THIAGO A. LIRA
ADRIANO . DENNANNI
RICARDO Y. NAGANO

NETMAP

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia de Computação, junto ao Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

São Paulo
2016

THIAGO A. LIRA
ADRIANO . DENNANNI
RICARDO Y. NAGANO

NETMAP

Texto apresentado à Escola Politécnica da Universidade de São Paulo como requisito para a conclusão do curso de graduação em Engenharia de Computação, junto ao Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

Área de Concentração:

Engenharia de Computação

Orientador:

Reginaldo Arakaki

São Paulo
2016

FICHA CATALOGRÁFICA

Lira et al.
netmap/ T. A. Lira, A. . Dennanni, R. Y. Nagano. São Paulo, 2016.
20 p.

Monografia (Graduação em Engenharia de Computação) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

1. Assunto #1. 2. Assunto #2. 3. Assunto #3. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais (PCS). II. t.

AGRADECIMENTOS

RESUMO

Mapas físicos tornam-se cada vez menos utilizados com o desenvolvimento progressivo de sistemas de posicionamiento cada vez melhores. O sistema americano GPS é possivelmente o mais utilizado, sendo que ele possibilita qualquer um ter informações sobre sua localização, dando apoio, por exemplo, à praticantes de trilhas e acampamentos, principalmente em casos de emergência. Porém, em ambientes fechados, as ondas eletromagnéticas utilizadas pelos satélites sofrem atenuações e interferências devidos aos materiais de construção, e assim o sistema perde precisão e não funciona com toda a precisão esperada. Como uma alternativa para esta dificuldade, procurou-se desenvolver um sistema, que consegue obter a posição do usuário em um ambiente fechado com precisão, sendo usado para isso técnicas de machine learning, aliadas com dados obtidos de redes em fio já instaladas no local. O sistema consistirá de um servidor central, onde serão enviados os dados e os mesmos serão processados. Os dados serão coletados por meio de um aplicativo de Android, este possuirá duas versões. A versão usuário usará os dados do servidor para localizar o usuário, a versão administrador irá coletar dados novos para serem usados em futuras medições.
PALAVRAS-CHAVE: Indoor, Localização, Wi-fi, Machine Learning

ABSTRACT

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 Apresentação

Com a modernização das tecnologias de telefonia móvel torna-se cada vez maior o número de pessoas com celulares associados a tecnologias de rede, como WiFi, 3G e 4G, que tem o potencial de fornecer informações sobre seu usuário a todo momento, tais como o conteúdo acessado por seus navegadores ou aplicativos, e também informações sobre posição e deslocamento. Dados de localização por si possuem pouco valor, mas quando aliados a outros conteúdos, é possível fornecer conteúdo personalizado em tempo real, reativo ao ambiente, passando a oferecer um grande retorno por um pouco mais de ocupação na banda[1]. Enquanto sistemas de posicionamento por satélite como GPS (EUA) ou GALILEO (Europa), por um lado, conseguem precisar a posição do usuário em até centímetros em um ambiente outdoor, por outro há uma dificuldade deste sistema em calcular o posicionamento em lugares fechados, devido basicamente à atenuação dos sinais causada pelas paredes e seus materiais.Tendo em vista o crescimento das cidades e consequente aumento no número de construções as pessoas cada vez passam mais tempo em ambientes fechados. A necessidade de serviços de localização indoor tem se tornado cada vez mais crítica[2]. Respondendo a essa necessidade surgiram alternativas para o posicionamento em ambientes fechado, entre eles há o uso da Identificação por Rádio Frequência (RFID ? Radio Frequency IDentification), do bluetooth, do Zigbee ou do Wi-fi. A determinação de posicionamento via Wi-fi é uma tecnologia que usa os sinais modulados do Wi-fi para; detectar a presença

de um aparelho, em seguida o sistema é capaz de triangular a posição deste aparelho a partir dos sinais recebidos pelo ponto de acesso [1]. Um dos primeiros exemplos de um sistema de posicionamento utilizando Wi-fi foi RADAR, desenvolvido pela Microsoft, que também criou o RightSPOT que utilizava um ranking das frequências moduladas pelos pontos de acesso, no lugar de usar a potência dos sinais para determinar a posição do aparelho. Tendo em vista este cenário e as condições tecnológicas atuais, nosso projeto procura apresentar uma solução para localização de pessoas em ambientes fechados, como shoppings e eventos em galpões. Para tal, será combinado o usa de Big Data à tecnologia de Wi-fi já citada e ainda implementando um algoritmo de machine learning. O Big Data, que possui como uma de suas principais características a capacidade de armazenar uma grande quantidade de dados, será usado para armazenar todas as leituras de intensidade dos sinais feitas durante uma fase inicial, chamada de treinamento. À partir dessa larga quantidade de dados recebidos serão empregados algoritmos de machine learning processando-os para permitir que o sistema consiga determinar a posição do usuário dentro do ambiente mapeado anteriormente. Esta abordagem se mostra interessante ao ponto de que sua implementação não necessita configuração particular na rede que será usada, uma vez que se baseia em leituras feitas pelo aparelho móvel e no processamento dos dados feitos em um servidor em nuvem. Também se destaca o fato de que em decorrência do machine learning à medida de que o sistema é usado em uma determinada localidade a precisão tende a aumentar, uma vez que cada vez há mais dados para serem consultados para ?Aprendizado?. Esse approach difere de outro também amplamente empregado em esquemas de localização indoor, que é o de modelar o próprio sinal de Wi-Fi, deduzindo a sua distância a partir da intensidade medida, como em [4] e [5]. Esse método se torna ineficiente porque se torna necessário que sejam conhecidas as coordernadas de todos as APs locais, e também o próprio modelo de atenuação dos sinais RSSI de WiFi sofrem todo o tipo de interferência tornando difícil a obtenção de uma boa precisão.

2 MACHINE LEARNING

2.1 Tratamento dos dados

2.1.1 Dados obtidos do Repositório UCI

Para grande parte dos testes de ***Cross-Validation***, foi usado ***dataset*** UJIIndoorLoc apresentado em (TORRES-SOSPEDRA et al., 2014). O ***dataset*** consiste em diversas medidas de potência de sinal Wi-Fi medidas com diversos aparelhos celulares em 3 prédios diferentes de uma faculdade. As medidas estão separadas por prédio, andar e sala. Para os fins dessa monografia, serão realizados testes com medidas sempre de um mesmo andar, afim de classificar as zonas de um mesmo ambiente.

2.1.2 Dados do Servidor

2.2 Formato dos dados tratados

A matriz de dados tratados usada diretamente pelos algoritmos de ML tem o seguinte formato:

$$
\begin{array}{cccccc}
ZoneID & BSSID_1 & BSSID_2 & ... & BSSID_n & \\
\left(\begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ 1 \end{array} \right. & \begin{array}{c} \text{-70} \\ \text{-89} \\ \text{-28} \\ \vdots \\ \text{-48} \end{array} & \begin{array}{c} \text{-92} \\ \text{-80} \\ \text{-120} \\ \vdots \\ \text{-36} \end{array} & \begin{array}{c} ... \\ ... \\ ... \\ \ddots \\ ... \end{array} & \begin{array}{c} \text{-87} \\ \text{-63} \\ \text{-35} \\ \vdots \\ \text{-29} \end{array} & \left. \begin{array}{c} Measure_1 \\ Measure_2 \\ Measure_3 \\ \vdots \\ Measure_n \end{array} \right)
\end{array}
$$

Cada coluna representa a medida de uma *BSSID* (i.e. um Roteador), ou seja, uma *feature* para o ML, e cada linha é um ponto de treino (i.e. a tupla de medidas para cada *BSSID*, nossas *features*.

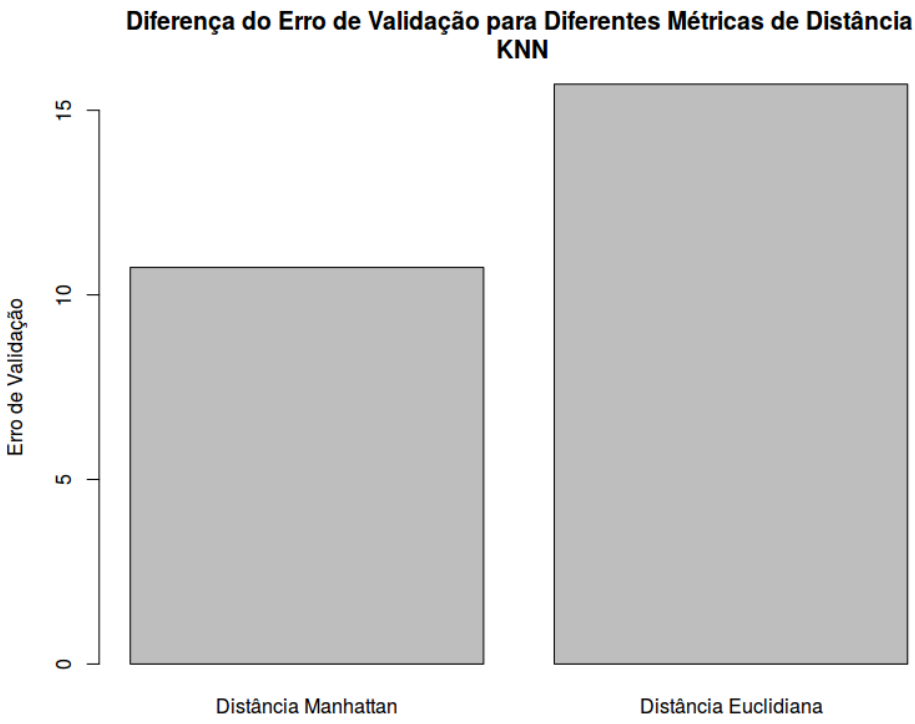
2.3 Modelos Usados e *Cross-Validation* de parâmetros

A chamada *Cross-Validation* dos modelos de ML é o teste para encontrar os parâmetros ótimos para o treinamento. Um dos métodos escolhidos de *Cross-Validation* foi o *K-Fold*. O método *K-Fold* divide o dataset em K subconjuntos de igual tamanho e então um dos conjuntos é usado como validação do treinamento feito pelos $K - 1$ subconjuntos restantes. O processo é repetido K vezes e em cada subdivisão possível podem ser usados valores de parâmetros de treino distintos. O restante dos testes foram feitos da maneira padrão em sistemas de ML: Dividindo-se o ***dataset*** em 80% de pontos de treino e 20% de pontos de validação, e então realizando respectivamente o treino e os testes com esses subconjuntos. Para essa última modalidade de *Cross-Validation* foram feitos também testes em que se usava o subconjunto de 20% para testes, mas uma quantidade crescente dos pontos de treino para o treinamento dos modelos (e.g. 10,15,20,25,30,35,40 ... até o total dos 80%), e em cada treinamento calcula-se a taxa de erro.

2.3.1 KNN : K-Nearest-Neighbors

Para o algoritmo de KNN, foram feitos dois testes de *Cross-Validation*. No primeiro, a métrica de distância foi decidida, e no segundo, o número de vizinhos (K). Para que fosse definida a métrica de distância, foi usado o método *K-Fold* com $K = 2$. E em cada *fold* o modelo foi treinado com uma métrica diferente. Foram calculados 20 conjuntos diferentes de *folds* e foi tirada a média do erro de validação para cada uma das métricas.

Figura 1: Erros para uso de Diferentes Distâncias



Depois, foi usado o método *K-Fold* com $K = 10$ para a decisão do número de vizinhos. Foram testados os valores de 1 até 10 para o número de vizinhos. Na imagem a seguir vemos detalhes desse teste.

Figura 2: Erro em função do número de vizinhos

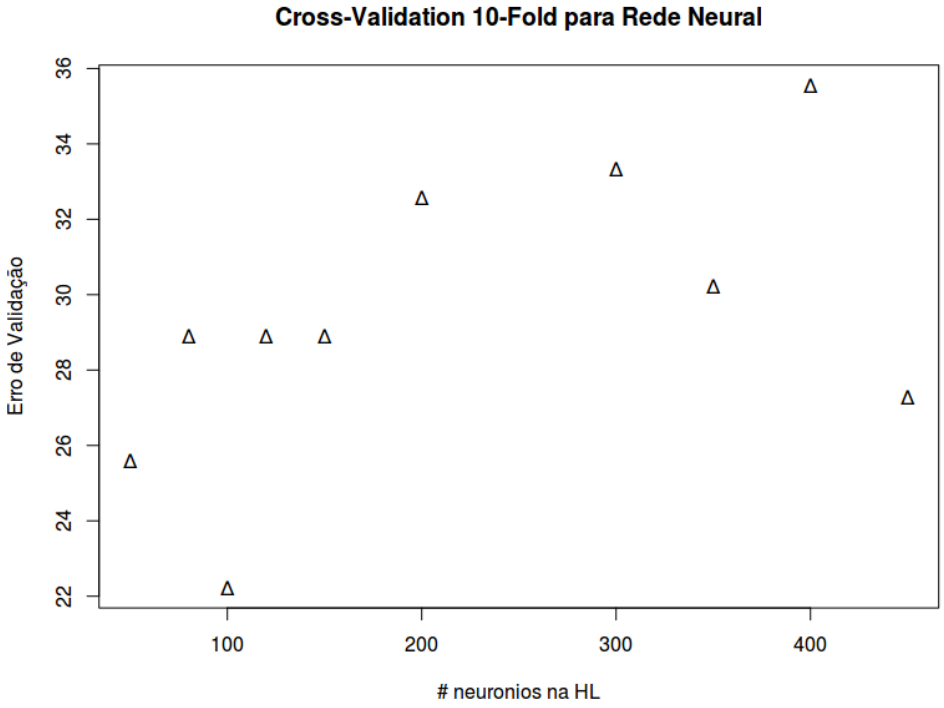
Podemos concluir então que (1) o erro é menor no geral com a distância de Manhattan e (2) o valor de K que minimiza o erro fica proximo de 4 vizinhos. No restante do trabalho foi escolhido o valor de $K = 4$ vizinhos.

2.3.2 Rede Neural

Para a *Cross-Validation* das Redes Neurais, também foi usado o método de *K-Fold* com $K = 10$. Para cada uns dos *folds* foi testado um número de neurônios em uma *hidden-layer* única. (Outros testes mostraram não valer a pena para o nosso problema usar mais de uma camada de *hidden layer* ou *deep learning*). Os resultados são mostrados a seguir:

Concluímos que o número ideal de neurônios na *hidden-layer* está próximo de 200. E é esse valor que usaremos daqui para frente.

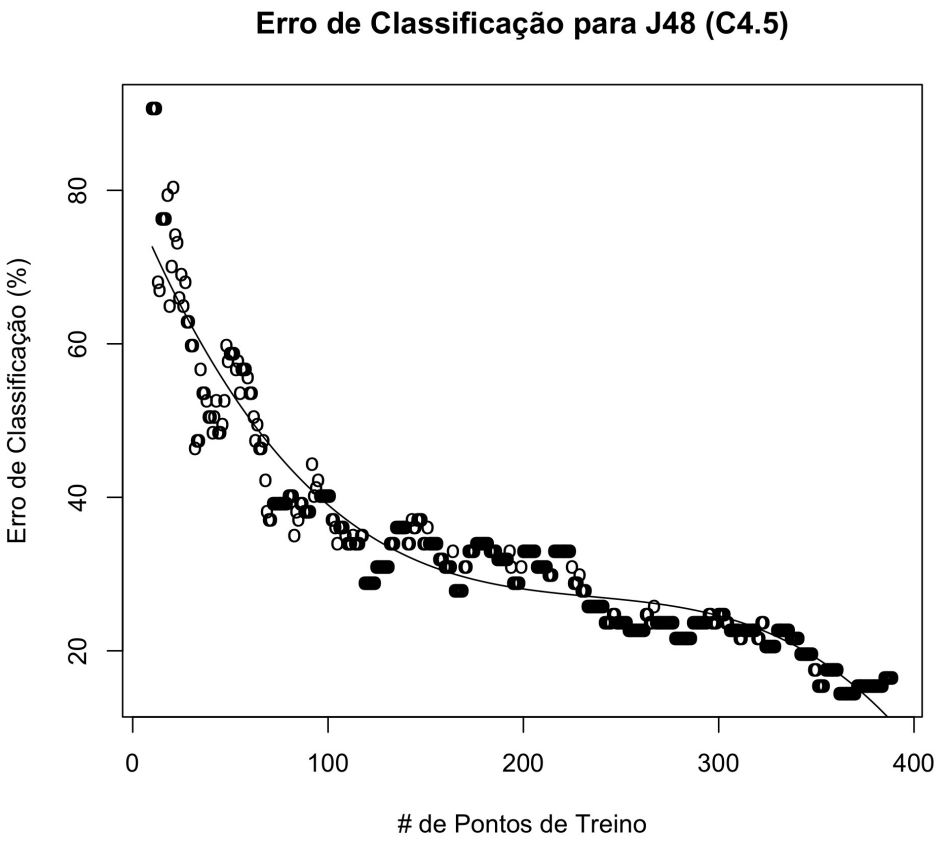
Figura 3: Erro para diversas quantidades de neurônios na rede neural.



2.3.3 Arvore de decisão

Baseado no trabalho apresentado por (BOZKURT et al., 2015), foi escolhida uma implementação do algoritmo C4.5 (QUINLAN, 1993) em Java, da biblioteca Weka, para serem geradas árvores de decisão. Os métodos porém foram chamados pela interface da Weka para R, chamada RWeka. A função J48 do Weka foi testada com o os pontos de treino referentes ao primeiro andar do prédio 1 (i.e. BuildingID 1, FloorID 0) do dataset UJIIndoorLoc. Foi levantada a curva de erro de classificação para uma quantidade crescente de pontos de treino (como explicado em 2.3).

Figura 4: Erro em função do número de vizinhos



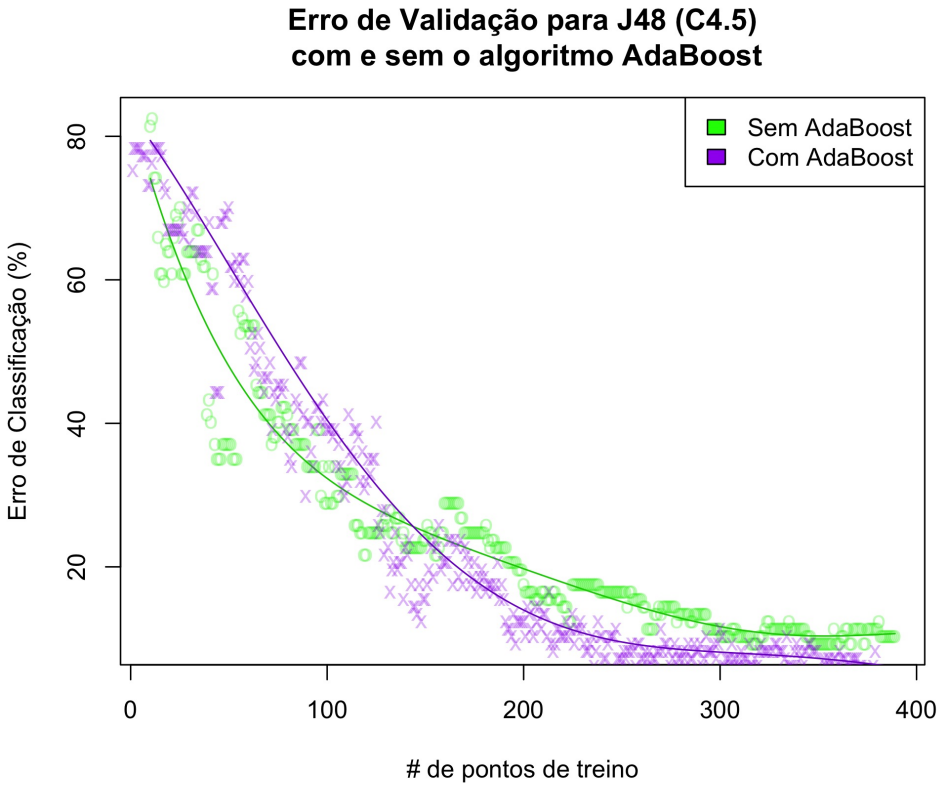
2.3.3.1 Acoplamento com o Algoritmo AdaBoost

Boosting é a ideia em ML de criar uma predição forte e precisa combinando diversas previsões mais fracas. O algoritmo AdaBoost (FREUND; SCHAPIRE, 1996) é um método para melhorar a precisão de classificadores fracos (i.e. *Weak Learners*) por meio de uma votação ponderada , que, por prova, se torna um classificador forte (SCHAPIRE,). Por definição, um classificador fraco é aquele que consistentemente consegue ser melhor que um chute para a classificação (e.g. o erro é menor que 50% para o caso de duas classes possíveis de classificação).

Também baseados em (BOZKURT et al., 2015). Usaremos o algoritmo AdaBoost e sua implementação na biblioteca Weka para melhorar a precisão das árvores de decisão C4.5. A seguir vemos um gráfico comparando a classificação para o mesmo andar do

dataset UJIIndoorLoc com os mesmos pontos de teste e uma quantidade crescente de pontos de treino, assim como feito no tópico anterior.

Figura 5: Comparação do C.45 com o uso do AdaBoost



Podemos notar que com um número grande de pontos de treino, obtemos uma melhora sensível de precisão para a classificação, e portanto, no restante do trabalho, iremos usar o algoritmo C.45 acoplado com o algoritmo AdaBoost.

2.4 Métodos de Votação

Os chamados sistemas de *Ensemble Learning* são aqueles em que uma classificação é feita com base em diversos modelos treinados. Em tópicos anteriores foi discutido o uso do algoritmo AdaBoost, que é um método desse tipo para melhorar o poder de classificadores fracos treinando iterativamente diversos modelos fracos progressivamente. Porém, para agregar os modelos usados até agora, iremos implementar algo

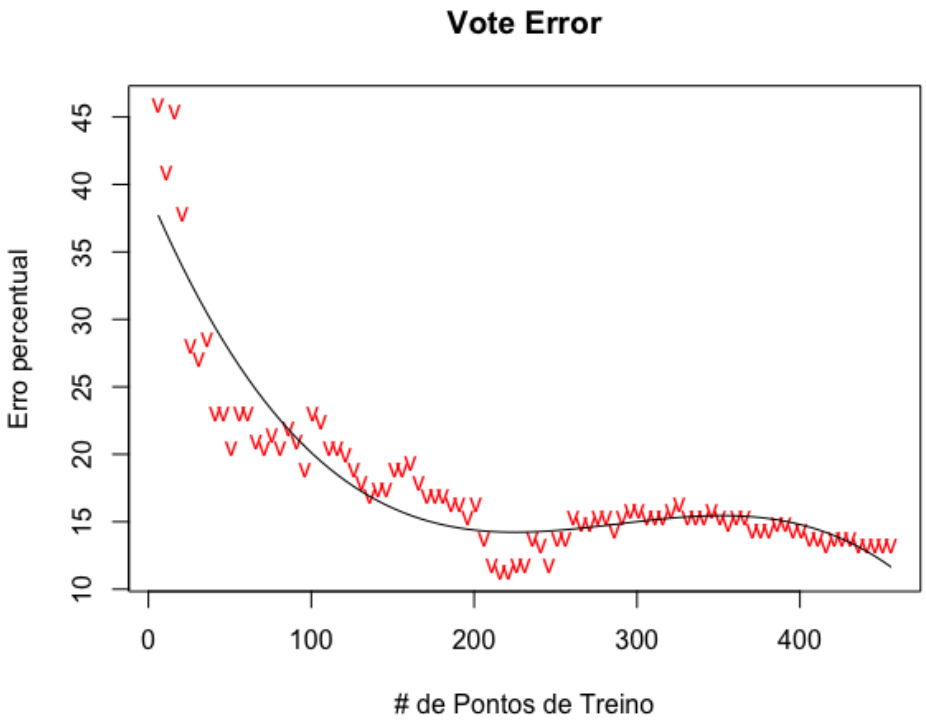
mais simples, porém seguindo a mesma lógica: Sistemas de Votação.

2.4.1 Votação Simples

Na votação simples os resultados da classificação de todos os modelos são contabilizados e é escolhido aquele com o maior número de ocorrências entre os classificadores. Os testes foram feitos calculando-se o erro de validação no mesmo dataset de treino com uma quantidade cada vez maior de pontos de treino, estes retirados de um dataset de treino também fixado.

A votação simples foi primeiramente testada com os algoritmos de Rede Neural, KNN e SVM com zonas escolhidas aleatoriamente do ***dataset*** UJIIndoorLoc

Figura 6: Erro de votação simples para uma quantidade crescente de pontos de treino.



Para comparação, foram calculados os erros de validação independentes de cada um dos algoritmos usados.

Figura 7: Erro do algoritmo KNN para uma quantidade crescente de pontos de treino.

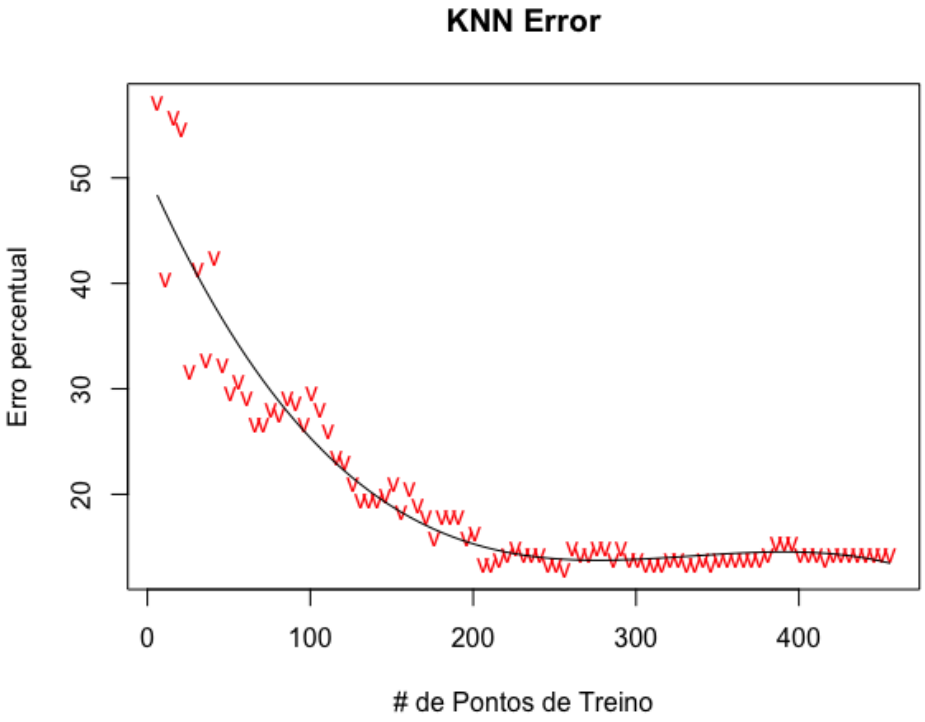


Figura 8: Erro da Rede Neural para uma quantidade crescente de pontos de treino.

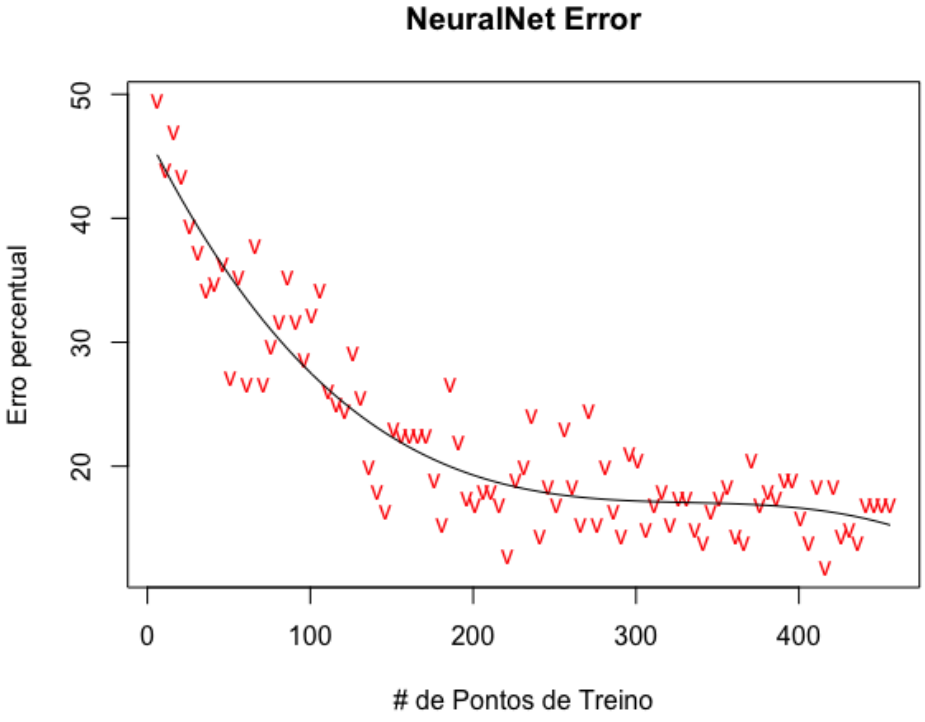
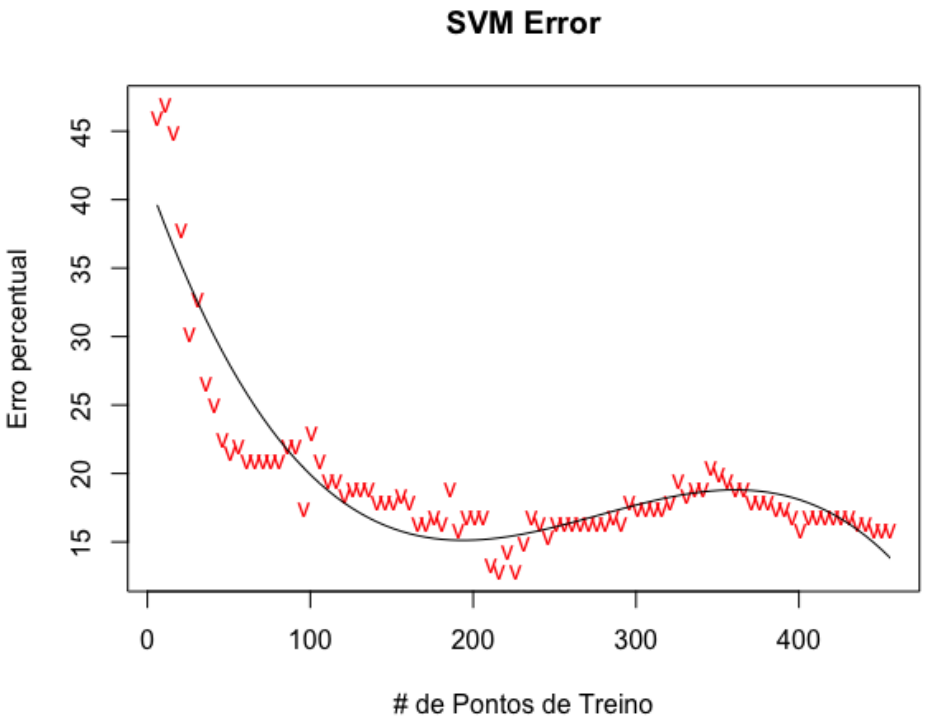


Figura 9: Erro do algoritmo SVM para uma quantidade crescente de pontos de treino.



REFERÊNCIAS

BOZKURT, S.; ELIBOL, G.; GUNAL, S.; YAYAN, U. A comparative study on machine learning algorithms for indoor positioning. In: *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on*. [S.l.: s.n.], 2015. p. 1–8.

FREUND, Y.; SCHAPIRE, R. E. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1996.

QUINLAN, R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

SCHAPIRE, R. E. *Explaining AdaBoost*.

TORRES-SOSPEDRA, J.; MONTOLIU, R.; MARTíNEZ-USó, A.; AVARIENTO, J. P.; ARNAU, T. J.; BENEDITO-BORDONAU, M.; HUERTA, J. Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems. In: *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*. [S.l.: s.n.], 2014. p. 261–270.