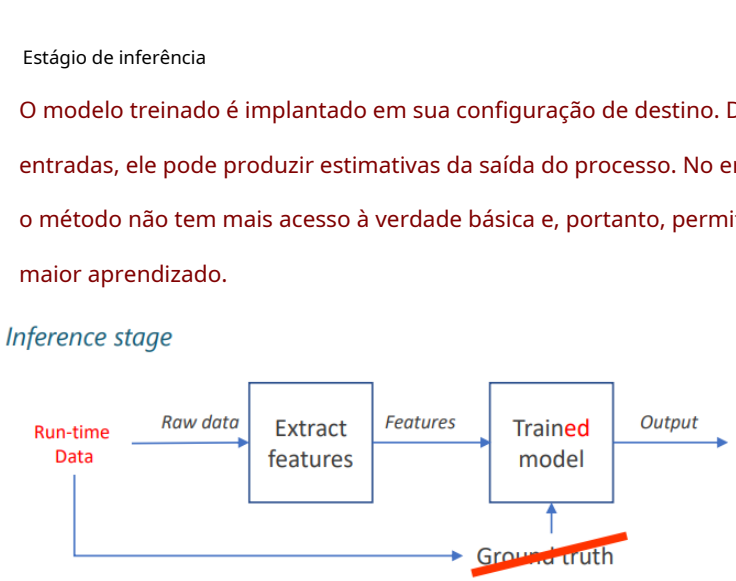
  Para cada conjunto de entradas, o método produz uma estimativa com probabilidade de erro.

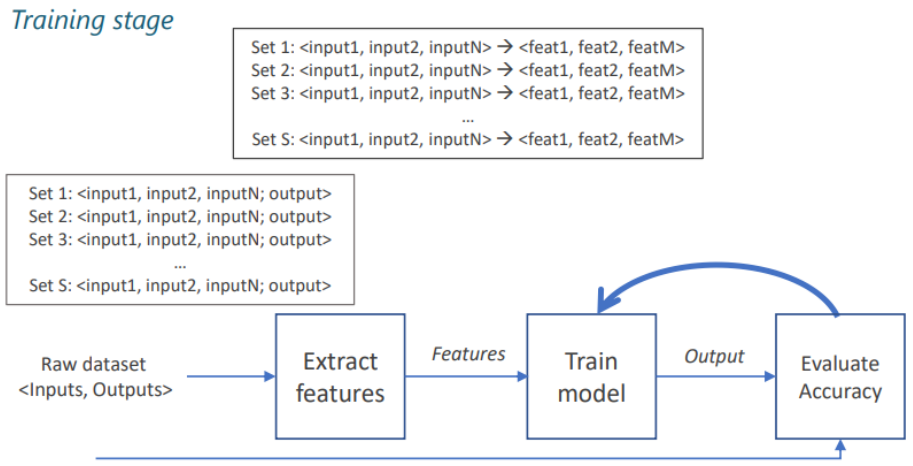
  O método compara a estimativa com o resultado real do processo (a Verdade Fundamental) e atualiza os processos internos do modelo para melhorar a precisão das estimativas.

  O processo é repetido até que o desempenho do método esteja dentro dos limites aceitáveis.

- Estágio de inferência

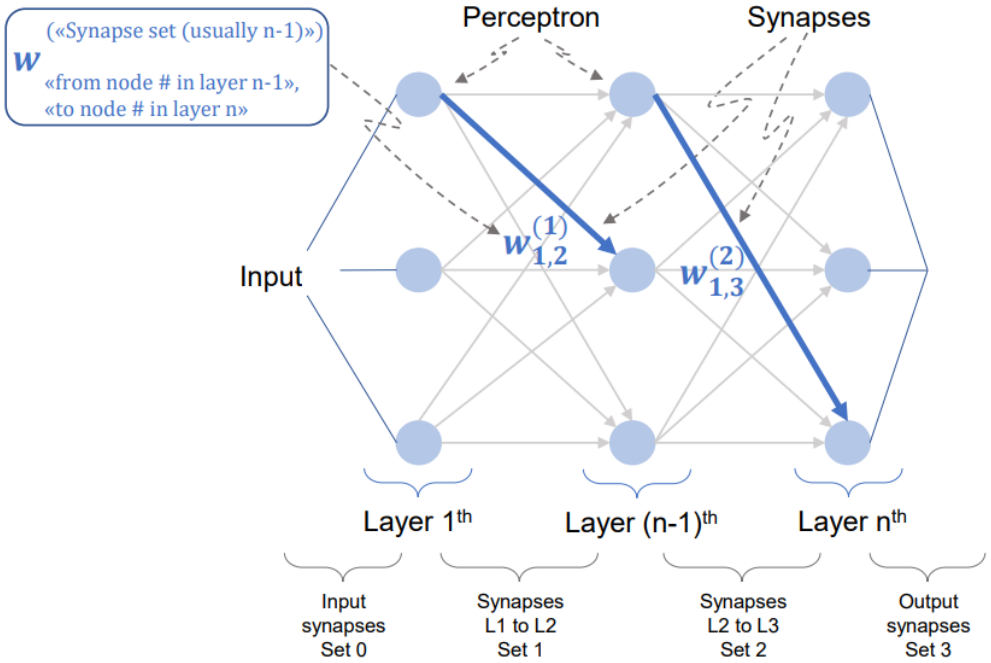
  O modelo treinado é implantado em sua configuração de destino. Dadas as entradas, ele pode produzir estimativas da saída do processo. No entanto, o método não tem mais acesso à verdade básica e, portanto, permite um maior aprendizado.

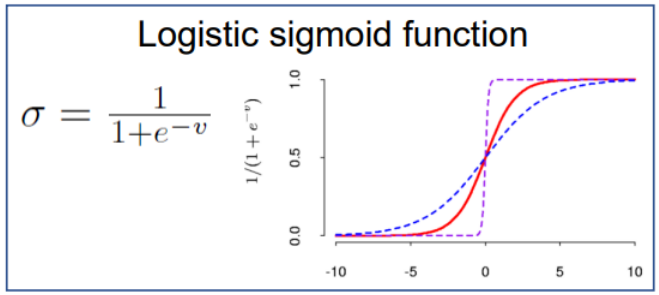




Redes neurais

- Um dos métodos de ML de maior sucesso
- Blocos de construção: Perceptron e Sinapse
- Perceptron: normalmente uma função que mapeia todo o intervalo natural em um intervalo limitado ([0,1] ou [-1,1])
  - Exemplo: função sigmóide logística, função ReLU, tanh, softmax, etc.
- Sinapses: conexões de perceptrons da camada (n-1)th com perceptrons da camada nth, cada um aplicando um peso ao valor transmitido
- Treinar Redes Neurais consiste principalmente em encontrar os pesos dessas sinapses





Aprendizagem por Reforço

- Tipo de técnica de aprendizado de máquina que permite que um agente aprenda em um ambiente interativo por tentativa e erro, usando feedback de suas próprias ações e experiências.

1. Meio ambiente —Mundo físico em que o agente opera

2. Estado —Situação atual do agente

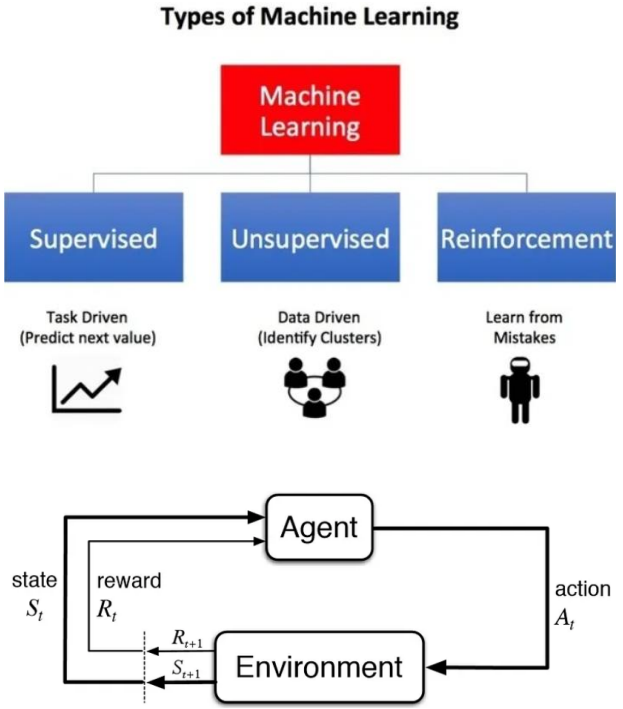
3. Recompensa —Feedback do meio ambiente

4. Política —Método para mapear o estado do agente para ações

5. Valor —Recompensa futura que um agente receberia ao realizar uma ação em um determinado estado

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

- **Q-aprendizagem:** atualiza valores Q que denotam o valor da ação executada *a* no estado *é*. A seguinte regra de atualização de valor é o núcleo do algoritmo Q-learning.



Exemplo de recompensa: melhor caminho com recursos: largura de banda do caminho / comprimento do caminho Taxa de aprendizado e fator de desconto:]0 1[

$$reward = \frac{\frac{100.0}{|P|} \times \sum_{l}^{l \in P} R_l}{E_{pl} \times K_{spl}}$$

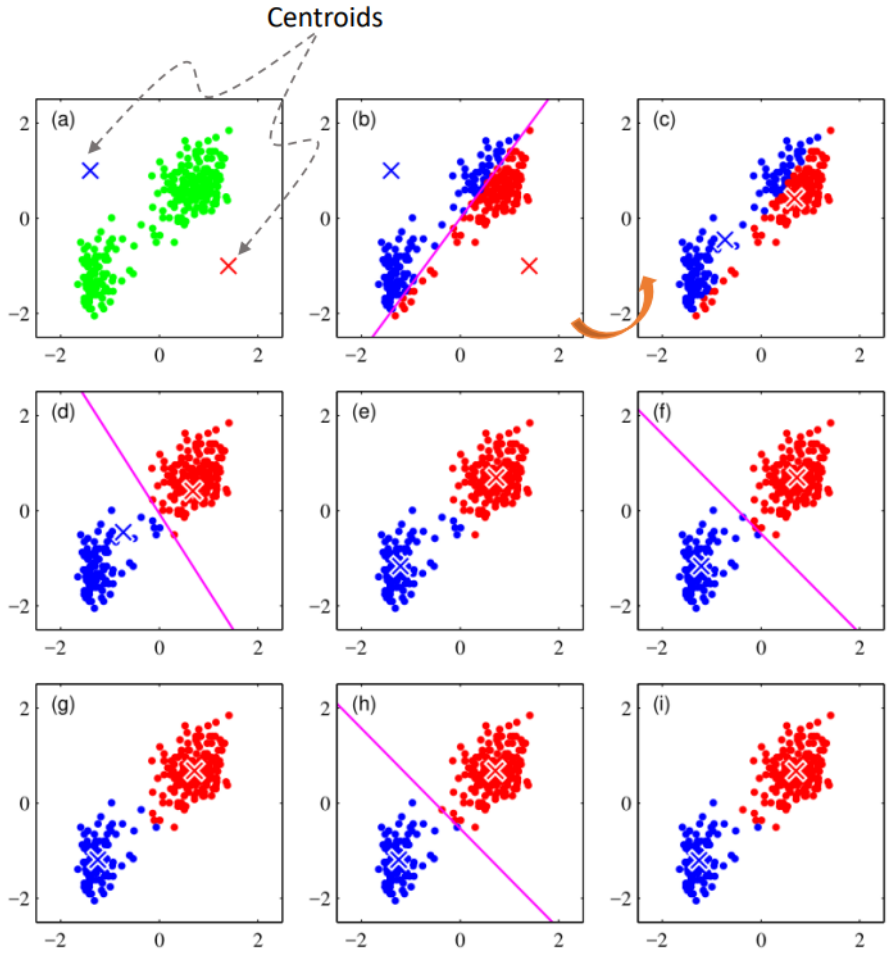
Jogos de computador (exemplo pacman), automação de robôs (RL é usado para permitir que o robô crie um sistema de controle adaptativo eficiente para si mesmo, que aprende com sua própria experiência e comportamento**),...**

Aprendizagem não supervisionada: K-significa

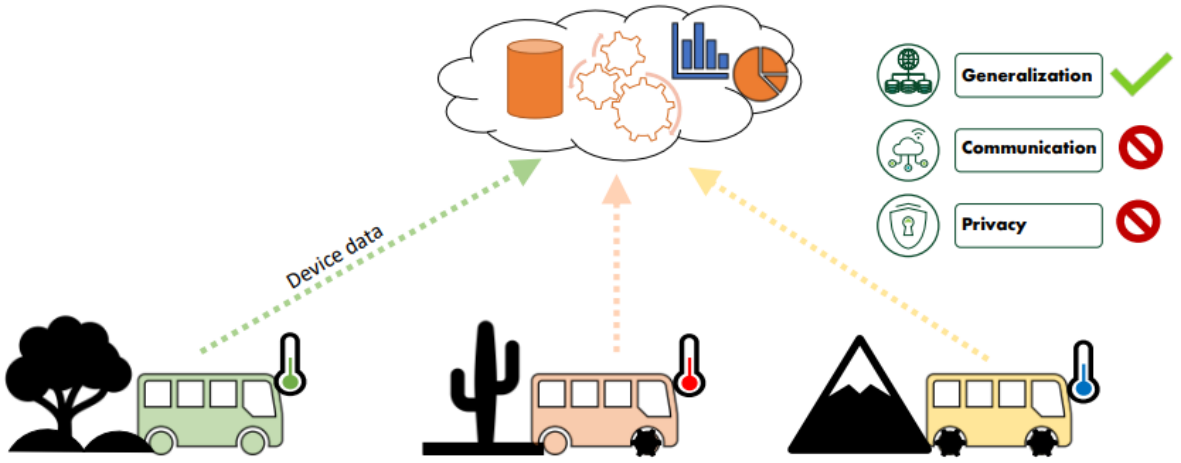
- Centróide: ponto sem dados que indica o centro do cluster conforme identificado por K-means

- Operação:

  1. Implante N centróides aleatoriamente (N proporcional ao número de classes esperadas)
  2. Atribuir pontos de dados aleatoriamente às classes
  3. Repita iterativamente
     1. Calcular o centro de gravidade de cada classe;
     2. O centróide é reposicionado nesse centro de gravidade
     3. Atualizar limite
  4. Pare quando as atualizações se tornarem insignificantes



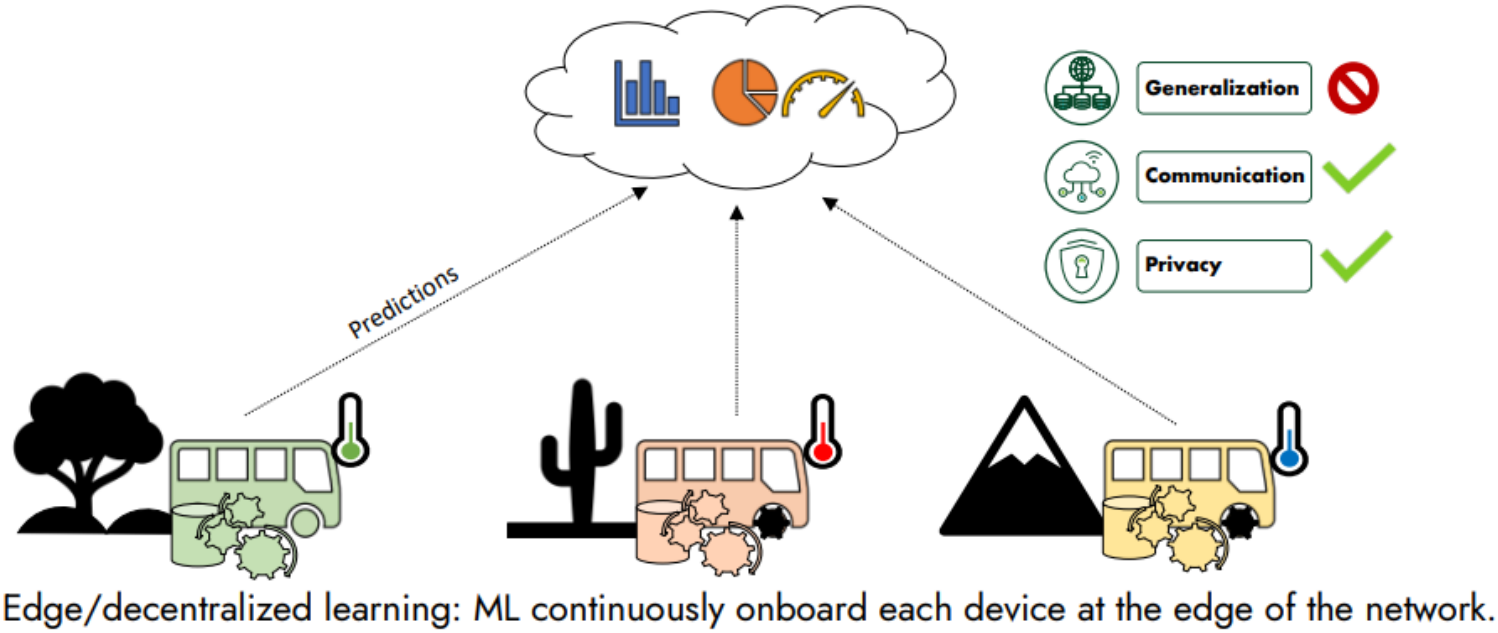
Aprendizagem: centralizada



Traditional centralized learning – ML runs in the cloud, gathering info from all connected devices and sending back a model.

- O modelo pode generalizar com base em dados de um grupo de dispositivos e, assim, funcionar instantaneamente com outros dispositivos compatíveis
- Os dados podem explicar todas as variações nos dispositivos e no seu ambiente
- Conectividade - os dados devem ser transmitidos através de uma conexão estável
- Largura de banda – por exemplo, uma nova subestação elétrica poderia gerar 5 GB/s
- Latência – aplicações em tempo real, por exemplo, automação, requerem latência muito baixa
- Privacidade – dados operacionais confidenciais devem permanecer no local

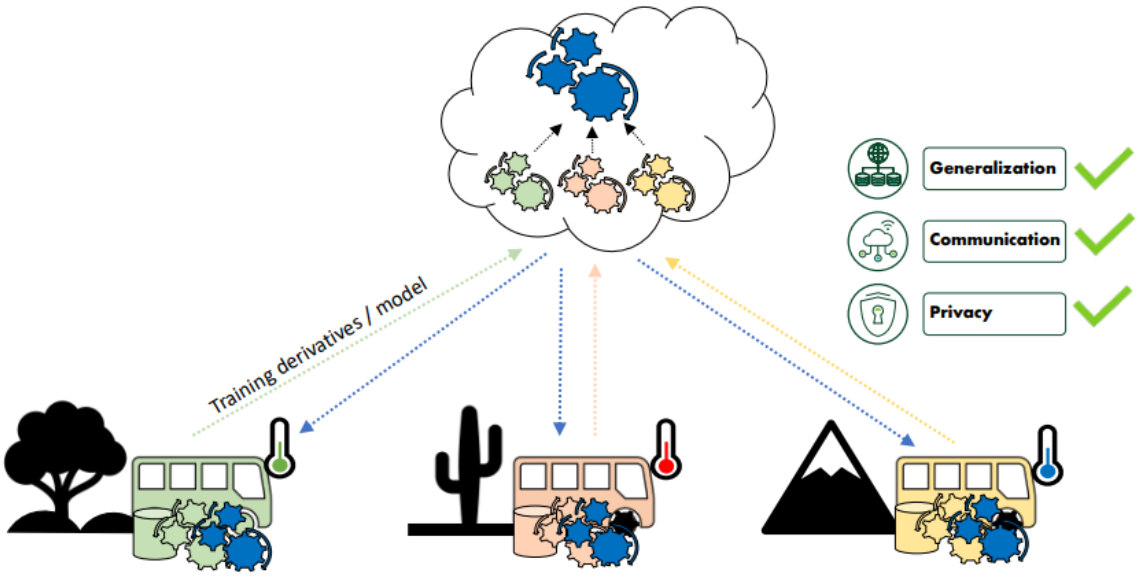
Aprendizagem: descentralizada



Edge/decentralized learning: ML continuously onboard each device at the edge of the network.

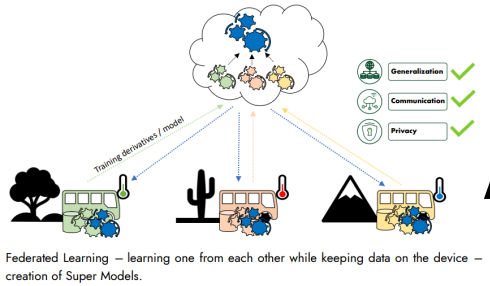
- ML executado no local, integrado em cada dispositivo conectado. Ao treinar continuamente o modelo de ML em streaming de dados, os dispositivos aprendem um modelo individual para seu ambiente.
- Cada modelo só precisa ser capaz de explicar o que é normal para si mesmo e não como isso varia em comparação com todos os outros dispositivos.
- Os modelos adaptam-se às mudanças ao longo do tempo, a aprendizagem não é limitada pela ligação à Internet e nenhuma informação confidencial precisa de ser transferida para a nuvem.
- Não é possível obter uma visão global e aprender

Aprendizagem: federado



Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

- Técnica de ML para treinar algoritmos em dispositivos de borda descentralizados enquanto mantém amostras de dados localmente
- O Google começou como o principal player
- Objetivo de treinar modelos de ML em bilhões de telefones celulares, respeitando a privacidade dos usuários
  - Envie apenas frações de resultados de treinamento, ou seja, derivados de treinamento, para a
  - nuvem Nunca armazene nada no dispositivo



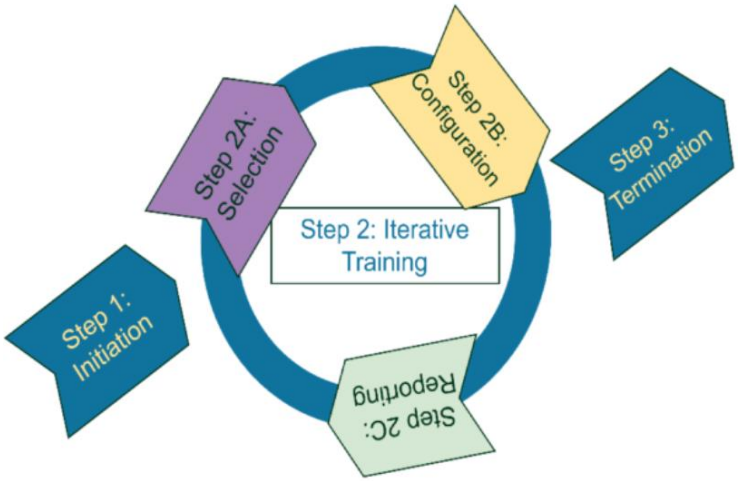
Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

Aprendizagem: federado

- Quando coletados na nuvem, os resultados parciais do treinamento podem ser montados em um novo supermodelo que, na próxima etapa, pode ser enviado de volta aos dispositivos
  - **Estrutura de código aberto do Google TensorFlow Federated**
- Inspeção de modelo – avaliação do comportamento do dispositivo através de seu modelo
- Comparação de modelos – comparando modelos na nuvem para encontrar valores discrepantes, supermodelos
- Aprendizado robusto – o aprendizado pode continuar mesmo se a conexão com a nuvem for perdida
- Inicialização personalizada – novos dispositivos podem começar com um modelo de um dispositivo semelhante, em vez de um supermodelo geral

Aprendizagem federada: iterativa

- FL emprega um método iterativo contendo múltiplas trocas cliente-servidor: rodada de aprendizagem federada
  - Difundir o estado atual/atualizado do modelo global para os nós contribuintes (participantes)
  - Treine os modelos locais nesses nós para produzir certas atualizações potenciais do modelo dos nós
  - Processar e agregar as atualizações dos nós locais em uma atualização global agregada para que o modelo central possa ser atualizado adequadamente
- O servidor FL é usado para este processamento e agregação de atualizações locais para atualizações globais
  - O treinamento local é realizado por nós locais em relação aos comandos do servidor FL



Aprendizagem: federado

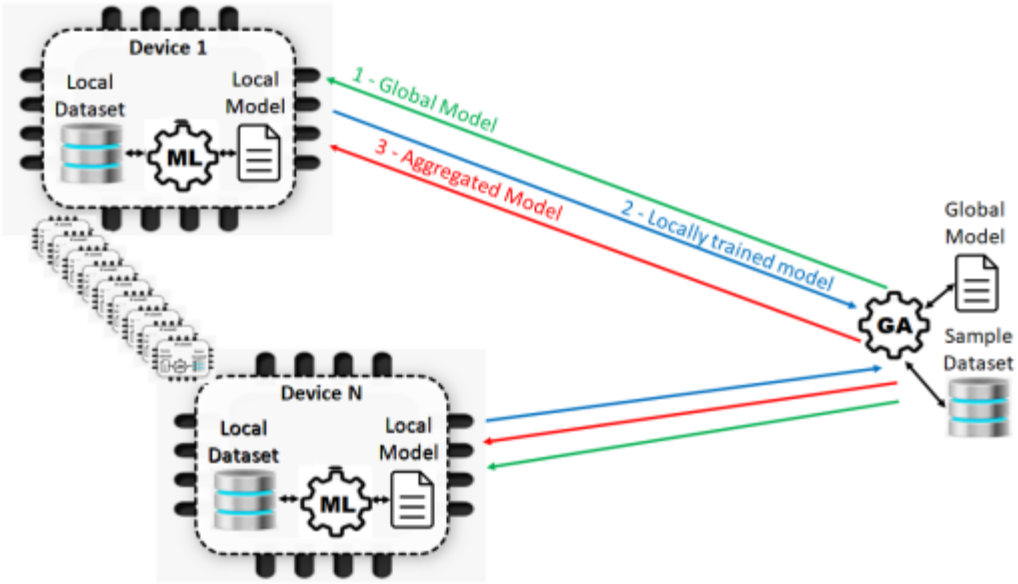
- A abordagem FL permite o processamento em massa de dados de forma distribuída
- Pode seguir uma arquitetura cliente-servidor

  Servidor envia o modelo a ser criado para os clientes (1 – linhas verdes)

  Os resultados da computação local são enviados ao servidor, que os agrega no modelo global (2 – linhas azuis) Retorna o novo modelo agregado aos clientes (3 – linhas vermelhas)

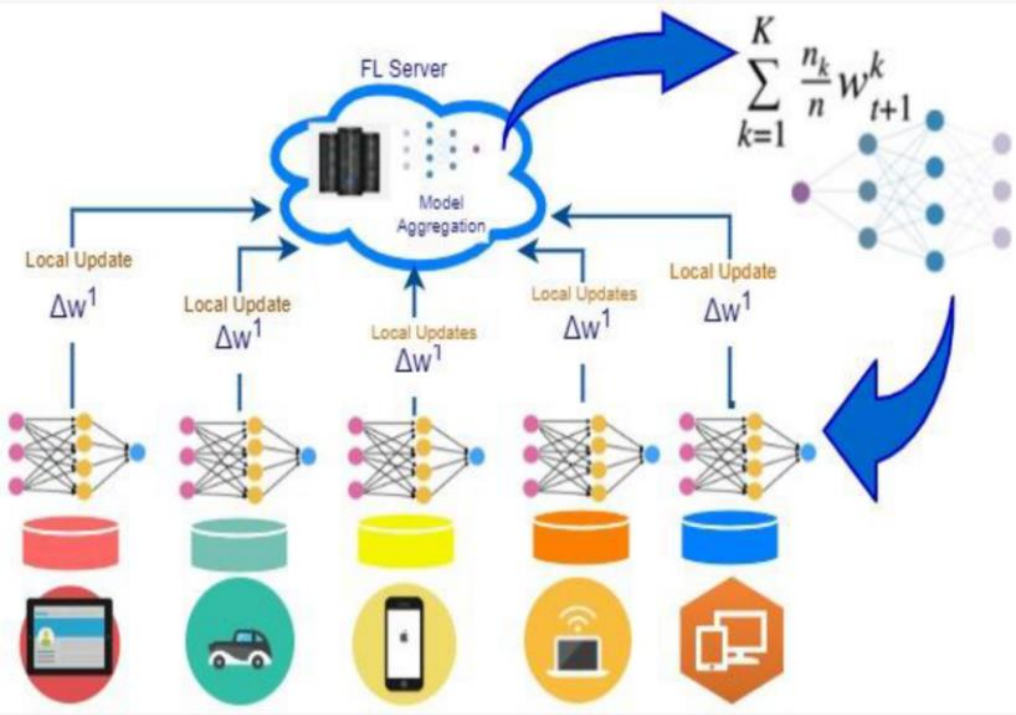
  Essa iteração, denominada rodada de aprendizagem federada (FLR), ocorre até que algum critério de parada seja alcançado, como convergência do modelo ou número máximo de iterações alcançado.

  Dispositivos Edge enviam apenas informações de seus modelos locais (parâmetros, hiperparâmetros (antes do treinamento), pesos, etc).



Aprendizagem: federado

- O aprendizado federado distribui o aprendizado profundo, eliminando a necessidade de agrupar os dados em um único local
- No FL, o modelo é treinado em diferentes locais em inúmeras iterações



Aprendizagem: agregação de modelos

- A agregação eficaz de modelos distribuídos entre dispositivos é essencial para a criação de um modelo global generalizado.
- Sua eficiência afeta a precisão, o tempo de convergência, o número de rodadas e a sobrecarga da rede.
- Descida gradiente estocástica federada (SGD): usa uma única instância do conjunto de dados para realizar o treinamento local no cliente por rodada de comunicação. O SGD requer um número substancial de rodadas de treinamento para produzir resultados confiáveis modelos. Este algoritmo é a base do aprendizado federado.
- O algoritmo FedAvg começa no SGD, mas cada cliente realiza localmente um trem usando os dados locais no modelo atual com múltiplas etapas de SGD antes de enviar os modelos de volta ao servidor para agregação

FedAvg reduz a sobrecarga de comunicação necessária para fazer upload e download do modelo FL

Exige que os clientes realizem cálculos mais totais durante o treinamento.

Época local:uma passagem completa do conjunto de dados de treinamento por meio o algoritmo

Algorithm 1 Algorithm FedAvg. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and n is the learning rate [3].

function SERVER-SIDE:
  initialize w0
  for each round $t = 1,2,...$ **do**
    $m \leftarrow \max(C \cdot k,1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1} \leftarrow \text{ClientUpdate}(k, wt)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for
end function

function CLIENT-SIDE:
  ClientUpdate(k,w): // Run on client k
  $B \leftarrow$ (split P_k into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in B$ **do**
      $w \leftarrow w - \eta \nabla l\,(w; b)$
    end for
  end for
  return w to server
end function

Aprendizagem: agregação de modelos

- Média Federada Tolerante a Falhas: capacidade de um sistema de computação continuar funcionando em caso de falha

  Pode tolerar que alguns nós fiquem offline durante a agregação segura

- Média Q-Federada: repondere o objetivo para alcançar justiça no modelo global

  Dá pesos maiores a dispositivos com baixo desempenho A

  distribuição de precisão da rede torna-se mais uniforme

  Fk(·) elevado a (q+1), q é um parâmetro que ajusta a quantidade de justiça a ser imposta.

$$\min_w f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)$$

- Otimização Federada: usa um otimizador de cliente durante as múltiplas épocas de treinamento e um otimizador de servidor durante a agregação do modelo

  ADAGRAD, ADAM e Yogi

Algorithm 1 FEDOPT

Input: x_0, CLIENTOPT, SERVEROPT
for $t = 0, \cdots, T-1$ **do**
  Sample a subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = \text{CLIENTOPT}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $x_{t+1} = \text{SERVEROPT}(x_t, -\Delta_t, \eta, t)$

Algorithm 2 FEDADAGRAD, FEDYOGI, and FEDADAM

Initialization: $x_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ **(FEDADAGRAD)**
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$

TensorFlow Federado

- Estrutura de código aberto para experimentar aprendizado de máquina e outros cálculos em dados descentralizados.
- Simulação local de cálculos descentralizados para todos os usuários do TensorFlow.
  - Arquitetura de modelo de ML de nossa escolha
  - Treine-o localmente em dados de todos os usuários
- Versão do conjunto de dados NIST que foi processado pelo projeto Leaf para separar os dígitos escritos por cada voluntário.

```python
# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
  dataset = source.create_tf_dataset_for_client(source.client_ids[n])
  return mnist.keras_dataset_from_emnist(dataset).repeat(10).batch(20)

# Wrap a Keras model for use with TFF.
def model_fn():
  return tff.learning.from_compiled_keras_model(
      mnist.create_simple_keras_model(), sample_batch)

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.build_federated_averaging_process(model_fn)
state = trainer.initialize()
for _ in range(5):
  state, metrics = trainer.next(state, train_data)
  print (metrics.loss)
```

federated_learning_for_image_classification.py hosted with ❤ by GitHub

TensorFlow Federado

• Treinar um modelo de ML com aprendizagem federada é um exemplo de computação federada

• Avaliar isso com base em dados descentralizados é outra

  – Conjunto de sensores que capturam leituras de temperatura

  – Calcule a temperatura média entre esses sensores

• Cada cliente calcula sua contribuição local

• Coordenador centralizado agrega todas as contribuições.

```
1  @tff.federated_computation(READINGS_TYPE)
2  def get_average_temperature(sensor_readings):
3    return tff.federated_average(sensor_readings)
```

get_average_temperature.py hosted with ❤ by GitHub

Fabaixo: Uma estrutura de aprendizagem federada amigável (http://flower.dev/)

- Estrutura de código aberto para experimentar aprendizado de máquina e outros cálculos em dados descentralizados.
- Capaz de usar em contêineres em uma estrutura federada, FedFramework

Containers:
/contêineres/lista
/containers/create/server /
containers/create/client /
containers/start
/contêineres/parar
/contêineres/remover

Listar contêineres:

- **http://10.0.22.37:8000/containers/list**

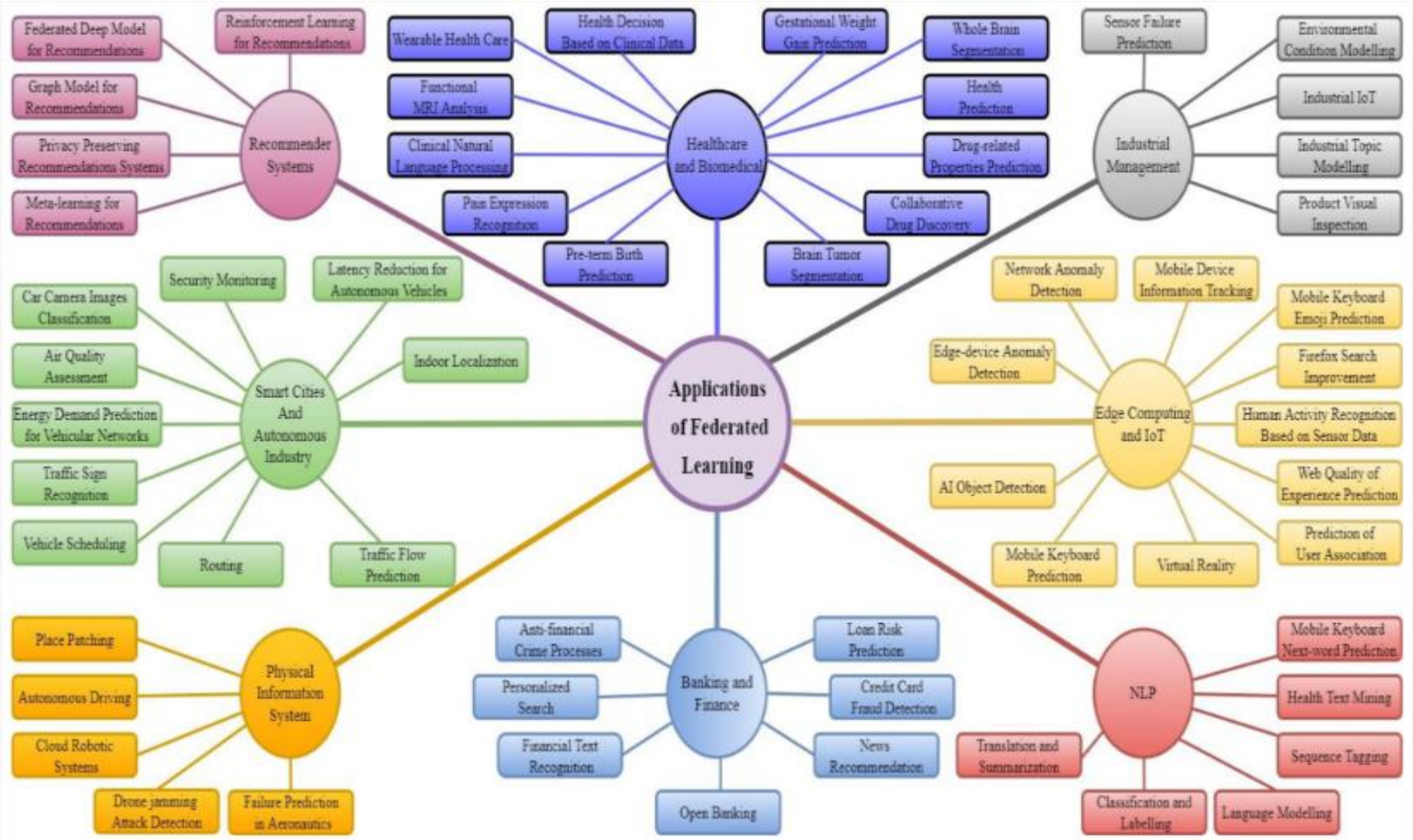
Criação de Servidor:

- **http://10.0.22.37:8000/containers/create/server?servidor_img =alimentado-servidor&porta =5010&eu ia =10&clientes =4&algoritmo =MédiaFed&modelo =cnn&rodadas =10&épocas =5 e prever =verdadeiro**

Implantação rápida de um ambiente de teste e execução de testes:

- **http://10.0.22.37:8000/run?img_server=fed-server&img_client =alimentado-cliente&modelo =cnn&clientes =4&rodadas =10&épocas =5&prever =verdade&prever_cliente =verdadeiro**

Inscrições para Federado



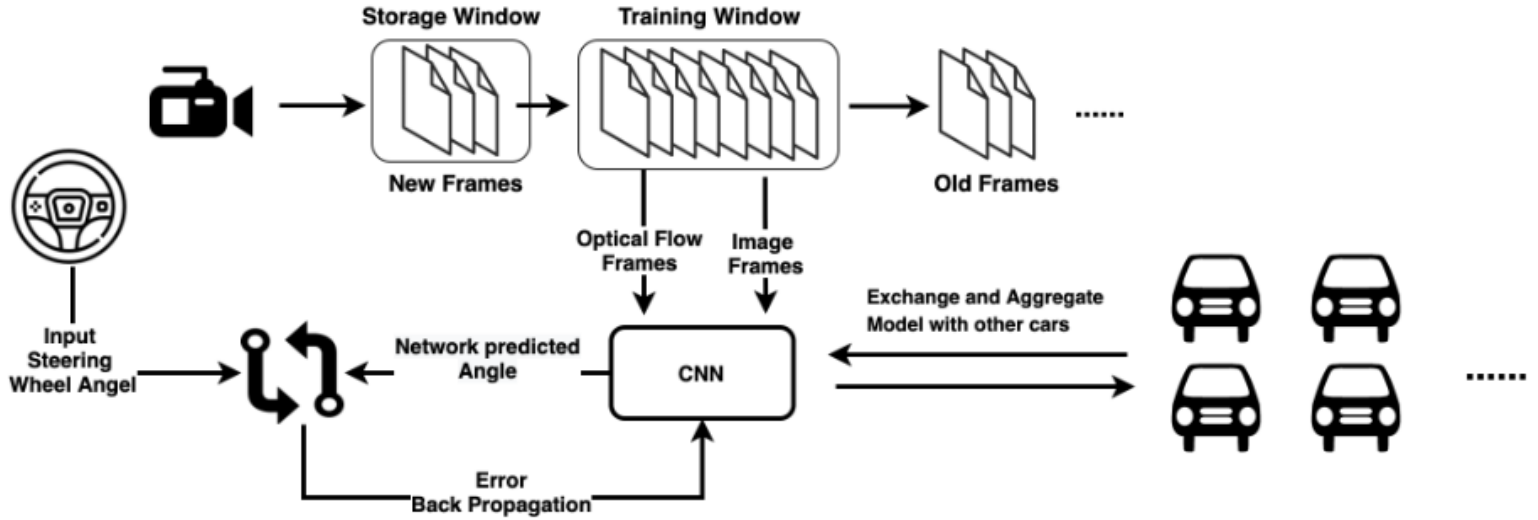
Inscrições para Federado

Domain	Applications
Edge computing	FL is implemented in edge systems using the MEC (mobile edge computing) and DRL (deep reinforcement learning) frameworks for anomaly and intrusion detection.
Recommender systems	To learn the matrix, federated collaborative filter methods are built utilizing a stochastic gradient approach and secured matrix factorization using federated SGD.
NLP	FL is applied in next-word prediction in mobile keyboards by adopting the FedAvg algorithm to learn CIFG [93].
IoT	FL could be one way to handle data privacy concerns while still providing a reliable learning model
Mobile service	The predicting services are based on the training data coming from edge devices of the users, such as mobile devices.
Biomedical	The volume of biomedical data is continually increasing. However, due to privacy and regulatory considerations, the capacity to evaluate these data is limited. By collectively building a global model for the prediction of brain age, the FL paradigm in the neuroimaging domain works effectively.
Healthcare	Owkin [31] and Intel [32] are researching how FL could be leveraged to protect patients‘ data privacy while also using the data for better diagnosis.
Autonomous industry	Another important reason to use FL is that it can potentially minimize latency. Federated learning may enable autonomous vehicles to behave more quickly and correctly, minimizing accidents and increasing safety. Furthermore, it can be used to predict traffic flow.
Banking and finance	The FL is applied in open banking and in finance for anti-financial crime processes, loan risk prediction, and the detection of financial crimes.

Aprendizagem federada em auto-dirigindo

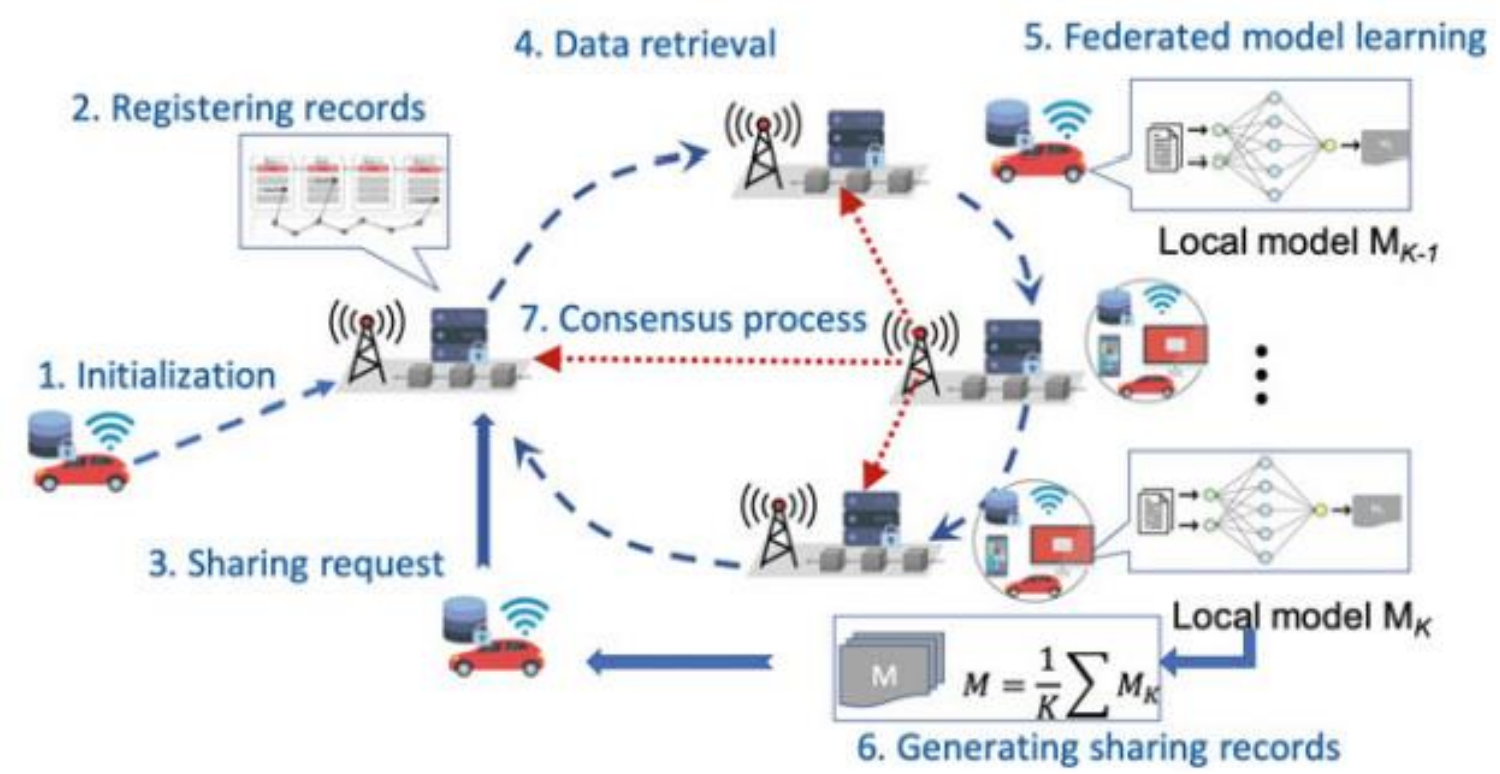
- Veículos de borda computam o modelo localmente; após completar cada época de treinamento local, eles recuperam a versão do modelo global e a comparam com sua versão local.

- Para formar uma consciência global de todos os modelos locais, o servidor central realiza a agregação com base na proporção determinada pelas versões do modelo global e local.
- O servidor de agregação retorna o resultado agregado aos veículos de ponta que solicitam o modelo mais recente.



Federado baseado em borda

- Compartilhamento de modelo habilitado pelo MEC
  - **Inteligência de borda para redes de borda sem fio e aprimora a inteligência conectada entre dispositivos finais em redes 6G.**



- https://www.pdl.cmu.edu/SDI/2019/slides/2019-09-05Federated%20Learning.pdf
- https://wires.onlinelibrary.wiley.com/doi/epdf/10.1002/widm.1443
- https://medium.com/tensorflow/introduzindo-tensorflow-federateda4147aa20041