Localização Indoor por Meio de Aprendizagem de Máquina Apoiada por Beacons Virtuais

Herbert de Oliveira[1,2]
orcid.org/0000-0002-9162-7030

Marcelo Daride Gaspar[1,2]
orcid.org/0000-0002-2249-8108

Victor Azevêdo[1,2]
orcid.org/0000-0002-3184-4527

Paulo Salgado[1]
orcid.org/0000-0002-2396-7973

Carmelo Bastos-Filho[1]
orcid.org/0000-0002-0924-5341

[1]Escola Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil. E-mail: hramos@fitec.org.br mgaspar@fitec.org.br; vmazevedo@fitec.org.br; paulosalgado10@gmail.com; carmelo.filho@upe.br

[2]Fundação para Inovações Tecnológicas - FITec, Recife, Brasil.

DOI: 10.25286/repa.v7i2.2219

Como citar este artigo pela NBR 6023/2018: Herbert de Oliveira; Marcelo Daride Gaspar; Victor Azevêdo; Paulo Salgado;Carmelo Bastos-Filho. Localização Indoor por Meio de Aprendizagem de Máquina Apoiada por Beacons Virtuais. Revista de Engenharia e Pesquisa Aplicada, v.7, n. 2, p. 65-74, 2022

RESUMO

Este artigo apresenta uma solução ao problema de localização *indoor* por meio de aprendizagem de máquina com o apoio de um novo conceito denominado *beacon* virtual. Esse conceito mostrou consideráveis ganhos em desempenho em modelos onde a representatividade dos dados é crucial na precisão das predições do modelo. *Beacons* virtuais também podem ser úteis em ambientes onde a instalação de *beacons* de referência em determinados pontos poderiam gerar transtornos à movimentação de pessoas e objetos em geral. A título de comparação de desempenho, a solução foi implementada considerando quatro algoritmos diferentes de aprendizagem de máquina, sendo dois deles lineares e os outros dois não lineares. Validações com dados reais apontaram o modelo baseado em *Multilayer Perceptron* (MLP) como o modelo de melhor desempenho entre os quatro modelos considerados no que diz respeito ao menor erro entre a posição predita e a real, sendo que a aplicação do conceito de *beacon* virtual fora determinante para tal resultado.

PALAVRAS-CHAVE: Localização *Indoor*; *Beacon*; BLE;

ABSTACT

This article presents a solution to the indoor location problem through machine learning with the support of a new concept called virtual beacon. This concept has shown considerable performance gains in models where the representativeness of the data is crucial in the accuracy of the model's predictions. Virtual beacons can also be useful in environments where the installation of reference beacons at certain points could cause disturbance to the movement of people and objects in general. By way of performance comparison, the solution was implemented considering four different machine learning algorithms, two of them linear and the other two non-linear. Validations with real data pointed the model based on Multilayer Perceptron (MLP) as the model with the best performance among the four models considered with regard to the smallest error between the predicted and the real position, and the application of the virtual beacon concept outside determinant for such a result.

KEY-WORDS: Indoor Localization; Beacon; BLE;

1 INTRODUÇÃO

Localização *indoor* é uma técnica utilizada para determinar a posição de objetos de interesse em ambientes fechados onde o desempenho de um sistema Sistema Global de Navegação por Satélite (*Global Navigation Satellite System* - GNSS) é insatisfatório. Essa técnica conta, atualmente, com diversos métodos e tecnologias de apoio. O método *fingerprint* é um dos métodos mais comumente adotados, principalmente quando se emprega algoritmos de aprendizagem de máquina.

O método *fingerprint* se baseia na força do sinal de rádio (*Radio Signal Strength Indicator* - RSSI) percebido pelos receptores ao redor da área de interesse e proveniente de um dispositivo denominado *beacon* que emite mensagens via ondas de rádio periodicamente. O nome da técnica provém da tese de que cada ponto da área de interesse em que o *beacon* estiver provocará um conjunto único de valores de RSSI percebidos pelos receptores, formando um identificador ou uma espécie de impressão digital daquela posição. No entanto, o RSSI é consideravelmente afetado por fatores externos aleatórios, o que implica valores sob variações imprevisíveis e que afetam drasticamente a precisão da localização.

Além disso, para que se obtenha precisão na localização é necessário que se treine o algoritmo com valores de RSSI obtidos em cada uma das centenas ou milhares de células do *grid* que representa a área de interesse, o que pode ser trabalhoso e pouco prático em determinados ambientes.

Para mitigar esses efeitos, apresenta-se um método baseado em dois pilares: 1) *beacons* fixos de referência para monitorar a variação do RSSI no ambiente e 2) *beacons* virtuais para facilitar a coleta de dados de treinamento em células do grid da área de interesse. Essa abordagem permitiu predições de posições xy com erro médio de 1,08 m.

Esse artigo está organizado da seguinte forma: a seção 1 apresenta uma breve introdução; na seção 2 são descritos alguns trabalhos relacionados apontando o estado da arte; uma descrição do ambiente de teste é feito na seção 3; e a seção 4 descreve a metodologia utilizada, a base de dados, o pré-processamento, os modelos de previsão e análise comparativa, na seção 5 são descritos os resultados, e na seção 6 as conclusões e trabalhos futuros.

1.1 OBJETIVO DO PROJETO

Este artigo tem como objetivo apresentar uma proposta para o problema de localização *indoor* baseada em modelos de aprendizagem de máquina e apoiada por *beacons* virtuais e por *beacons* de referência reais. Objetiva-se também a comparação de desempenho entre quatro modelos de aprendizagem de máquina implementados no que diz respeito ao menor erro entre a posição predita e a posição real.

2 TRABALHOS RELACIONADOS

Há na literatura muitos trabalhos relacionados à solução do problema de localização indoor por meio de algoritmos baseados no método *fingerprint* com apoio de aprendizado de máquina **[1]**.

Em **[2]** é apresentada uma solução para navegação *indoor* por celular baseada no sinal de *gateways* Wi-Fi do ambiente e no modelo *Support Vector Regression* (SVR). Os autores compararam os resultados obtidos com os resultados de modelos baseados em *Artificial Neural Network* (ANN) e modelo probabilísticos. A abordagem SVR, de acordo com os autores, apresentou o melhor desempenho em relação à precisão da estimativa da localização, chegando a obter um erro de 0,48 m para até 50% das medições e 1,40 m para até 90% das medições.

Em **[3]** é apresentado uma proposta onde foram examinados alguns algoritmos de aprendizagem de máquina para localização indoor baseado nos sensores disponíveis nos *smartphones*. Foi possível encontrar algoritmos com acurácia de até 0,76 m no mundo real sem a necessidade de *hardware* específico. Foi desenvolvido um aplicativo *Android* para fazer a coleta de dados. Foi feita a coleta e depois a análise dos dados obtidos. O objetivo era avaliar o desempenho não só estático e *offline*, mas também no mundo real e em movimento.

Na parte *offline*, segundo o artigo, o algoritmo que teve melhor desempenho foi o algoritmo K, apresentando um erro médio de 1,13 e 0,76 metro para x e y, respectivamente e um erro médio absoluto de 1,36 metro. O algoritmo *RBFRegressor* apresentou erro médio de 1,37 e 1,33 metro para x e y, respectivamente e erro absoluto médio de 1,91 metro. Os autores também concluíram que, utilizando uma base de dados menor para treino, 50% da base usada originalmente, melhorava um

pouco a acurácia dos algoritmos. Isso levou à hipótese de que o uso de dados completos, nesse caso, estava provocando *overfitting*.

No modo *online*, o algoritmo K, levou entre 30 e 45 segundos para calcular a posição enquanto o algoritmo RBF fazia a previsão quase que instantaneamente. Foi proposto um modelo híbrido que segmentou o conjunto de dados em segmentos menores para treinar o algoritmo K. Dessa forma, foi possível melhorar o desempenho *online* desse algoritmo. O modelo híbrido proposto conseguiu uma acurácia e velocidade similar aos modelos *offline*.

Em **[4]** é apresentado um método baseado em *Artificial Neural Network* (ANN) para posicionamento interno utilizando-se RSSI. O método foi testado com três tipos diferentes de conjuntos de dados de treinamento em um ambiente heterogêneo. Cerca de 40 amostras de dados de treinamento foram usadas em um espaço de amostra de 8 × 9 m. O estudo concluiu que a localização do usuário móvel pode ser determinada no ambiente interno com diferentes níveis de exatidão e precisão, dependendo do tipo de conjunto de dados de treinamento usado. As coordenadas x-y calculadas usando a média direcional deram a melhor exatidão e precisão. Os resultados mostraram que o modelo ANN conseguiu alcançar uma acurácia média de 0,7 metro.

Em **[5]** foi proposto um algoritmo de localização interna por *fingerprinting* baseado em arquiteturas profundas (*deep architectures*) devido à alta complexidade dos ambientes internos. Foram utilizados os modelos mais comuns para regressão como o *Deep Neuronal Network* (DNN), *Deep Belief Network*, *Restricted Boltzmann Machines* (RBM) e *Deep Belief Network* (DBN) com a primeira camada com *Gaussian-Bernoulli* DBN (GB-DBN).

Os resultados experimentais demonstraram que os *deep models* forneceram um desempenho de generalização eficiente em ambientes internos. Eles têm a desvantagem de exigir altos recursos de processamento quando são treinados na fase *off-line*. No entanto, *deep models* são rápidos para executar a predição durante a fase *on-line*. Os 3 modelos tiveram uma acurácia média de menos de 2 metros, nos dados simulados. O modelo DNN obteve o melhor resultado com apenas 1,00598 metro de erro médio.

3 AMBIENTE DE TESTES

O ambiente de testes considerado neste trabalho consiste em uma área interna retangular livre com dimensões de 7,46 m x 3,67 m, conforme representa Figura 1. Nessa área, foram instalados 3 *gateways* BLE e 14 *beacons* BLE de referência cujas coordenadas x-y são previamente conhecidas. A decisão pelo uso de *beacons* de referência nesse projeto foi baseada em três objetivos explicados a seguir.

O primeiro objetivo é produzir os valores de RSSI usados para se treinar os modelos. Durante o treinamento dos modelos, considera-se que há um *beacon* exatamente na mesma posição de um dos *beacons* de referência. Dessa forma, utiliza-se os valores de RSSI e posição desse *beacon* de referência como se fosse um *beacon* de teste.

O segundo objetivo é prover um contexto para cada nova estimativa de posição. Sabe-se que os valores de RSSI variam consideravelmente. Dessa forma, os *beacons* de referência podem informar ao modelo, durante a predição, a situação atual do ambiente em termos de valores de RSSI. Assim, para predizer a posição, o modelo recebe, como entrada: 1) os valores de RSSI obtidos por cada um dos três *gateways* referentes ao *beacon* de que se deseja obter a posição e 2) mais valores de RSSI obtidos por cada um dos três *gateways* referentes aos 14 *beacons* de referência. Ou seja, para predizer a posição de um *beacon*, o modelo recebe como entrada 45 valores de RSSI. A Equação (1) permite obter a dimensionalidade do problema.

$$dim = n_g(n_{br} + 1) \quad \textbf{(1)}$$

Onde:

dim: $dimensionalidade\ do\ problema$

n_g: $número\ de\ gateways$

n_{br}: $número\ de\ beacons\ de\ referência$

Finalmente, o terceiro objetivo seria o de criar *beacons* virtuais visando melhor desempenho nas predições realizadas pelos modelos. Os *beacons* de referência estão fisicamente localizados em determinados pontos balizados às paredes da área considerada. Para quaisquer outros pontos diferentes desses os modelos não teriam uma referência para treinamento. Para esses casos, a predição contaria apenas com a capacidade de inferência do próprio modelo a partir apenas dos 14 *beacons* de referência.

Opcionalmente, poder-se-ia instalar mais *beacons* em outros pontos da área de

DOI: 10.25286/repa.v7i2.2219

rastreamento. Porém, essa opção pode se apresentar pouco prática. Outra opção seria a instalação de tais *beacons* apenas para treinamento. No entanto, o modelo precisa ser retreinado periodicamente, toda vez que o erro da estimativa de posição de um *beacon* de referência ultrapassar determinado patamar. Normalmente, os ambientes reais apresentam certa dinâmica no que diz respeito à inclusão e/ou remoção de novos obstáculos físicos e à presença de interferências eletromagnéticas de diferentes intensidades e frequências.

Para mitigar essas dificuldades, foi desenvolvido o conceito de ***beacons* virtuais**. Trata-se de um *beacon* não real que existe apenas logicamente. Nesse conceito, dado um determinado ponto na área de interesse, estima-se o valor de RSSI que cada um dos três *gateways* receberia se nesse ponto houvesse um *beacon* real. A estimativa é baseada na Equação (2).

$$d = 10^{\frac{C_r - R}{10\gamma}} \quad \textbf{(2)}$$

Onde:

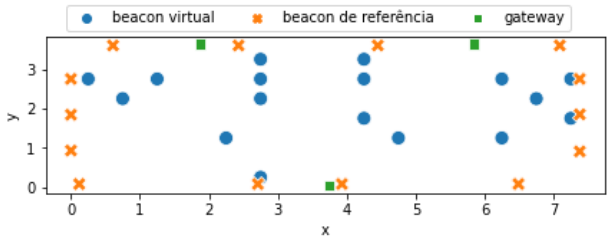
$\boldsymbol{d}$: *distância do beacon ao gateway*
$\boldsymbol{C_r}$: *RSSI observado pelo gateway com beacon a um metro de distância do gateway.*
$\boldsymbol{R}$: *RSSI observado pelo gateway com beacon a uma distância a ser determinada pela equação.*
$\boldsymbol{\gamma}$: *perda da força do sinal ao longo de sua propagação.*

Nesse trabalho, os parâmetros *Cr* e *γ* são estimados por um estimador força bruta, já que as faixas de valores desses parâmetros são conhecidas e relativamente estreitas. O estimador utiliza os valores *d* e *R* dos *beacons* de referência, pois, esses são valores conhecidos *a priori* pelo algoritmo.

Os *beacons* virtuais são criados apenas onde há a interseção de pelo menos três retas ligando um *beacon* de referência qualquer a um determinado *gateway*. Isso porque os parâmetros estimados não variam de forma considerável ao longo da reta "*beacon* de referência-*gateway*". Isso permite que os *beacon* virtuais ofereçam valores de qualidade, isto é, valores confiáveis ao treinamento dos modelos.

Com essa restrição, foi possível criar 17 *beacons* virtuais espalhados pela área de interesse, conforme apresenta a Figura 1. Os *Beacons* virtuais permitiram que o número de pontos com RSSI conhecidos usados no treinamento do modelo mais que dobrasse, passando de 14 pontos para 31 pontos.

Figura 1 – Disposição de *gateways*, *beacons* de referência e *beacons* virtuais



Fonte: próprio autor

4 METODOLOGIA

A metodologia empregada considerou as seguintes etapas: 1) aquisição de dados de treinamento/teste; 2) aquisição de dados de validação; 3) criação dos *beacons* virtuais; 4) pré-processamento de dados; 5) criação de modelos lineares e não lineares; 6) seleção dos modelos e 7) validação.

4.1 AQUISIÇÃO DE DADOS DE TESTE/TREINAMENTO

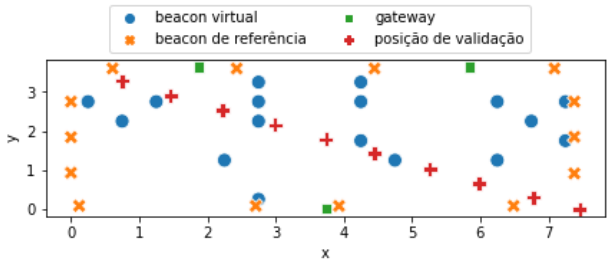
Os *beacons* utilizados nesse trabalho transmitem uma mensagem BLE *Eddystone* a cada 300 ms. Cada mensagem transmitida é recebida simultaneamente por 3 *gateways* BLE/Wi-Fi. Os *gateways* determinam o RSSI de tal mensagem e incluem esse valor em uma mensagem MQTT que inclui, além do RSSI, o identificador do *gateway*, o MAC do *beacon* que transmitiu a mensagem e o *timestamp* do momento da recepção da mensagem BLE pelo *gateway*. Os três *gateways* têm os seus relógios internos sincronizados por um servidor de tempo na nuvem. A mensagem MQTT é enviada via WiFi para um servidor local que organiza os dados em estruturas de dados adequadas em memória.

4.2 AQUISIÇÃO DE DADOS DE VALIDAÇÃO

A aquisição de dados de validação segue o mesmo modelo apresentado acima. Porém, definiu-se dez pontos ao longo de uma das diagonais da área considerada nos quais, em cada momento, um *beacon* de validação foi disposto. Distribuindo-se pontos de validação ao longo de uma diagonal, garante-se que cada ponto de validação tenha uma coordenada com valores de x e de y diferentes, o que é interessante para uma validação mais isenta de tendências. A partir dessa definição, gerou-se dez arquivos de RSSI relacionados aos respectivos pontos. Dessa forma, a comparação entre os valores preditos e o valor real de posição do *beacon* de validação pode ser realizada de forma

inequívoca. A Figura 2 apresenta o diagrama dos pontos de validação.

Figura 2 – Disposição dos pontos de validação, *gateways*, *beacons* de referência e *beacons* virtuais.



Fonte: próprio autor

4.3 CRIAÇÃO DE BEACONS VIRTUAIS

A seção 3 explicou o conceito de *beacons* virtuais. A criação de *beacons* virtuais é uma das etapas da metodologia. Conforme explicado, os *beacons* virtuais são criados na interseção entre pelo menos três retas que ligam *beacons* de referência a *gateways* diferentes.

4.4 PRÉ-PROCESSAMENTO

Foram utilizados dois níveis de pré-processamento. O nível mais baixo, é responsável por filtrar os dados de RSSI adquiridos. Essa filtragem é realizada por meio de Filtro de *Kalman*.

No nível mais alto de pré-processamento, cada valor de RSSI filtrado é sincronizado e colocado em um *dataframe*.

Em seguida, considera-se os *beacons* de teste. Para cada uma das 600 amostras, um dos 14 *beacons* de referência é considerado como *beacon* de teste.

Em seguida, cria-se uma cópia dos dados adquiridos e considera-se a mesma técnica acima para criar *beacons* de teste, porém, agora, a partir dos *beacons* virtuais.

4.5 CRIAÇÃO DE MODELOS LINEARES E NÃO LINEARES.

Neste estudo, embora esteja-se considerando um problema não-linear, foram implementados algoritmos lineares e não-lineares, de forma a se obter um comparativo de performance entre eles.

Os algoritmos utilizados foram:

- Lineares:
  a) Regressão logística;
  b) *k-Nearest Neighbors* (kNN) para regressão.
- Não-lineares:
  a) *Multi-layer Perceptron* (MLP) R*egressor*;
  b) *Support Vector Regression* (SVR).

4.6 TREINAMENTO DOS MODELOS

Encontrar um modelo adequado e definir seus parâmetros é um grande desafio na aprendizagem supervisionada. De acordo com **[6]**, um problema recorrente no treinamento de um método é o ajuste excessivo ou o*verfitting*, onde o modelo se ajusta bem aos dados de treinamento, mas não consegue atingir a mesma precisão em um conjunto de dados de teste independente.

Uma estratégia para reduzir o *overfitting* é a validação cruzada *k-fold*, que repete o processo de aprendizagem *k* vezes, com diferentes conjuntos de treinamento e validação.

Além disso, um estimador pode obter modelos mais acurados, de acordo com a sintonia dos hiperparâmetros (parâmetros que não são aprendidos através dos estimadores). Desse modo, é possível e recomendado realizar uma rede de pesquisa (*Grid Search*) no espaço dos hiperparâmetros para encontrar o melhor *score* da validação cruzada **[7]**.

Adotou-se a validação cruzada conciliada ao *Grid Search* através do objeto *GridSearchCV* da biblioteca *scikit-learn.* Ele realiza a validação cruzada com o método empregado e retorna um classificador que atinge a melhor pontuação, bem como pontuações para todas as combinações de parâmetros **[7]**.

Ressalta-se que os pontos x-y usados no teste são os mesmos usados durante o treinamento, embora os valores de RSSI sejam diferentes, já que na fase de pré-processamento linhas duplicadas são removidas do *dataframe* de treinamento. Essa abordagem permite que o modelo seja retreinado periodicamente contando apenas com a infraestrutura fixa do ambiente constituída por *gateways* e *beacons* de referência, além dos *beacons* virtuais.

4.7 VALIDAÇÃO

A etapa de validação usou pontos x-y jamais vistos pelos modelos durante a fase de treinamento/teste. Conforme descrito na seção 4.2, foram definidos dez pontos de validação ao longo de uma das diagonais da área considerada. Os pontos usados na validação são aqueles apresentados na Figura 2.

DOI: 10.25286/repa.v7i2.2219

5 RESULTADOS

Conforme mencionado, foram treinados quatro modelos diferentes: dois deles baseados em algoritmos lineares e outros dois baseados em algoritmos não lineares. Essa abordagem visa comparar o desempenho desses algoritmos face ao problema de localização *indoor* que é um problema não linear.

Cada modelo é composto por dois submodelos: uma para predizer o valor da coordenada x (eixo das abscissas) e outro para a coordenada y (eixo das ordenadas).

Como mencionado na subseção **Erro! Fonte de referência não encontrada.**, o teste foi realizado com valores de posições utilizados durante o treinamento. Embora os dados de treinamento e testes apresentassem valores de coordenadas iguais entre várias linhas do conjunto de dados, cada linha é garantidamente única, pois os 45 valores de RSSI não são os mesmos entre duas linhas quaisquer. Isto é, os dados de treinamento e teste são, a rigor, diferentes.

Foram geradas versões dos modelos utilizando dados de treinamento com e sem *beacons* virtuais, de forma a avaliar a melhoria alcançada pela técnica de *beacons* virtuais.

A Tabela 1 apresenta a performance do treinamento para os modelos de cada algoritmo. Os hiperparâmetros foram selecionados por *Grid Search CV*, conforme subseção 5.1. São apresentados o *score* médio e o desvio padrão dessa média para cada modelo. Observa-se que o melhor *score* foi para o modelo SVR sem *beacons* virtuais. A análise da seção 0, onde são avaliados os desempenhos dos modelos, mostrará que esses valores não são suficientes para uma conclusão final sobre o modelo de melhor desempenho.

Tabela 1 – Desempenho de treinamento dos modelos com e sem *beacons* virtuais.

		mean score		std score	
	Eixo	abscissas	ordenadas	abscissas	ordenadas
Algoritmo	VBeacon				
KNN	não	0.936606	0.978593	0.091197	0.017488
	sim	0.951963	0.903743	0.009310	0.050819
LR	não	0.876783	0.649668	0.027345	0.034938
	sim	0.683725	0.490486	0.030625	0.032021
MLP	não	0.919182	0.810989	0.013548	0.052078
	sim	0.830773	0.651473	0.052429	0.126263
SVR	não	0.960275	0.975739	0.043419	0.005495
	sim	0.941758	0.882392	0.007827	0.022467

5.1 GRID SEARCH CV

Visando obter modelos otimizados, empregou-se um *Grid Search* de cada algoritmo considerado (kNN, LR, MLP e SVR), para cada eixo do plano cartesiano estudado. Além disso, foram avaliados modelos considerando e desconsiderando os *beacons* virtuais. Desse modo, para cada algoritmo, realizou-se quatro *grid searchs*. Devido ao fato de cada algoritmo possuir seus próprios hiperparâmetros, não é possível avaliar o mesmo campo para todos os modelos. Todas as validações cruzadas foram feitas considerando cinco *folds*.

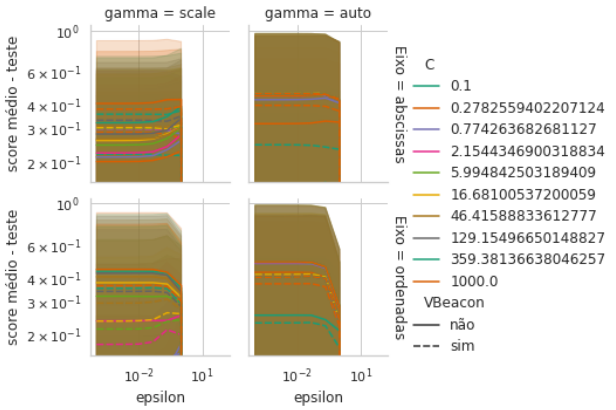
5.1.1 Regressão Logística

Para o modelo baseado em Regressão logística, o campo de variação dos hiperparâmetros foi definido como:

```
{'C': array([1.000e-02, 7.197e-02, 5.179e-
01, 3.728e+00, 2.683e+01, 1.931e+02,
1.389e+03, 1.000e+04]), 'solver':
['newton-cg', 'lbfgs', 'liblinear', 'sag',
'saga'], 'penalty': ['l2']}
```

A Figura 3 apresenta o resultado da execução do algoritmo *Grid Search CV* para o modelo baseado em Regressão Logística considerando tanto a abordagem com *beacons* virtuais quanto sem *beacons* virtuais. Na mesma figura, apresenta-se diferentes curvas de *score* médio de teste considerando diferentes valores do parâmetro C. Observa-se que o desempenho de treinamento com *beacons* virtuais apresenta desempenho melhor em relação à abordagem sem *beacons* virtuais.

Figura 3 – Treinamento de modelos baseados em Regressão Logística para x (acima) e y (abaixo)



Fonte: próprio autor

Os hiperparâmetros dos modelos selecionados de Regressão Logística, através de *Grid Search CV* e elencados por *score*, podem ser vistos na Tabela 2.

Tabela 2– Hiperparâmetros dos modelos selecionados de regressão logística.

	VBeacon	Eixo	C	penalty	solver
0	não	abscissas	1389.495494	l2	lbfgs
1	não	ordenadas	0.071969	l2	liblinear
2	sim	abscissas	3.727594	l2	newton-cg
3	sim	ordenadas	3.727594	l2	lbfgs

5.1.2 kNN para Regressão

Para o modelo *k-Nearest Neighbors* o campo de variação dos hiperparâmetros foi definido como:

```
{'n_neighbors': array([ 1,  3,  5,  7,  9,
11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31,
33, 35, 37, 39, 41, 43, 45]), 'weights':
['uniform', 'distance'], 'metric':
['minkowski', 'euclidean', 'manhattan',
'chebyshev'], 'algorithm': ['ball_tree',
'kd_tree', 'brute']}
```

Calculou-se o score médio de teste (curvas não apresentadas) considerando pesos diferentes e métricas diferentes para se calcular a distância aos vizinhos: distância Euclidiana e distância Manhattan. Observou-se que o desempenho de treinamento com beacons virtuais apresentou desempenho melhor em relação à abordagem sem beacons virtuais para a estimativa do valor da coordenada x. Para a coordenada y, não houve diferença expressiva.

Os hiperparâmetros dos modelos selecionados de kNN, através de *Grid Search CV* e elencados por *score*, podem ser vistos na Tabela 3.

Tabela 3– Hiperparâmetros dos modelos de selecionados de KNN Regressor.

	VBeacon	Eixo	algorithm	metric	n_neighbors	weights
0	não	abscissas	ball_tree	minkowski	5	distance
1	não	ordenadas	brute	chebyshev	1	uniform
2	sim	abscissas	kd_tree	chebyshev	3	distance
3	sim	ordenadas	kd_tree	chebyshev	3	distance

5.1.3 MLP Regressor

Para o modelo baseado em MLP *Regressor* o campo de variação dos hiperparâmetros foi definido como:

```
{'activation': ['relu', 'tanh', 'logistic',
'identity'], 'hidden_layer_sizes': [(50,
100), (50, 150), (100, 50), (100, 150),
(150, 50), (150, 100), (50, 100, 150), (50,
150, 100), (100, 50, 150), (100, 150, 50),
(150, 50, 100), (150, 100, 50), 50, 100,
150],'solver': ['adam', 'lbfgs'],
'learning_rate': ['constant', 'adaptive',
'invscaling']}
```

No resultado do *Grid Search*, observou-se que, para os três modelos de melhor score de treinamento (curvas não apresentadas), a abordagem com beacons virtuais apresentou melhor resultado geral. Em todos esses casos, o otimizador selecionado foi o lbfgs.

Os hiperparâmetros dos modelos selecionados para o modelo MLP, através de *Grid Search CV* e elencados por *score*, podem ser vistos na Tabela 4.

Tabela 4– Hiperparâmetros dos modelos selecionados de MLP Regressor.

	VBeacon	Eixo	activation	hidden_layer_sizes	learning_rate	solver
0	não	abscissas	logistic	(100, 150)	constant	lbfgs
1	não	ordenadas	logistic	(150, 50)	constant	lbfgs
2	sim	abscissas	tanh	(100, 150, 50)	constant	lbfgs
3	sim	ordenadas	tanh	(150, 100, 50)	constant	lbfgs

5.1.4 SVR

Para o modelo baseado em SVR o campo de variação dos hiperparâmetros foi definido como:

```
  {'C': [1.000e-01, 2.783e-01, 7.743e-01,
2.154e+00, 5.995e+00, 1.668e+01,
4.642e+01, 1.292e+02, 3.594e+02,
1.000e+03],
 'epsilon': array([1.000e-04, 4.642e-04,
2.154e-03, 1.000e-02, 4.642e-02, 2.154e-
01,1.000e+00, 4.642e+00, 2.154e+01,
1.000e+02]),
   'gamma': ['scale', 'auto'],
  'kernel': ['rbf', 'sigmoid']}
```

O resultado da execução do algoritmo *Grid Search* apontou a abordagem com *beacons virtuais* como a abordagem de melhor *score*.

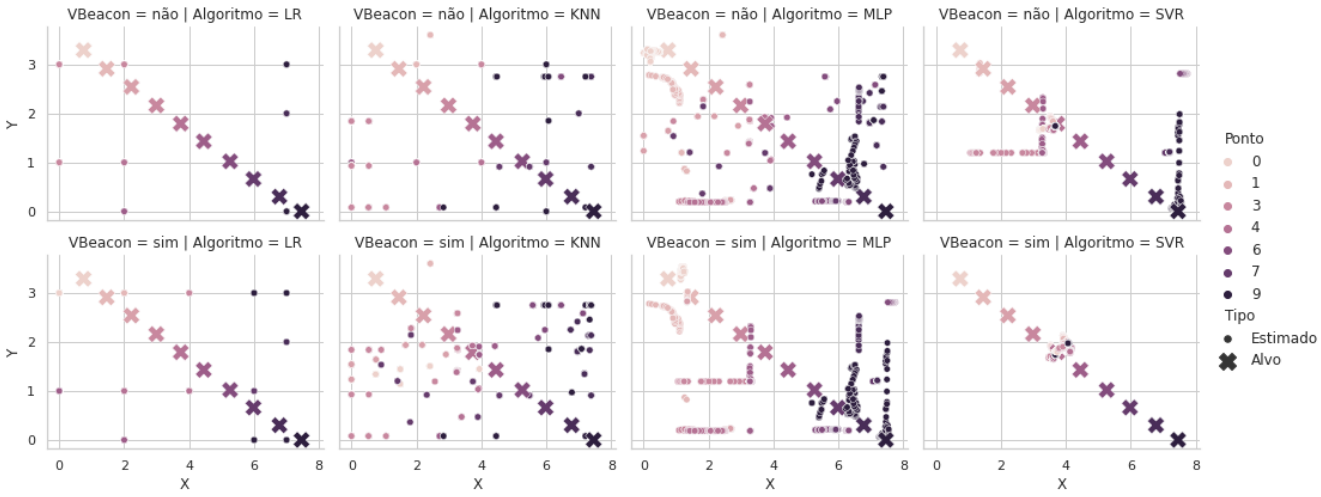
Os hiperparâmetros selecionados para os modelos SVR através de *Grid Search CV* e elencados por score, podem ser vistos na Tabela 5.

DOI: 10.25286/repa.v7i2.2219

Tabela 5– Hiperparâmetros dos modelos de selecionados de SVR.

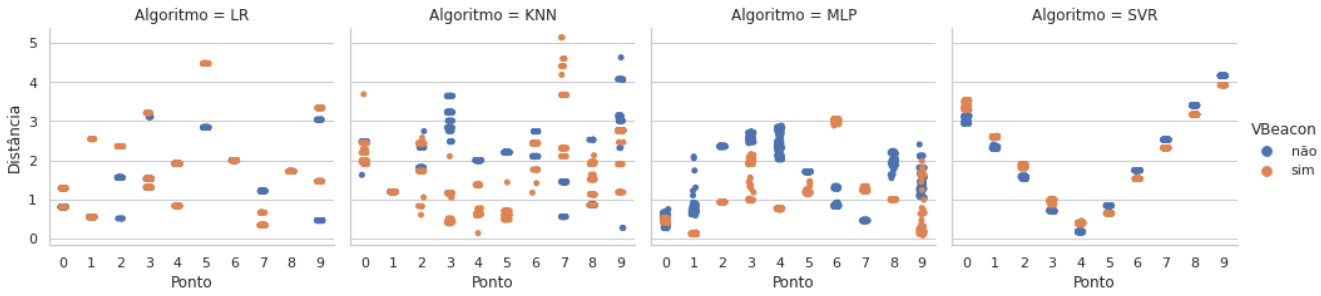
	VBeacon	Eixo	C	epsilon	gamma	kernel
0	não	abscissas	5.994843	0.000100	auto	rbf
1	não	ordenadas	46.415888	0.000100	auto	rbf
2	sim	abscissas	5.994843	0.046416	auto	rbf
3	sim	ordenadas	5.994843	0.046416	auto	rbf

Figura 4 – Representação gráfica dos pontos estimados comparado aos reais. para cada algoritmo.



Fonte: próprio autor

Figura 5 – Distribuição da distância euclidiana entre o valor real e estimado, para cada ponto.



Fonte: próprio autor

5.2 AVALIAÇÃO DOS MODELOS

Conforme mencionado na seção 4.7, os modelos foram submetidos a um conjunto de dados de validação. Os dez pontos de validação foram distribuídos ao longo de uma diagonal da área considerada. Para cada ponto de validação, foram capturadas cerca de 85 amostras de valores de RSSI. Cada amostra corresponde aos valores de RSSI de 14 *beacons* de referência lidos por 3 *gateways*, mais os valores de RSSI do *beacon* de validação lidos por 3 *gateways* totalizado 45 valores de RSSI por amostra. Após pré-processamento, obteve-se cerca de 50 amostras por ponto de validação.

5.2.1 POSIÇÃO ESTIMADA X POSIÇÃO REAL

A Figura 4 apresenta graficamente os pontos estimados por cada modelo em sua versão com e sem *beacon* virtual. Observa-se que, no modelo SVR há uma certa tendência em se estimar a posição na região central da área considerada.
A Figura 5 apresenta as distâncias Euclidianas entre os pontos estimados e os pontos reais para cada modelo. Essa figura auxilia a visualização da Figura

4. Tais distâncias podem ser interpretadas como o erro da estimativa. Observa-se, graficamente, que o modelo MLP na versão com *beacons* virtuais apresentou os menores erros e as menores variações desses erros.

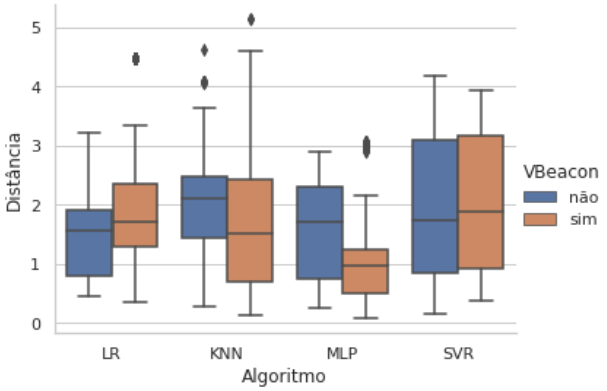
A Tabela 1 apresentada na Seção 5 apontou o modelo kNN como o modelo com melhor *score* de treinamento. Mas quando esse modelo foi submetido a entradas referentes a posições não experimentadas durante o treinamento, seu desempenho não se mostrou como o melhor entre os quatro modelos considerados.

5.2.2 Análise *Boxplot*

Para complementar a análise de desempenho dos modelos, foram gerados gráficos do tipo *boxplot*, apresentados na Figura 6.

Esses gráficos consideram os erros de estimativas de posições do *beacon* de validação pelos quatro modelos em suas versões com e sem *beacons* virtuais. Observa-se que o modelo MLP em sua versão com *beacons* virtuais apresentou o melhor desempenho entre todos os modelos considerando: menor mediana (2º quartil), menor distância ente o 1º e 3º quartis, menor erro mínimo, menor erro máximo e menores *outliers*. Observa-se o quão benéfico são os *beacons* virtuais para esse modelo: a versão sem *beacons* virtuais se apresenta como um dos piores modelos enquanto a versão com *beacons* virtuais se apresentou como o melhor modelo entre os quatro modelos considerados, conforme Figura 6. Possivelmente, isso pode ser explicado pela necessidade desse modelo em ser treinado utilizando-se um conjunto de dados que apresente considerável grau de representatividade.

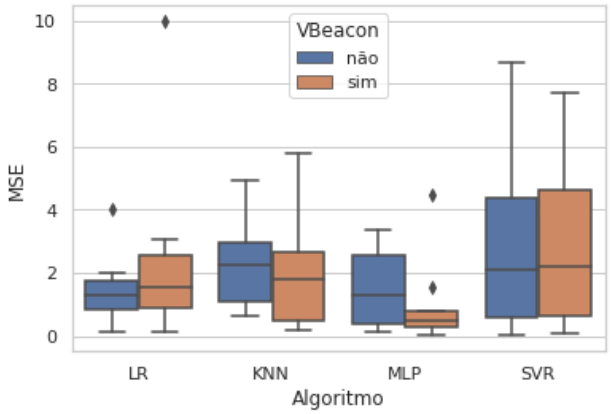
Figura 6 – Gráfico boxplot considerando os erros de estimativas dos modelos (Distância Euclidiana).



Fonte: próprio autor

De forma semelhante, foram gerados, também, os gráficos *boxplot* considerando o Erro Médio Quadrático (do inglês *Mean Square Error* - MSE), conforme apresenta a Figura 7.

Figura 7 – Gráfico *boxplot* considerando MSE dos modelos.



Fonte: próprio autor

Novamente, o modelo MLP apresentou o melhor desempenho no que diz respeito a: menor mediana (2º quartil), menor distância ente o 1º e 3º quartis, menor erro mínimo, menor erro máximo e menores outliers.

5.2.3 Desempenho médio

Para finalizar a análise de desempenho dos modelos, foram calculadas as médias dos erros e as médias dos MSEs entre os dez pontos de validação, apresentadas na Tabela 6.

Tabela 6 – Desempenho médio dos modelos considerando distâncias Euclidianas entre os dez pontos de validação

		Distância				
		mean	std	var	min	max
Algoritmo	VBeacon					
KNN	não	2.040795	0.767906	0.589680	0.268366	4.625993
	sim	1.640463	0.929969	0.864842	0.135185	5.138666
LR	não	1.524760	0.749606	0.561909	0.460000	3.207133
	sim	1.872253	1.152809	1.328968	0.340588	4.470727
MLP	não	1.539557	0.773761	0.598706	0.258421	2.883295
	sim	1.078848	0.809977	0.656063	0.079855	3.060254
SVR	não	1.985613	1.227710	1.507273	0.157649	4.164514
	sim	2.021222	1.166989	1.361864	0.366998	3.923879

Embora o MSE não seja de interpretação direta, pois, nesse caso, sua grandeza seria o m^2, a Tabela 7 apresenta, entre outros itens calculados de apoio, a média dos MSEs calculada entre os dez pontos de validação. Sugere-se interpretar, de uma forma simplificada, como quanto menor o MSE, melhor será o desempenho do modelo.

DOI: 10.25286/repa.v7i2.2219

Tabela 7 – Desempenho médio dos modelos considerando o MSE médio entre os dez pontos de validação

		MSE				
		mean	std	var	min	max
Algoritmo	VBeacon					
KNN	não	2.401979	1.532891	2.349754	0.626717	4.946148
	sim	1.885312	1.715608	2.943310	0.190087	5.782606
LR	não	1.419175	1.093763	1.196318	0.149850	4.023700
	sim	2.380317	2.827481	7.994651	0.105059	9.993700
MLP	não	1.467407	1.205017	1.452066	0.105008	3.344318
	sim	0.916059	1.331259	1.772252	0.007058	4.498282
SVR	não	2.841675	2.818007	7.941164	0.014598	8.652753
	sim	2.830747	2.637890	6.958462	0.077530	7.684418

Sendo assim, observa-se que o modelo MLP em sua versão com *beacons* virtuais apresenta o melhor desempenho com respeito ao MSE médio e ao MSE mínimo, considerando todos os dez pontos de validação.

6 CONCLUSÃO

Esse trabalho teve por objetivo desenvolver quatro modelos de aprendizado de máquina para resolver o problema de localização *indoor* e comparar o desempenho desses modelos entre si.

Concluiu-se que, para a abordagem considerada, o modelo de melhor desempenho considerando o quesito menor erro por distância Euclidiana foi o modelo MLP treinado com *beacons* de referência reais e *bacons* virtuais.

Foi introduzido o conceito de *beacon* virtual, visando a melhoria da representatividade dos dados de treinamento. Os *beacons* virtuais foram responsáveis por uma melhoria considerável no desempenho dos modelos, sobretudo, para o desempenho do modelo MLP.

Trabalhos futuros apontam algumas possibilidades de melhorias de desempenho. A primeira delas seria aumentar o número de *beacons* virtuais os quais se mostraram benéficos ao desempenho dos modelos. Para se aumentar o número de *beacons* virtuais pode-se, por exemplo, aumentar o número de *gateways*.

A segunda possibilidade, seria treinar o algoritmo considerando o ambiente de rastreamento sob diferentes situações em que obstáculos diferentes estariam presentes. Isso poderia aumentar a robustez do algoritmo e conduzi-lo a aplicações práticas.

A terceira possibilidade de melhoria seria utilizar os *beacons* virtuais na predição. Atualmente, eles são usados apenas durante o treinamento. Com isso, a dimensionalidade do problema aumentaria.

Observou-se que, uma abordagem mista que envolve modelos de aprendizagem de máquina, modelos estocásticos (tais como o Filtro de *Kalman*) e modelos analíticos (equação da distância em função do RSSI) podem agregar suas forças com o objetivo de melhorar o desempenho em relação a uma abordagem sem tal sinergia.

REFERÊNCIAS

[1] AHASANUN, Nessa *et al*. **A Survey of Machine Learning for Indoor Positioning**. IEEE Acces. Dezembro, 2020. DOI 10.1109/ACCESS.2020.3039271.

[2] SHI, Ke *et al*. **Support Vector Regression Based Indoor Location** em IEEE 802.11 Environments, *Mobile Information Systems,* p. 14, 2015.

[3] MEHMOOD, Hamid; TRIPATHI, Nintin Kumar e TIPDECHO, Taravudh. **Indoor Positioning System Using Artificial Neural Network**, *Journal of Computer Science,* pp. 1219-1225, 10 de agosto de 2010.

[4] MASCHARKA, David e MANLEY, Eric. **LIPS: Learning Based Indoor Positioning System using mobile phone-based sensors** em *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, EUA, 2016.

[5] FÉLIX, Gibrán; SILLER, Mario e ÁLVAREZ, Ernesto Navarro. **A fingerprinting indoor localization algorithm based deep learning** em *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, Austria, 2016.

[6] KRAMER, Oliver. **Machine Learning** em *Machine Learning for Evolution Strategies*, Springer International Publishing, 2016, pp. 35--43.

[7] KRAMER, Oliver. **Scikit-Learn** em *Machine Learning for Evolution Strategies*, Springer International Publishing, 2016, pp. 45--53.