DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Redes Neurais para Apoio à Análise de Performance de Atletas Profissionais de Futebol

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Gabriel Ratto Goulart

Florianópolis, julho de 2019

Redes Neurais para Apoio à Análise de Performance de Atletas Profissionais de Futebol

Gabriel Ratto Goulart

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Marcelo Stemmer

Banca Examinadora:

Robson Carlos da Motta
Orientador na Empresa

Prof. Marcelo Stemmer
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Prof. Cleber Jorge Amaral, Avaliador

Luis Artur Kretzer Tavares Sobral, Debatedor

Henrique Montagna Hartwig, Debatedor

"The only mode of obtaining an answer to these questions of the senses is to forego all low curiosity, and, accepting the tide of being which floats us into the secret of nature, work and live, work and live, and all unawares the advancing soul has built and forged for itself a new condition, and the question and the answer are one."

(Ralph Waldo Emerson)

Resumo

A aplicação da tecnologia no futebol vêm aumentando muito nos últimos anos, principalmente na captura de dados para análise de performance dos atletas. Em ligas de alto nível, clubes utilizam dispositivos de GPS de frequência de captura maior que 10 Hz. Estes dispositivos contudo, possuem um alto custo que impossibilita atletas amadores e clubes de médio porte de terem uma mesma inteligência sobre sua performance. Com o objetivo de possibilitar que estes atletas e clubes pudessem ter análises de performance com qualidade similar a clubes de alto poder aquisitivo somente com *smartwatches* e *smartphones*, a Joga optou pelo desenvolvimento de algoritmos de aprendizado de máquina capazes de suprir a perda de informação que estes dispositivos apresentam por ter frequência de captura de 1 Hz. Assim, em posse de uma base de dados de GPS de 10 atividades de futebol capturadas por um dispositivo de 10 Hz, foi feita uma amostragem destes dados em frequências distintas a fim de simular dados capturados por dispositivos diferentes. Em seguida, foi criada uma rede neural capaz de estimar o quanto a distância calculada com pontos amostrados deveria ser corrigida para ser similar à distância mensurada com dispositivos de 10 Hz de frequência. Além disso, foi aplicado um filtro de Kalman nos dados de GPS a fim de minimizar ruídos de medição. Por fim, foi realizado um experimento controlado onde atletas percorreram um circuito de distâncias conhecidas com um *smartwatch* de GPS de 1 Hz e um dispositivo de GPS de 15 Hz a fim de avaliar a eficácia do modelo desenvolvido. A distância medida pelo modelo final de rede neural com filtro de Kalman aplicado nos dados de 1 Hz provou ter um erro menor do que a distância calculada com dados de 15 Hz.

Palavras-chave: aprendizado de máquina, futebol, rede neural, filtro de Kalman, GPS.

Abstract

The usage of technology in football has been increasing over the last few years, specially in the capture of data for athlete's performance analysis. On top level leagues, clubs use GPS devices with a frequency of 10 Hz. These devices however, are too expensive, making impossible for amateur athletes and medium-level clubs to have the same kind of intelligence over their performance. With the goal of enabling these athletes and clubs to have performance analysis of similar quality of top level clubs using only smartphones and smartwatches, Joga opted for the development of machine learning algorithms capable of making up for the loss of information that these devices present for having a frequency of 1 Hz. Thus, with a GPS database of 10 football activities recorded with a 10 Hz device, a sampling was made with different frequencies in order to simulate data from the same activities recorded with different devices. A neural network was then created, capable of estimating how much the distance calculated between sampled location points should be corrected in order to be similar to the distance measured with 10 Hz data. A Kalman Filter was also applied to the GPS data in order to minimize measurement noise. At last, a controlled experiment was made where athletes ran iver a circuit of known distances with a smartwatch of a 1 Hz GPS and a 15 Hz GPS device in order to evaluate the accuracy of the developed model. The distance measured by the final neural network model with the Kalman Filter on 1 Hz data proved to have a smaller error than the distance measured by 15 Hz data.

Keywords: machine learning, football, neural network, Kalman filter, GPS.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FIFA: Federação Internacional de Futebol

CBF: Confederação Brasileira de Futebol

GPS: Global Positioning System

Hz: Hertz

m: metro

m/s: Metro por Segundo

km: Quilômetro

km/h: Quilômetro por Hora

JSON: JavaScript Object Notation

Sumário

1 Introdução

A Joga é uma empresa que fornece análises de performance para atletas profissionais e amadores de futebol através da captura e processamento dos dados de GPS gerados a partir de dispositivos *wearables* que os jogadores utilizam junto ao corpo durante as partidas.

Visto que a enorme maioria dos dispositivos *wearables* do mercado, celulares e *smartwatches* possuem uma frequência de captura de 1 Hz, há uma grande perda de informação que prejudica a análise de performance dos atletas, dado que o futebol é um esporte de bruscas movimentações. Tentando oferecer um serviço de qualidade em um mercado competitivo e emergente como este e com o objetivo de proporcionar inteligência de ponta para qualquer equipe ou atleta independente de seu poder aquisitivo, a Joga entendeu que seria melhor suprir esta perda de informação através de ferramentas de engenharia e matemática do que exigir do usuário um dispositivo com uma frequência de captura maior, mas com um mais alto custo.

O objetivo deste trabalho é o desenvolvimento de uma rede neural em linguagem Python capaz de suprir esta perda de informação através da estimação da movimentação real dos jogadores utilizando dados de dispositivos de 10 Hz como base de treinamento. Como objetivo mais específico, há a necessidade de manter a diferença relativa entre a distância mensurada por um GPS de 10 Hz e a calculada pela rede neural em no máximo 15%, sendo também desejável a manutenção do erro do cálculo distância percorrida em alta velocidade (uma métrica de alta relevância para fisiologistas) em no máximo 33%.

1.1 Justificativa

Instrumentos de medição com melhor eficácia e resolução possuem uma melhor capacidade de prover dados de melhor qualidade. Porém, no contexto específico de dispositivos de GPS, tais instrumentos, com uma frequência de captura de GPS acima de 5 Hz, possuem um custo muito maior do que aqueles com frequência de captura de 1 Hz, maioria do mercado. Celulares e *smartwatches*, hoje presentes em uma parcela significativa da população, já possuem um GPS integrado com uma frequência de captura de 1 Hz. Assim, visando proporcionar o mesmo serviço de análise de performance para qualquer atleta que tenha um celular ou um *smartwatch*, a Joga procurou melhorar seus algoritmos de análise de dados a fim de oferecer uma inteligência de alta qualidade para todo o espectro de atletas de futebol, de amadores a profissionais, de escolinhas a categorias de base. Por fim, ao se posicionar como uma empresa de análise de desempenho para qualquer atleta que tenha um destes dispositivos, a Joga se diferencia de seus concorrentes que em sua

maioria exigem que o atleta gaste uma quantia considerável em aparelhos específicos para o uso em um jogo de futebol.

Além disso, procurando manter-se na vanguarda de inovações tecnológicas, a Joga entendeu que o uso de inteligência artificial pode diferenciar o seu produto, visto que ela pode ser aplicada a inúmeras outras áreas de análise de desempenho de um jogo de futebol, desde a identificação de lesões até o reconhecimento de padrões de falhas táticas.

1.2 Estrutura do Documento

O trabalho é organizado em diferentes capítulos da seguinte maneira:

- O capítulo 2 introduz a empresa e a formalização do problema dos quais este trabalho está inserido;
- O capítulo 3 aborda a fundamentação teórica das ferramentas utilizadas neste projeto;
- O capítulo 4 apresenta o desenvolvimento do projeto;
- O capítulo 5 apresenta o resultado da implementação do projeto;
- Por fim, o capítulo 6 apresenta a conclusão do trabalho feito ao longo do programa do Projeto de Fim de Curso e perspectivas a respeito de trabalhos futuros.

2 Futebol e Tecnologia

2.1 Contexto

O futebol é o esporte mais popular do mundo. De acordo com um estudo da FIFA realizado em 2006, cerca de 265 milhões de pessoas praticam o esporte ao redor do mundo [1]. Dos campos profissionais até traves feitas com chinelos, das bolas de material sintético criadas com alta tecnologia até bolas feitas de meia utilizadas por crianças, o futebol é um esporte que transcende barreiras econômicas, culturais e sociais.

Mesmo com algumas modificações em suas regras desde a sua criação no século XIX, o futebol, em sua essência, pouco foi alterado. Porém, a maneira pela qual o esporte interage com outros aspectos de nossas vidas têm mudado de forma drástica com a evolução tecnológica. O futebol profissional tornou-se uma indústria multibilionária [2] e altamente competitiva, e com esta brusca mudança foi natural que tecnologia de ponta fosse criada para ajudar os clubes de futebol a serem mais bem sucedidos. Seja na interação com seus torcedores, seja na avaliação de desempenho de seus atletas, técnicas que eram utilizadas em indústrias de engenharia foram rapidamente incorporadas na cultura executiva do futebol.

Podemos traçar a origem da aplicação de tecnologia no futebol para análise de desempenho de equipes na década de 1950, quando um veterano inglês da II Guerra Mundial passou a registrar todas as ações de mais de 2200 jogos de futebol, incluindo a final da Copa do Mundo de 1958, vencida pelo Brasil [3]. A aplicação de análise estatística para otimizar a performance tática da equipe já havia sido inaugurada, o próximo passo foi então, tentar otimizar a performance física dos atletas.

Com a queda do custo de fabricação de tecnologia de ponta, novas abordagens foram aplicadas ao esporte, sendo uma delas a de avaliação de desempenho físico. Com o intuito de otimizar a performance física dos atletas profissionais, clubes e seleções passaram a investir em maneiras inovadoras de mensurar o desempenho físico dos atletas em jogos e treinos. Seja com medidores de batimentos cardíacos ou por dispositivos de GPS, jogadores passaram a utilizar equipamentos junto ao corpo capazes de captar dados relevantes para a equipe de preparação física.

2.2 Fisiologia do Futebol

A distância total percorrida por um jogador de futebol durante uma partida é uma das principais medidas de performance física deste tipo de atleta, que pode correr entre

9000 m e 12000 m jogando em alto nível, sendo que entre 75% e 85% desta distância é percorrida em baixas velocidades [4]. Movimentações feitas em altas velocidades, também conhecidas como eventos de alta intensidade, são importantes medidas de desgaste físico do jogador, visto que atletas se exercitando em alta intensidade podem apresentar uma inabilidade de manter a performance por muito tempo [5]. Além disso, foi mostrado que há uma correlação significante entre distâncias percorridas em alta intensidade e o nível de ácido lático no sangue, representando um baixo nível de oxigenação sanguíneo [6].

Não há um consenso claro entre os times profissionais de futebol na definição de um limite do que seria uma velocidade baixa ou alta, com alguns clubes trabalhando com um limite entre 12 km/h e 16 km/h. Muitas pesquisas recentes definem 14 km/h (cerca de 4 m/s) como a menor velocidade possível de um evento de alta intensidade [7], sendo um valor trabalhado também por fisiologistas de alguns clubes brasileiros. Assim, foi denifido como 14 km/h o limiar de velocidade de eventos em alta intensidade para o desenvolvimento deste trabalho.

Como o futebol é composto por movimentações bruscas que podem mudar de direção em qualquer instante, a mensuração da velocidade de um atleta é um desafio complexo. Uma das maneiras mais eficazes de se mensurar a velocidade de um jogador é através de captura de vídeo [8]. Como este método exige o uso de tecnologias de alto custo, além de não ser escalável e de difícil portabilidade, o uso de dispositivos de GPS tornou-se a principal alternativa para clubes profissionais com o objetivo de análise de performance de seus atletas. O grande problema destes dispositivos é que sua eficácia depende da sua frequência de captura. Em experimentos controlados, dispositivos de GPS com frequência de captura entre 1 e 5 Hz subestimam a distância total por entre 3.7% e 5.7%, além de subestimar a velocidade média e máxima por entre 10% e 30%, com o erro sendo mais alto em altas velocidades e em frequentes mudanças de direção [9]. Este alto erro em mudanças instantâneas de velocidade é resultado direto da limitada frequência de captura de GPS, sendo menor em dispositivos com frequência de 10 Hz [10].

O grande problema é que dispositivos com maior frequência de captura possuem um custo muito maior do que aqueles de 1 Hz, amplamente disponibilizados no mercado, presentes em *smartphones* e *smartwatches*. Como uma boa análise de performance é impossível sem dados adequados, o alto custo de dispositivos de GPS de 10 Hz cria uma barreira de entrada para clubes brasileiros e latino-americanos que desejam otimizar o trabalho de seus fisiologistas, mas que não possuem tantos recursos financeiros para tal. Foi nesse contexto que nasceu a empresa Joga.

2.3 A Joga

A Joga é uma empresa que oferece análise de performance para atletas profissionais e amadores de futebol através da captura e processamento dos dados de GPS gerados a partir de dispositivos que os jogadores utilizam junto ao corpo durante as partidas.

A Joga nasceu com o objetivo de oferecer a mesma inteligência de análise de dados para clubes profissionais e atletas amadores interessados em melhorar seu próprio desempenho físico e tático. Com o intuito de se tornar a maior empresa de inteligência de futebol no Brasil em 2020, a Joga foca em duas vertentes. No futebol profissional, equipes podem utilizar *smartwatches* para captura de dados de GPS e acelerômetro durante treinos e partidas e de uma plataforma virtual, onde o resultado da análise de desempenho dos dados adquiridos é disponibilizada em tempo real para a equipe de preparação física do clube. Já no futebol amador, há o aplicativo *mobile* Joga, no qual o usuário pode importar dados de GPS coletados de *smartwatches* ou simplesmente jogar com o celular junto ao corpo, seja com braçadeiras ou com coletes vendidos exclusivamente pela Joga para este propósito, para enfim ter uma noção de seu próprio desempenho físico e tático.

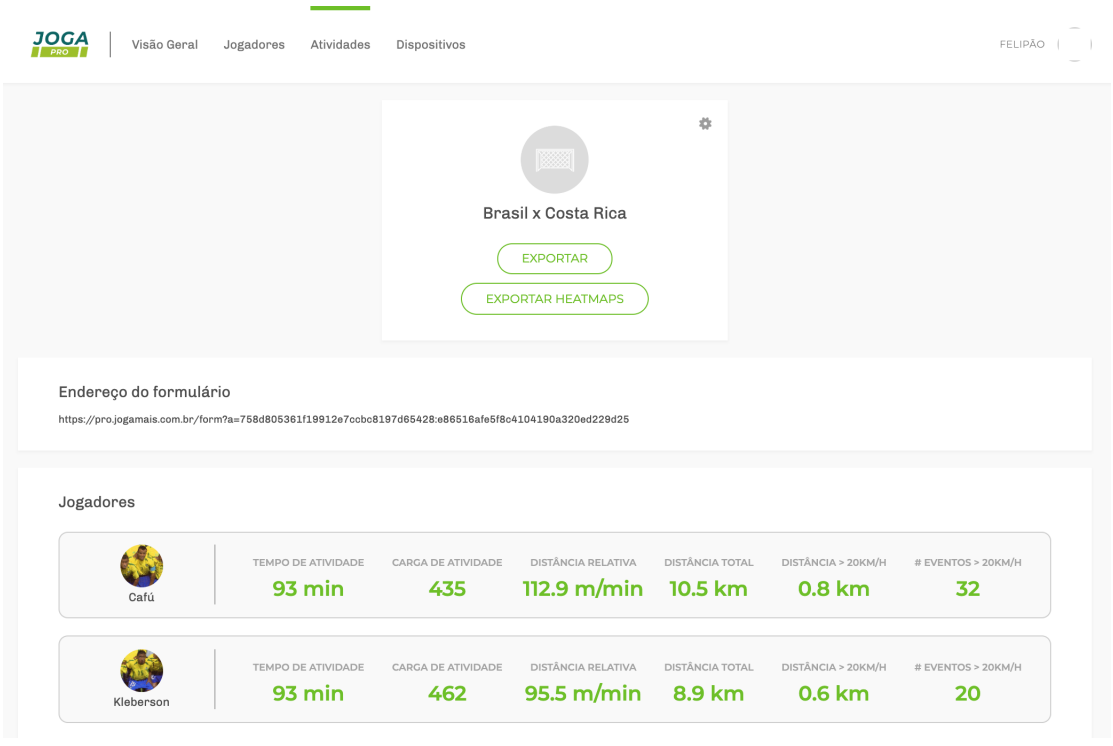

Figura 1 – Plataforma de análise de performance profissional da Joga.

Atletas profissionais dos clientes da Joga utilizam *smartwatches* Samsung Gear Fit durante treinos e partidas, capturando dados de GPS com frequência de 1 Hz e dados de acelerômetro. Estes dados são então processados pelos algoritmos desenvolvidos pela Joga para extrair métricas de performance como distâncias percorridas e arrancadas, além do mapa de calor de movimentação do atleta, que serão disponibilizados aos departamentos

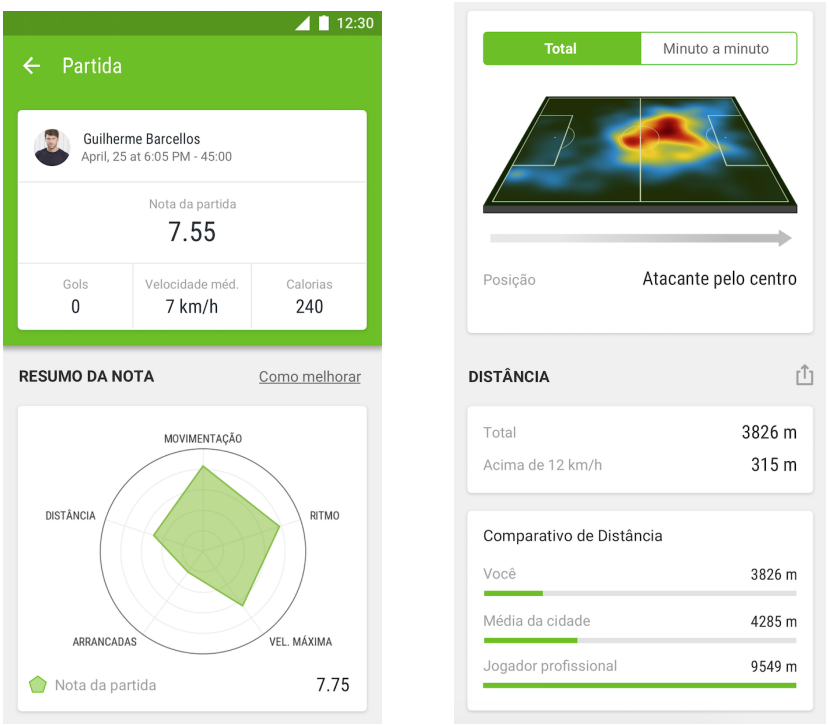

Figura 2 – Aplicativo Joga para Android

de fisiologia dos respectivos clubes através de uma plataforma *web*, visualizada na figura 1. Atualmente, a Joga oferece serviços para dez clubes profissionais que disputam as séries B, C e D do Campeonato Brasileiro de Futebol organizado pela CBF.

Esta análise de performance também pode ser oferecida para jogadores amadores de futebol, ao utilizar o aplicativo *mobile* da Joga, visualizado na figura 2, seja em *smartphones* ou *smartwatches*. Até o mês de junho deste ano, cerca de 2800 usuários já haviam realizado uma partida de futebol utilizando o aplicativo da Joga.

2.4 Formalização do Problema

Foi visto na seção 2.2 que a mensuração das distâncias total e percorrida em alta intensidade é de suma importância para a avaliação fisiológica de um jogador profissional de futebol. A eficácia da mensuração destas métricas por dispositivos de GPS varia de acordo com a sua frequência de captura. Como a diretriz da empresa é a de utilizar dispositivos de GPS de baixo custo (com frequência de 1 Hz), foi decidido compensar a falta de dados provinda da baixa frequência de captura destes dispositivos através de algoritmos de aprendizado de máquina. Para que esta compensação fosse feita de forma eficiente, foi necessário antes entender de forma objetiva as diferenças das métricas finais de atividades capturadas de forma simultânea por dispositivos de GPS de 1 Hz utilizados pela Joga e dispositivos de GPS com frequência de captura mais alta.

Assim, em posse do dispositivo de GPS K-sport com frequência de captura de 10

Atividade	Distância Total K-sport (m)	Distância Total Gear Fit (m)	Diferença Relativa
1	1171	1463	24.93%
2	1038	1168	12.52%
3	3438	3727	8.41%
4	3007	2533	15.76%
5	2004	2535	26.49%
6	2680	3228	20.45%
7	1076	1271	18.12%
8	982	1328	35.23%
9	1533	1492	2.67%
10	1259	1584	25.81%
11	999	1231	23.22%
		Média	19.42%

Tabela 1 – Diferenças entre as distâncias totais mensuradas pelo K-sport (10 Hz) e o Samsung Gear Fit (1 Hz)

Hz e de um Samsung Gear Fit, foram capturados dados de 11 atividades diferentes de futebol, sendo quatro delas atividades de um jogador profissional do Figueirense Futebol Clube, equipe de Florianópolis, que participou de treinos utilizando um colete especial para abrigar os dispositivos de GPS. As outras sete atividades foram coletadas por atletas da equipe de futebol da UFSC.

As diferenças entre as distâncias de cada dispositivo é visível na tabela 1. Foi calculado que a média das diferenças relativas entre a distância mensurada pelo K-sport e pelo Samsung Gear Fit foi em torno de 19.24%. Esta tabela aponta apenas a diferença no cálculo final da distância entre dispositivos de frequências diferentes. De maneira similar, a tabela 2 mostra as diferenças relativas no cálculo da distância em alta intensidade (definida anteriormente como distância percorrida acima de 14 km/h) entre os dois dispositivos.

É importante ressaltar de que não sabemos qual foi de fato a distância percorrida pelo jogador de cada atividade, portanto não há como saber o erro real de cada um dos dispositivos. Contudo, como visto na seção 2.2, dispositivos de maior frequência tendem a ter uma melhor acurácia. Além disso, muitos artigos da literatura subestimam as diferenças das medições entre dispositivos pelo fato deles serem utilizados em experimentos controlados, que não replicam as movimentações bruscas de atividades reais de futebol.

Por fim, para o desenvolvimento do projeto, foi considerado como objetivo principal a minimização da diferença relativa entre a distância mensurada pelo Samsung Gear Fit e a mensurada pelo dispositivo K-sport para abaixo de 15%, tentando aproximar mensurações de dados de 1 Hz e 10 Hz. Como objetivo secundário, definiu-se a minimização da diferença relativa do cálculo da distância percorrida em alta intensidade para menos de 33%, mantendo-se dentro do desvio padrão dos experimentos feitos por [11] com GPS de 10 Hz.

Atividade	Distância em Alta Intensidade K-sport (m)	Distância em Alta Intensidade Gear Fit (m)	Diferença Relativa
1	200	161	19.50%
2	60	42	30.00%
3	312	157	49.68%
4	471	153	67.52%
5	235	100	57.45%
6	393	174	55.73%
7	266	195	26.69%
8	276	179	35.14%
9	338	123	63.61%
10	223	169	24.22%
11	284	199	29.93%
		Média	41.77%

Tabela 2 – Diferenças entre as distâncias em alta intensidade mensuradas pelo K-sport (10 Hz) e o Samsung Gear Fit (1 Hz)

2.5 Solução Proposta

A diferença entre as distâncias mensuradas pelos dispositivos supracitados deve-se à diferença da frequência de captura principalmente em movimentações não-lineares, que constituem boa parte do que um jogador de futebol faz durante uma atividade. Para que a Joga pudesse utilizar dispositivos de 1 Hz de frequência com a mesma acurácia de dispositivos de 10 Hz, foi proposta a criação de um modelo de aprendizado de máquina capaz de estimar o quanto uma distância entre dois pontos espaciais com um intervalo de tempo maior do que 0.1 s deveria ser alterada para se assemelhar a uma distância mensurada por um dispositivo de 10 Hz no mesmo intervalo de tempo. Este modelo, ao estimar a distância em um dado instante, deverá se basear em distintas métricas de movimentação de instantes anteriores e posteriores ao momento em questão como mudança de ângulo, tempo entre capturas e até mesmo velocidade medida. Esta proposta foi baseada na hipótese de que a diferença entre as distâncias é maior em movimentações bruscas ou curvilíneas e que estas variáveis influenciam de forma conjunta para o cálculo da distância entre dois pontos de uma maneira não trivial, tornando aprendizado de máquina uma alternativa atraente para a solução deste problema. Como um jogo de futebol é composto por inúmeras dessas movimentações, esta diferença entre as distâncias vai se acumulando ao longo da partida.

Antes de escolher um método de aprendizado, é necessário definir o que de fato se quer aprender e como o conhecimento será adquirido. Como temos como objetivo a minimização das diferenças das distâncias medidas entre um GPS de 1 e de 10 Hz, seria ideal comparar, instante a instante, o quanto a distância mensurada pelo GPS de 1 Hz se

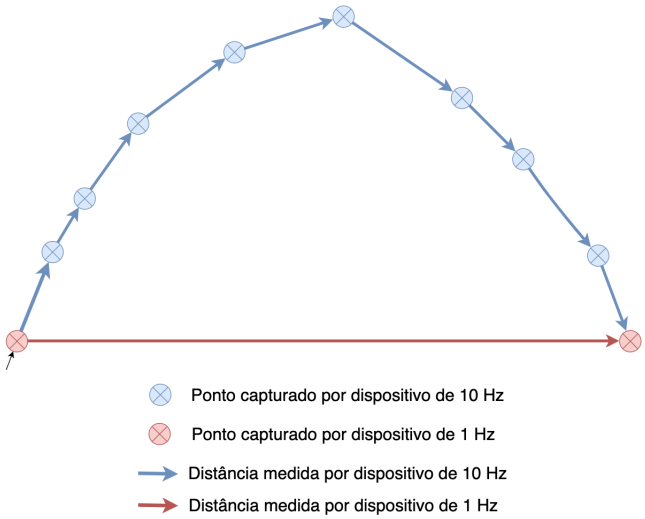

Figura 3 – Ilustração de uma possível diferença entre distâncias calculadas por dispositivos de 1 Hz e de 10 Hz.

diferencia da mensurada pelo GPS de 10 Hz. Contudo, embora os dados de GPS possuam registros do instante de tempo do qual eles foram capturados, há um atraso entre a captura e o registro do dados espaciais que torna a sincronização desses dados difícil de ser feita.

Com esta perspectiva em mente, foi decidido que a melhor abordagem seria a amostragem dos dados de GPS de 10 Hz em períodos de tempo distintos, simulando uma captura dos mesmos dados espaciais mas com uma frequência menor. A distância entre estes pontos amostrados seria então similar à distância mensurada por um dispositivo de menor frequência. Esta abordagem torna a comparação entre as distâncias mensuradas por frequências diferentes mais fiel à realidade, criando uma base de dados ideal para o projeto em questão. Partindo do princípio que diferentes variáveis influenciam de maneiras singulares no cálculo da distância entre dois pontos, foi proposto que o modelo de aprendizado de máquina escolhido fosse uma rede neural, dada a sua eficácia de aprender relações não-lineares entre diferentes variáveis.

Dada a necessidade de utilizar, no treinamento do modelo, dados que na prática não seriam tão similares assim aos utilizados no produto da Joga, este projeto foi dividido em três partes distintas:

- Criação de modelo capaz de estimar distâncias com capturas de GPS amostrados em frequências distintas de dados de 10 Hz do K-sport;
- Validação de que este modelo seria capaz de generalizar para dispositivos diferentes, comparando distâncias totais de atividades registradas com K-sport e Samsung Gear Fit;
- Avaliação final do modelo aplicado em um experimento controlado onde há conhecimento das distâncias percorridas.

3 Fundamentação Teórica

Com a tecnologia cada vez mais presente no nosso cotidiano, a quantidade de dados coletados por dispositivos eletrônicos aumentou de forma exponencial na última década. Em 2013, 90% dos dados da internet foram criados nos dois anos antecedentes [12]. Assim, com uma quantidade de dados colossal, o desafio não só de extrair informações relevantes mas também de descartar aquilo que é ruído tornou-se um dos objetivos mais almejados pela engenharia do início deste século. Porém, com o mundo tornando-se cada vez mais complexo, com dados tornando-se cada vez menos linearmente independentes, as ferramentas matemáticas e estatísticas de inferência tiveram de se adaptar.

Modelos estatísticos mais antigos passaram a se tornar ineficientes em face à esta enorme quantidade de dados disponíveis. A necessidade da criação de modelos capazes não apenas de interpretar uma enorme quantidade de dados em um rápido tempo de processamento, mas também de serem flexíveis o suficiente para inferir relações não-lineares entre múltiplas variáveis, surgiu e extrapolou para outros campos. O problema deixou de ser apenas estatístico, e invadiu áreas de ciências da computação, engenharia da informação, neurociência, linguística, psicologia e filosofia.

É dessa perspectiva que o campo de inteligência artificial vêm crescendo nos últimos anos. Estudos indicam que investimentos neste campo mais do que quadriplicará entre 2017 e 2021 [13], enquanto a área de Aprendizado de Máquina será destino de 60% destes investimentos [14]. A ciência da computação define pesquisa em inteligência artificial como o estudo de "agentes inteligentes", ou seja, qualquer dispositivo ou máquina capaz de reconhecer elementos do ambiente ou contexto do qual ele está inserido e tomar ações que maximizam a chance de alcançar seus objetivos. Coloquialmente, o termo "inteligência artificial"é usado para a descrição de máquinas capazes de performar tarefas similares à funções cognitivas humanas, como aprender, reconhecer padrões ou solucionar problemas. Objetivos tradicionais do campo de inteligência artificial incluem a automatização de raciocínios de inferência, representação de conhecimento, aprendizado, planejamento, processamento de linguagem natural, reconhecimento e manipulação de objetos e até mesmo uma consciência artificial [15].

Assim, é possível distinguir duas frentes de ação desta promissora área: robótica e aprendizado de máquina [15], uma focada na interação da máquina com o mundo real, e a outra na maneira pela qual ela interpreta esta interação. As ferramentas utilizadas neste trabalho são limitadas a aprendizado de máquina.

3.1 Aprendizado de Máquina

Aprendizado de máquina engloba o processo pelo qual sistemas computacionais aprendem programas através de dados [16] com instruções limitadas, baseando-se somente em reconhecimento de padrões e inferências dos dados fornecidos. Assim, o principal objetivo do aprendizado de máquina é generalizar além das observações presentes nos dados de treinamento, além de aprender e otimizar a sua própria lógica de generalização com a entrada de novas informações.

Embora o objetivo de aprendizado de máquina seja, de forma abstrata, o mesmo para todas as suas aplicações, a maneira pela qual ele será implementado varia de situação para situação. Há uma enorme variedade de algoritmos de aprendizado de máquina diferentes, desenvolvidos para solucionar tipos de problemas específicos. De forma geral, é possível dividir estes algoritmos em dois grupos distintos de acordo com o tipo de aprendizagem a ser feita (embora existam algoritmos que tiram proveito destes dois tipos de aprendizado de forma simultânea): aprendizagem não-supervisionada e aprendizagem supervisionada.

3.1.1 Aprendizagem não-supervisionada

Aprendizagem não-supervisionada é uma abordagem utilizada para heuristicamente descobrir e descrever padrões previamente desconhecidos em um conjunto de dados não categorizado [17]. Por isso, tal abordagem é caracterizada como não-supervisionada, visto que não há informação prévia sobre os padrões existentes (ou não) no conjunto de dados.

3.1.2 Aprendizagem supervisionada

Ao contrário da aprendizagem não-supervisionada, na aprendizagem supervisionada há a presença de um supervisor, ou seja, uma informação prévia sobre os dados em questão na qual a máquina pode avaliar a eficácia de seu aprendizado [17]. Os principais algoritmos de aprendizagem se dividem em duas categorias distintas: classificação ou regressão. Em problemas de classificação, para cada observação do conjunto de dados é atribuída uma classe ou categoria diferente e assim, a máquina deverá aprender a classificar uma nova observação em uma destas classes de acordo com as variáveis que compõe esta observação. Em problemas de regressão, para cada observação, ou valor de entrada, há um valor numérico de saída e assim, será objetivo da máquina aprender a estimar um valor de saída para cada conjunto de valores de entrada.

De forma sucinta, algoritmos de aprendizado de máquina são compostos por três componentes distintos [16]:

- Modelo: para cada tipo de algoritmo há uma maneira distinta pela qual o conjunto

de dados de treinamento será interpretado e utilizado pelo algoritmo. Algoritmos de aprendizado de máquina tendem a atualizar este modelo de acordo com o comportamento dos outros dois componentes. A eficácia deste modelo é determinada pelo componente de avaliação do algoritmo.

- Avaliação: cada algoritmo possui uma função de avaliação distinto que varia de acordo com a estrutura de dados e do problema em questão. Esta função mensura a qualidade de aprendizado do algoritmo, comparando as saídas estimadas com as saídas reais do conjunto de dados de treinamento, e assim, ela deverá ser minimizada ou maximizada, dependendo do algoritmo em questão. O processo de minimização ou maximização da função de avaliação é feito pelo componente de otimização do algoritmo.

- Otimização: a maneira pela qual a função de avaliação será otimizada varia de algoritmo para algoritmo. Diferentes métodos de otimização existem para variados problemas, mas a eficácia de um algoritmo de aprendizado de máquina depende muito da escolha de um bom algoritmo de otimização. Com o resultado desta otimização, o algortimo enfim atualiza os parâmetros do modelo.

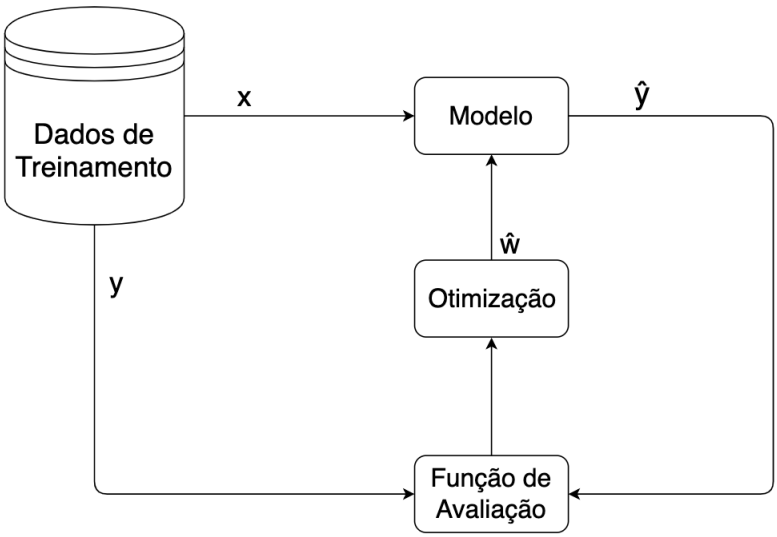

Figura 4 – Fluxograma de um algoritmo de aprendizagem de máquina.

A tabela 3 ilustra exemplos de cada um dos três componentes de um algoritmo de aprendizado de máquina. Muitas vezes há uma metonímia na referência à um algoritmo, tomando-se o nome do modelo como o nome do algoritmo em si, visto que algumas funções de avaliação e de otimização são mais adequadas a modelos específicos enquanto que outras não são em nada adequadas. No entanto, cabe ao engenheiro responsável pela implementação do algoritmo a escolha do melhor modelo, função de avaliação e otimização para o problema a ser resolvido.

Modelo	Avaliação	Otimização
K-Nearest Neighbor	Taxa de Erro/Acurácia	Algoritmos Gulosos
Máquinas de Vetor Suporte	Precisão e Revocação	Branch and bound
Naive Bayes	Erro Quadrático	Gradiente Descendente
Regressão Logística	Verossimilhança	Gradiente Conjugado
Árvores de Decisão	Probabilidade Posterior	Programação Linear
Random Forest	Ganho de Informação	Programação Quadrática
Redes Neurais	Custo/Utilidade	
Redes Bayesianas	Margem	
Regressão Polinomial		

Tabela 3 – Exemplos de componentes de algoritmos de aprendizado de máquina

Ademais, para que este algoritmo possa ser executado, é necessário uma etapa de pré-processamento de dados, filtrando observações que possam alterar ou impedir a execução normal do algoritmo, como dados ausentes ou a presença de *outliers*. Esta etapa é específica para cada projeto e cabe ao engenheiro responsável ser capaz de identificar possíveis problemas que possam surgir no tratamento destes dados.

3.1.3 Treinamento e Validação

Uma vez definido a primeira versão do algoritmo de aprendizado supervisionado a ser utilizado (que poderá mudar ao longo do processo), é necessário enfim realizar e validar o aprendizado. Normalmente, os dados são separados em dados de treinamento (utilizados no algoritmo de aprendizado) e dados de validação, um conjunto de dados o que servirá para avaliar a eficácia do modelo em generalizar o seu aprendizado para observações que não foram incluídas na fase de treinamento enquanto se ajusta parâmetros do próprio modelo. Esta separação dos dados é necessária para evitar *over-fitting* do modelo, ou seja, um modelo que não é capaz de generalizar o aprendizado para além dos dados de treinamento.

O método de avaliação do modelo nos dados de validação é similar à função de avaliação utilizada pelo algoritmo. Este método também será definido pelo engenheiro de acordo com as especificidades do problema em questão. Além disso, é comum a definição de alguns objetivos a serem cumpridos pelo modelo, como um valor mínimo de erro considerado como aceitável. A validação portanto tenta resumir a eficácia do modelo em um único indicador de desempenho.

Com a separação dos dados e um método de validação definidos, o processo de treinamento e validação é feito seguindo, de forma geral, o fluxograma da imagem 5. Este processo contudo, pode ser feito de forma iterativa variando a maneira pela qual a base de dados é dividida entre dados de treino e de validação, através da técnica de validação cruzada.

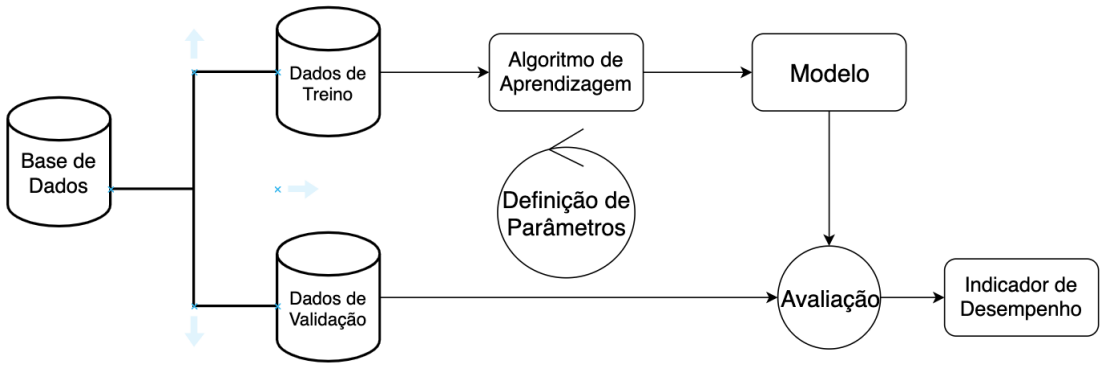

Figura 5 – Fluxograma de treinamento e validação de um modelo de aprendizado de máquina.

3.1.3.1 Validação Cruzada

Validação cruzada é uma técnica de validação de modelo onde o processo de divisão da base de dados em dados de treino e de validação é feito múltiplas vezes, a fim de avaliar a eficácia do modelo de generalizar o seu aprendizado para observações independentes com parâmetros específicos e de detectar possíveis ocorrências de *over-fitting* [18].

Um dos métodos de validação cruzada é o *k-fold*, no qual a base de dados é dividida em k diferentes subconjuntos. Um destes subconjuntos será utilizado como dados de validação, enquanto os outros $k - 1$ subconjuntos servirão como dados para treinamento do modelo. Ao final deste processo, haverá portanto k diferentes números indicando o desempenho do modelo, e qualquer tomada de decisão sobre o processo de treinamento se baseará nestes valores e no quanto eles variaram entre si [18], podendo tanto ajustar parâmetros específicos do modelo como também alterá-lo ou até refletir sobre como alterar os dados de treinamento, através de um processo chamado de *feature engineering*.

3.1.3.2 Feature Engineering

O resultado da primeira versão de um modelo de aprendizado de máquina pode não satisfazer as exigências pré-concebidas na idealização do projeto. Isto acontece porque o algoritmo escolhido não foi adequado para o problema em questão, ou porque os dados de treinamento não estão em sua melhor forma. Neste último caso, há a necessidade de modificá-los afim de fornecer uma base de dados de treinamento com informações mais adequadas para o aprendizado, um processo conhecido como *feature engineering*. Tipicamente é nesta fase que a maior parte do esforço de projetos de aprendizado de máquina é gasto [16].

Feature engineering não possui uma heurística ou um algoritmo claro comum para todos os projetos. Esta etapa é específica ao domínio de cada um dos problemas a serem resolvidos. Cabe ao engenheiro entender o contexto no qual os dados foram coletados e

como eles podem fornecer informações de valor para o algoritmo de aprendizado, podendo assim gerar hipóteses de variáveis que poderão ser relevantes, criando modelos com elas, e enfim avaliando o seu desempenho.

Assim, o escopo de um projeto de aprendizado de máquina geralmente segue, de forma sucinta, o seguinte fluxograma [19]:

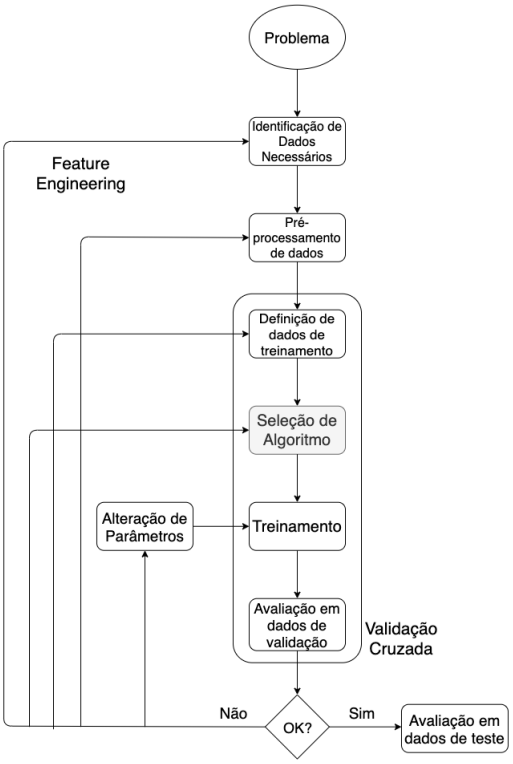

Figura 6 – Fluxograma de processo de aprendizado de máquina.

3.1.4 Teste

É importante ressaltar que, quando em posse de um conjunto de dados relativamente grande, é recomendável a criação de um terceiro conjunto de dados, chamado de dados de teste, completamente independente dos dados de treinamento e de validação, sendo utilizado após o processo iterativo de definição de parâmetros do modelo. Assim, enquanto que os dados de validação podem ser redefindos a cada iteração da definição de parâmetros do modelo, os dados de teste servem apenas para avaliar a eficácia do modelo final, a fim de promover uma maneira de comparar modelos diferentes em um mesmo conjunto de dados que não foi incluido no treinamento.

Alguns projetos de aprendizado de máquina podem ignorar por completo a distinção entre dados de validação e dados de teste, mas neste trabalho essa distinção será necessária. Ademais, é muito comum a confusão de nomenclatura entre dados de validação e dados de teste, mas neste trabalho seguiremos a definição dada por [20]:

- Dados de Treinamento: dados usados para o aprendizado;
- Dados de Validação: dados usados para definição de parâmetros do modelo (i.e. número de camadas de uma rede neural), separados dos dados de treinamento;
- Dados de Teste: dados utilizados especificamente para avaliar a performance de um modelo já definido e treinado.

Projetos de aprendizado de máquina são portanto, processos iterativos, onde cada uma das seguintes etapas é possível de ser revista ou refeita:

- Seleção de dados necessários e coleta: escolha de quais dados (e em que formato) deverão ser coletados.
- Pré-processamento de dados: remoção e/ou tratamento de observações nulas e *outliers* e tratamento de ruído. Cabe aqui também a alteração no formato dos dados a serem treinados ou criação de novas variáveis com base nos dados coletados.
- Divisão dos dados em conjuntos de treinamento e de validação: esta etapa pode ser revisitada caso se opte pela execução de validação cruzada.
- Seleção do Algoritmo: escolha do algoritmo e modelo de aprendizado a ser utilizado.
- Treinamento do algoritmo com base nos dados de treinamento.
- Avaliação do modelo nos dados de validação: verificação da capacidade do modelo de generalizar o aprendizado em dados que não foram utilizados para treinamento.

  Caso a validação não seja satisfatória, é necessário entender o porquê, ou levantar hipóteses que possam justificar o não cumprimento das exigências de projeto. Variadas ações podem ser tomadas aqui dependendo do diagnóstico feito, e este processo é refeito a partir da etapa a ser modificada e enfim validado.
- Avaliação de performance do modelo em questão em dados de teste independente e comparação com modelos de parâmetros ou algoritmos diferentes para que se possa escolher o melhor.

3.2 Rede Neural

O estudo de redes neurais artificiais foi inspirado na observação de que o método de aprendizado de sistemas biológicos é baseado em complexas redes de neurônios interconectados [21]. De forma geral, redes neurais artificiais são compostas por unidades (neurônios) que recebem um conjunto de entradas numéricas, possivelmente saídas de outros neurônios, e produzem uma única saída numérica, que possivelmente será uma das

entradas de outro neurônio. Assim, redes neurais artificiais tentam replicar a poderosa arquitetura de processamento de sistemas biológicos nervosos, através de várias unidades de processamento interligados entre si.

A arquitetura mais comum de redes neurais é a de Perceptron Multicamadas, utilizando um algoritmo de aprendizado chamado *backpropagation*. De forma sucinta, cada saída numérica de cada neurônio é resultado de alguma função de transferência de uma combinação linear dos valores de entrada, e cada valor de saída é multiplicado por um peso antes de servir como entrada de outro neurônio, este último sendo ajustado ao longo do processo de aprendizado do algoritmo. Os neurônios são divididos em camadas, que poderão ter funções geralmente não-lineares internas diferentes de acordo com o problema em questão e o aprendizado é feito através da alteração dos pesos das ligações dos neurônios entre camadas [21]. Esta arquitetura foi utilizada neste trabalho e seu funcionamento será aprofundado ao longo deste capítulo.

As redes neurais se tornaram em sistemas computacionais altamente utilizados em projetos de aprendizado de máquina por sua ampla flexibilidade e customização, sendo possível alterar sua estrutura e funções internas quando necessário, servindo tanto para aprendizado supervisionado quanto para não-supervisionado e para problemas de classificação e regressão. No entanto, em muitos domínios o uso de rede neural não é ótimo, por não oferecer nenhuma informação clara sobre como as variáveis contribuem para a análise (quando este é um dos objetivos) e também por ter um tempo de aprendizado mais lento do que outros algoritmos [22].

A seguir, nos apronfundaremos em cada um dos componentes das redes neurais a fim de entender como é feito o aprendizado utilizando estes sistemas.

3.2.1 Neurônio

Cada unidade de redes neurais artificiais, conhecidos como neurônios, são funções matemáticas modeladas a partir de neurônios biológicos que recebem excitações sinápticas de outros neurônios e geram uma outra excitação para outros neurônios. Cada neurônio j possui como entrada uma série de m valores x_i, cada um multiplicado por um peso w_{ji}. A soma destes valores ponderados, chamada aqui de v_j, serve como entrada para uma função de transferência ϕ, cujo resultado será a saída y deste neurônio. Em algumas arquiteturas, há também a presença de um *bias* b_j, que nada mais é do que um sinal de valor 1 com um peso b [23]. Podemos visualizar o modelo do neurônio artifical na figura 7, embora também seja possível representar o processo matemático de cada neurônio j através da seguinte equação matemática:

$$y_j = \phi(\sum_{i=1}^{m} x_i w_{ji} + b_j) \tag{3.1}$$

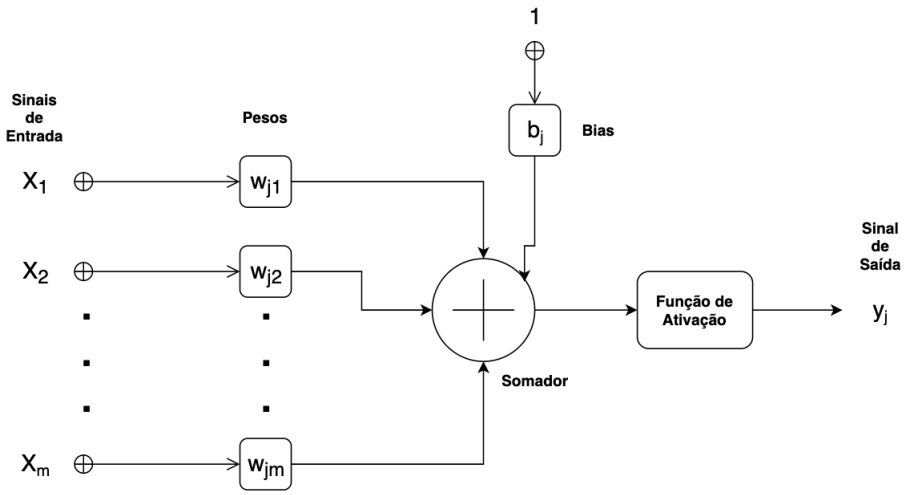

Figura 7 – Modelo de um neurônio artificial

3.2.2 Função de Ativação

A escolha da função de ativação é essencial para a performance da rede neural. Tipicamente, todos os neurônios de uma mesma camada possuem a mesma função de ativação, embora isso não seja necessário. Redes neurais com mais de uma camada tendem a utilizar funções não-lineares como funções de ativação, visto que se usassem funções lineares não haveria a necessidade de ter mais de uma camada em sua topologia. A maioria das funções de ativação possuem sua imagem limitada no intervalo de $[0, 1]$ ou $[-1, 1]$, embora existam funções de ativação com uma imagem mais ampla. A idéia por trás de funções com imagens limitadas deste tipo é análoga à taxa de potencial de ação de neurônios biológicos [24]. Alguns exemplos de função de ativação serão listadas a seguir:

- Função Identidade

$$\phi(x) = x \tag{3.2}$$

- Função Degrau

$$\phi(x) = \begin{cases} 0, & \text{para } x < 0 \\ 1, & \text{para } x \geq 0 \end{cases} \tag{3.3}$$

- Função Retificadora (ReLU)

$$\phi(x) = \begin{cases} 0, & \text{para } x < 0 \\ x, & \text{para } x \geq 0 \end{cases} \tag{3.4}$$

- Função Sigmóide

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

- Função Tangente Hiperbólica

$$\phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.6}$$

Funções de ativação binárias como a função degrau possuem uma grande desvantagem de não permitir valores contínuos em suas saídas, fazendo com que os sinais entre neurônios se limitem a serem ativados ou não. Funções lineares como a função identidade permitem a existência de variações na taxa de ativação. É importante ressaltar que modelos construídos com esta função de ativação podem ter problemas de convergência no processo de aprendizado, visto que pesos de entradas de neurônios de caminhos mais correlacionados com a saída tendem a aumentar sem limites, já que esta função não é normalizável.

Este mesmo problema é encontrado na função retificadora, dada a natureza de uma imagem sem limite superior. Além disso, neurônios de modelos criados com esta função de ativação podem acabar se tornando inativos independente das entradas, entrando em estados considerados como "mortos". Além disso, o fato desta função não ser diferenciável em zero pode complicar o processo de aprendizado, embora isto possa ser facilmente corrigido na implementação do algoritmo. Por outro lado, a natureza simples desta função traz benefícios computacionais no processo de aprendizado.

Funções como a sigmóide e a tangente hiperbólica não possuem alguns dos problemas mencionados anteriormente, como a ausência de limites na imagem da função, ou descontinuidades. A função sigmóide por exemplo, é considerada como uma representação realista do processo de ativação de neurônios biológicos, com sua saída permanecendo em zero até que alguma entrada seja recebida. A taxa de ativação aumenta rapidamente até atingir uma assíntota em 100% [25].

Já a função tangente hiperbólica, embora com comportamento similar à função sigmóide, possui como diferencial o fato dela ser centrada em zero, permitindo que sua saída assuma valores negativos e positivos, que podem trazer vantagens para a performance do processo de aprendizado [26].

3.2.3 Variações de Arquitetura

Diferentes arquiteturas de redes neurais artificiais existem, cada uma com suas vantagens em relação às outras. A arquitetura utilizada neste trabalho é a de perceptron multicamadas, Esta arquitetura é uma classe de redes neurais artificiais do tipo *feed-forward*, isto é, redes das quais as conexões entre os neurônios não formam um ciclo. Com exceção dos neurônios de entrada, cada neurônio da rede perceptron multicamadas utiliza uma função de ativação não-linear.

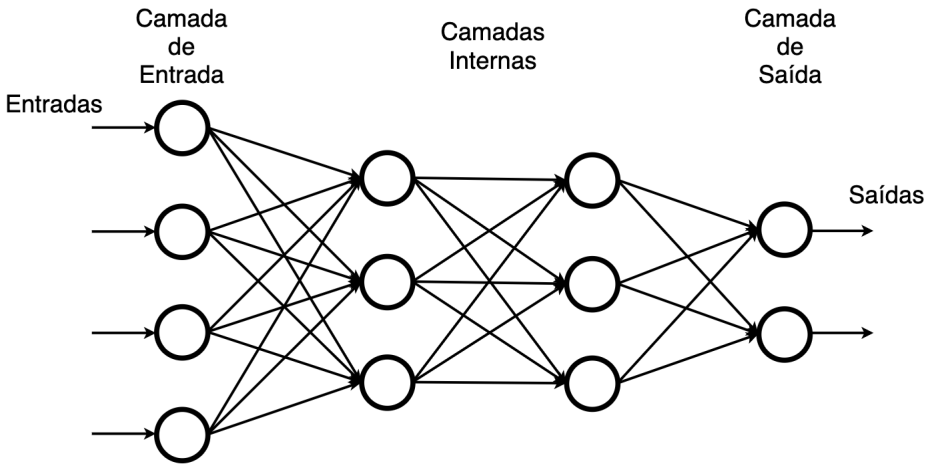

Figura 8 – Exemplo de arquitetura de rede perceptron multicamadas, com duas camadas internas além da camada de entrada e de saída.

Sua característica principal é a presença de pelo menos três camadas de neurônios distintas, uma de entrada, ao menos uma interna e uma última camada de saída. Sua arquitetura é uma generalização de redes perceptron de uma única camada, onde a relação entre as entradas e saídas é feita de forma direta como visto na equação 3.1. Embora um eficiente classificador binário, redes perceptron de uma única camada são somente capazes de classificar dados linearmente separáveis, tornando-a incapaz de aprender uma função XOR [23].

Um tipo popular de redes perceptron multicamadas é a rede neural convolucional, projetada especificamente para reconhecer formas bidimensionais com alto grau de invariância em relação a translação, escala e inclinação, sendo altamente utilizada em classificadores de imagens. [26].

Ao contrário de redes *feed-forward*, redes neurais recorrentes possuem realimentações entre as camadas internas, permitindo que a ativação de neurônios formem uma dinâmica temporal com memória de estados anteriores, tornando-as aplicáveis para reconhecimento inteligente de escrita e de fala, domínios temporais com complexas correlações em diferentes escalas de tempo [27].

3.3 Aprendizado da Rede Neural

Para cada diferente tipo de arquitetura existem diferentes algoritmos de aprendizado, considerando também o tipo de aprendizado a ser feito, supervisionado ou não-supervisionado. Na rede perceptron multicamadas, o aprendizado supervisionado é feito a partir de um conjunto de regras para o ajuste pesos da rede neural por minimização de erro através do algoritmo de *backpropagation*. Este processo de aprendizado é realizado calculando o erro entre a resposta estimada pela rede para cada registro de dados e seu respectivo valor real. O ajuste dos pesos sinápticos é feito com o objetivo de minimizar este erro.

Nesta arquitetura, as entradas de cada um dos neurônios são vetores $x_j(n)$ gerados por um ou mais neurônios de camadas anteriores (cada um com diferentes vetores de entrada). O argumento n aponta o instante discreto de tempo no qual cada vetor x_j foi gerado, ou seja, a etapa n do processo iterativo de ajuste dos pesos sinápticos. O sinal de saída de cada neurônio j é denotado por $y_j(n)$, sendo resultado da função de ativação ϕ aplicada à soma dos valores de entrada $x_j(n)$ com seus respectivos pesos $w_j(n)$ (resultado considerado aqui como o vetor $v_j(n)$). Este valor é comparado com cada valor desejado $t_j(n)$ pelo supervisor do processo de aprendizado, gerando os erros $e_j(n)$. Assim, temos:

$$e_j(n) = t_j(n) - y_j(n) \tag{3.7}$$

Estes valores de erro $e_j(n)$ serão utilizados como mecanismo de controle para o ajuste dos pesos sinápticos da rede, com o objetivo de tornar a saída $y_j(n)$ mais próxima do valor desejado $t_j(n)$, ou em outras palavras, com o objetivo de minimizar o erro $e_j(n)$ em termos de uma função de custo C. Como a saída $y_j(n)$ é definida em termos dos vetores de entrada $x_j(n)$ e de seus respectivos pesos sinápticos $w_j(n)$ (estes sendo as únicas variáveis que o aprendizado pode alterar), a minimização da função de custo C é feita em relação aos pesos sinápticos das entradas. Uma ilustração destas variáveis é visível na imagem 9. Podemos definir a função C em termos dos erros $e_j(n)$ (onde o conjunto Q inclui todos os neurônios da última camada) como:

$$C(n) = \frac{1}{2}\sum_{j \in Q} e_j^2(n) \tag{3.8}$$

3.3.1 Gradiente Descendente

O problema de minimizar a função de custo C pode ser simples, caso tenhamos apenas uma única observação como base de dados de treinamento e se a arquitetura da rede for relativamente simples, contendo apenas dois neurônios na camada de entrada e um único neurônio na camada de saída. Neste caso, a minimização da função de custo C depende apenas de dois valores. A magnitude do problema aumenta de acordo com a complexidade da topologia da rede e também com a quantidade de observações.

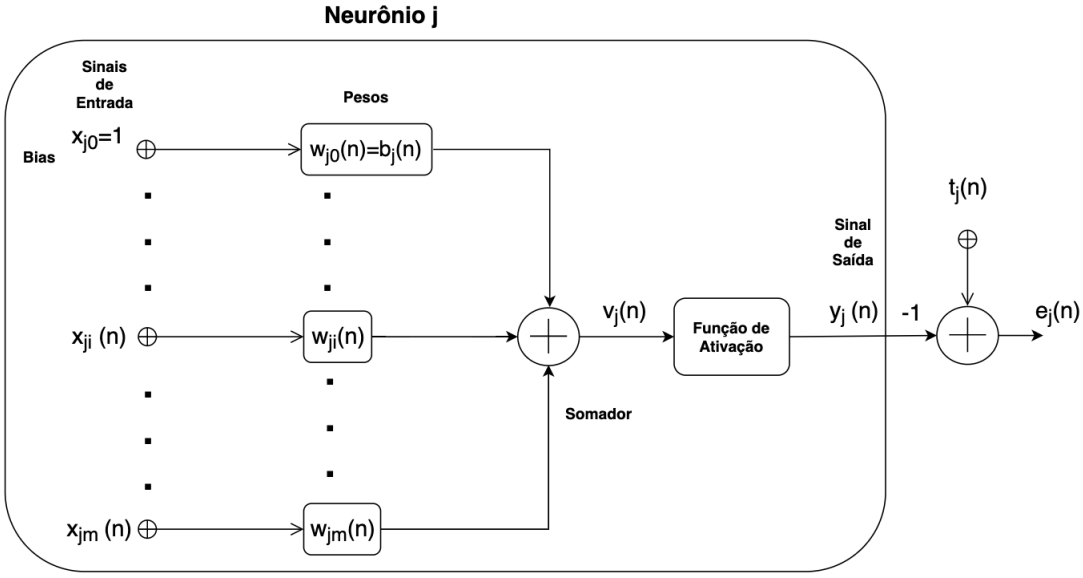

Figura 9 – Generalização do modelo do neurônio artificial da figura 7 considerando as variáveis utilizadas em cada etapa n do aprendizado.

Para poder realizar esta minimização de forma eficiente independente da quantidade de pesos sinápticos a ser considerada, o processo de aprendizado incorpora o algoritmo do gradiente descendente para encontrar o mínimo da função C na superfície gerada pelos pesos sinápticos w_j. Este algoritmo é iterativo e fácil de ser entendido. Primeiramente, é calculado o gradiente da função de custo ∇C em relação aos pesos sinápticos w_j:

$$\nabla C(w_j) = \left(\frac{\partial C}{\partial w_{j1}}, \frac{\partial C}{\partial w_{j2}}, ..., \frac{\partial C}{\partial w_{jm}} \right) \tag{3.9}$$

Como o cálculo do gradiente da função de custo C aponta o sentido e direção na superfície definida pelos pesos sinápticos w_j na qual C terá o maior incremento possível, a atualização dos pesos sinápticos será feita dando um pequeno passo em direção oposta ao gradiente. Assim, a variação dos pesos sinápticos Δw_j é definida por:

$$\Delta w_j = -\eta \nabla C(w_j) \tag{3.10}$$

Aqui, η é o tamanho do passo a ser dado na direção oposta ao gradiente em cada iteração, conhecido também como taxa de aprendizado. A correção singular do peso sináptico w_{ji} na etapa n será definida então como:

$$\Delta w_{ji} = -\eta \frac{\partial C(n)}{\partial w_{ji}(n)} \tag{3.11}$$

3.3.2 Backpropagation

Este algoritmo de gradiente descendente da função de custo C em função dos pesos sinápticos deverá ser aplicado de forma simultânea a todos os neurônios, caso contrário o

processo de aprendizado demoraria muito tempo. Assim, há a necessidade de expressar esta correção dos pesos sinápticos definida na equação 3.11 em função dos sinais de entrada, de saída e dos respectivos erros. Esta forma de correção simultânea dos pesos sinápticos define o algoritmo de *backpropagation*. Para simplificar o entendimento, vamos definir algumas notações seguindo a lógica da imagem 9.

Para cada neurônio j em cada iteração n, temos $v_j(n)$ definido como:

$$v_j(n) = \sum_{i=0}^{m} x_{ji} w_{ji} \tag{3.12}$$

Onde o valor de entrada x_{ji} é igual ao valor de saída do neurônio i da camada anterior, ou y_i. Esta equação também altera a notação da equação 3.1 considerando o bias b_j como o peso w_{j0} do primeiro sinal de entrada x_{j0}, este sendo igual à 1. Este valor $v_j(n)$ servirá como sinal de entrada da função de ativação ϕ_j do neurônio j, resultando na saída $y_j(n)$.

$$y_j(n) = \phi_j(v_j(n)) \tag{3.13}$$

Este valor de saída $y_j(n)$ será comparado ao valor desejado de treinamento $t_j(n)$ resultando no erro $e_j(n)$, como denotado na equação 3.7. O gradiente da função de custo C em relação ao peso sináptico w_{ji} é também conhecido como o fator de sensibilidade, determinando a direção de busca na superfície de pesos sinápticos para o peso w_{ji}. Podemos defini-lo em termos dos sinais de entrada, saída e erro de cada neurônio através da regra da cadeia:

$$\frac{\partial C(n)}{\partial w_{ji}(n)} = \frac{\partial C(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.14}$$

Diferenciando a equação 3.8 em relação à $e_j(n)$ teremos:

$$\frac{\partial C(n)}{\partial e_j(n)} = e_j(n) \tag{3.15}$$

Diferenciando a equação 3.7 em relação à $y_j(n)$ teremos:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.16}$$

Em seguida, podemos diferenciar a equação 3.13 em relação à $v_j(n)$ e assim obteremos:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \phi_j'(v_j(n)) \tag{3.17}$$

Por fim, diferenciando a equação 3.12 em relação ao peso sináptico $w_{ji}(n)$, obteremos:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = x_j(n) \tag{3.18}$$

Substituindo as últimas quatro equações na equação 3.14, teremos:

$$\frac{\partial C(n)}{\partial w_{ji}(n)} = -e_j(n)\phi'_j(v_j(n))x_j(n) \tag{3.19}$$

As equações 3.11 e 3.19 resultam em:

$$\Delta w_{ji} = \eta\delta_j(n)x_j(n) \tag{3.20}$$

Onde $\delta_j(n)$ é chamado de gradiente local, que pode ser entendido como a variação da função de custo em relação à ponderação das entradas, sendo definido como:

$$\delta_j(n) = -\frac{\partial C(n)}{\partial v_j(n)} = e_j(n)\phi'_j(v_j(n)) \tag{3.21}$$

Visto que a correção dos pesos sinápticos passa pela derivada da função de ativação ϕ, é recomendável que ela seja diferenciável em todos os pontos. Por isso, a escolha de qual função de ativação é ideal como explicada na subseção 3.2.2 é importante.

3.3.2.1 Backpropagation em Camadas Internas

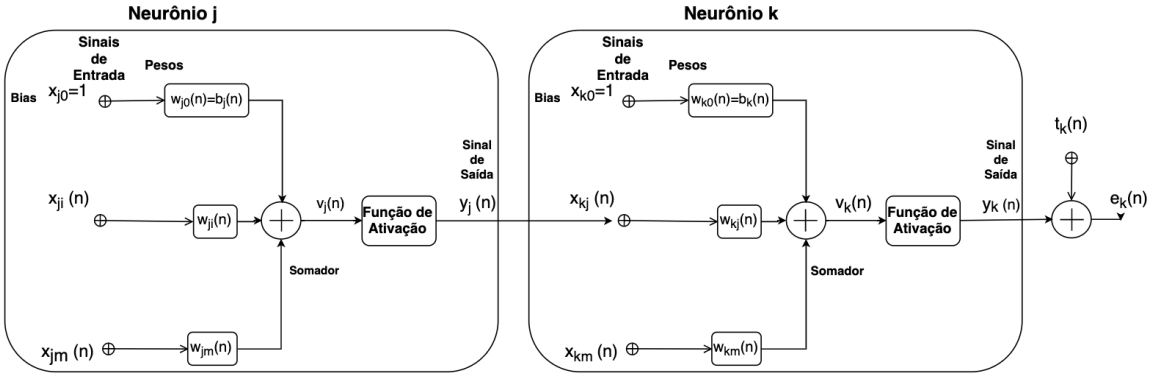

Figura 10 – Ilustração da ligação de neurônios em camadas diferentes

Esta definição do gradiente local δ_j em termos do erro $e_j(n)$ serve apenas para quando o neurônio j está na última camada, onde há uma valor de treinamento desejado $t_j(n)$ que possa ser comparado com a saída $y_j(n)$. Contudo, em arquiteturas de mais de uma camada, não há valores de treinamento desejados para as saídas dos neurônios das camadas internas. Assim, o erro $e_j(n)$ para um neurônio j de uma camada interna será definido de forma recursiva pelos sinais de erro dos neurônios da última camada da qual o neurônio j está conectado. É aqui que o algoritmo de *backpropagation* torna-se mais complexo. Podemos redefinir a equação 3.21 para o gradiente local $\delta_j(n)$ de um neurônio j de uma camada interna como:

$$\delta_j(n) = -\frac{\partial C(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial C(n)}{\partial y_j(n)}\phi'_j(v_j(n)) \tag{3.22}$$

Para calcular a derivada parcial $\partial C(n)/\partial y_j(n)$, voltaremos à equação 3.8 com uma notação um pouco diferente, onde um neurônio k é um neurônio da última camada, neste caso a camada posterior à camada do neurônio j, visível na figura 10:

$$C(n) = \frac{1}{2}\sum_{k\in Q} e_k^2(n) \tag{3.23}$$

Diferenciando os dois lados da equação acima em relação à $\partial y_j(n)$, temos:

$$\frac{\partial C(n)}{\partial y_j(n)} = \sum_{k\in Q} e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{3.24}$$

Pela regra da cadeia, segue que:

$$\frac{\partial C(n)}{\partial y_j(n)} = \sum_{k\in Q} e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.25}$$

Porém, sabemos que:

$$e_k(n) = t_k(n) - \phi_k(v_k(n)) \tag{3.26}$$

,

Logo, temos:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\phi_k'(v_k(n)) \tag{3.27}$$

Além disso, podemos definir $v_k(n)$ como:

$$v_k(n) = \sum_{j=0}^{m} x_{kj} w_{kj} = \sum_{j=0}^{m} y_j w_{kj} \tag{3.28}$$

onde o valor de entrada x_{kj} do neurônio k é igual à saída y_j do neurônio j da camada anterior. Isso faz com que a derivada parcial de $v_k(n)$ em relação à $y_j(n)$ seja igual ao peso sináptico da conexão do neurônio j com o neurônio k, ou w_{kj}. Enfim podemos reescrever a equação 3.25 como:

$$\frac{\partial C(n)}{\partial y_j(n)} = -\sum_{k\in Q} e_k(n)\phi_k'(v_k(n))w_{kj}(n) = -\sum_{k\in Q} \delta_k(n) w_{kj}(n) \tag{3.29}$$

Por fim, juntando as equações 3.22 e 3.29, podemos definir o gradiente interno $\delta_j(n)$ do neurônio j de uma camada interna como:

$$\delta_j(n) = \phi_j'(v_j(n)) \sum_{k\in Q} \delta_k(n) w_{kj}(n) \tag{3.30}$$

3.3.3 Algoritmo

O algoritmo de *backpropagation* (com tradução literal de retropropagação) é chamado assim porque a cada iteração, a correção dos pesos sinápticos é feito de traz para frente, ou seja, começando da última camada e retropropagando as correções até chegar na primeira camada. Com os pesos sinápticos corrigidos, o valor de saída da rede é recalculado e comparado com o valor de treinamento e o processo de correção dos pesos através da retropropagação é feito novamente.

Podemos então resumir o algoritmo de treinamento de uma rede neural artificial em seis etapas:

1. Inicialização aleatória dos pesos em números próximos à zero;
2. Aplicar a primeira observação dos dados de treinamento na camada de entrada;
3. *Forward Propagation*: propagar os valores de saída dos neurônios ao longo da rede até a última camada, calculando então o valor de saída $y(n)$;
4. Comparar a saída $y(n)$ com o valor de treinamento $t(n)$ e mensurar o erro $e(n)$;
5. *Backpropagation*: da última camada até a primeira o erro é retropropagado, calculando o gradiente local de cada neurônio a partir do erro da última camada e ajustando seus pesos sinápticos de forma proporcional à quanto eles contribuem para a função de custo C;
6. Repetição das etapas 2-5 depois de *cada* observação (aprendizado do tipo *online*) ou a cada subconjunto de dados (aprendizado por *batch*);

Por fim, este processo se repete até que haja uma convergência do processo de minimização da função de custo C, até que o número de iterações atinja um limite máximo ou até que o erro atinga um valor mínimo que seja aceitável, um limite pré-definido pelo projeto em si. Assim, a escolha da taxa de aprendizado η torna-se crucial para a eficácia do processo, visto que, se este valor for muito grande, o algoritmo de minimização pode não convergir, mas se for muito pequeno, há a possibilidade de uma demora no aprendizado.

3.4 Filtro de Kalman

Este projeto aborda a minimização do erro de métricas de performance derivadas de mensurações de latitudes e longitudes capturados por dispositivos móveis com GPS (*Global Positioning System*) integrado. Estas medidas porém, apresentam algumas incertezas e ruídos, muitas vezes ocasionadas por fatores externos. Portanto, há a necessidade de se fazer uma ponderação dos dados existentes, para que se obtenha estimativas confiáveis de dados

de localização. O filtro de Kalman é um algoritmo que oferece uma maneira para que estes ruídos e erros de mensuração sejam minimizados, sendo utilizado não apenas em projetos que envolvem estimação de trajetórias, mas também em predição de séries temporais e em processamento de sinais. Particularmente, o filtro de Kalman extendido, uma versão do filtro de Kalman extendido para sistemas não-lineares é comumente usado em sistemas de mensurações de GPS [28]. Contudo, neste trabalho abordaremos o filtro de Kalman simples, que no desenvolvimento deste trabalho provou-se eficaz para a minimização de ruídos de captura dos dados de localização dos jogadores de futebol.

Em posse das mensurações ruidosas captadas ao longo do tempo, o filtro de Kalman estima valores que tendem a ser mais precisos do que as próprias mensurações ruidosas. O filtro de Kalman se resume então a um processo recursivo capaz de reduzir a soma dos quadrados das diferenças entre os valores reais e os estimados pelo próprio algoritmo. Neste processo há a presença de duas etapas distintas, a da predição e da atualização. A predição, ou estimativa *a priori*, é uma estimativa do estado atual com base nos dados estimados anteriormente, sem incluir as mensurações no tempo atual. Em seguida, o estado atual é atualizado com a mensuração do tempo atual, gerando uma correção na estimativa *a priori*, a estimativa *a posteriori*. A derivação das equações abaixo pode ser encontrada no artigo original de Rudolf Kálmán [29].

3.4.1 Predição

O filtro de Kalman assume que o verdadeiro estado x_k em um instante de tempo k pode ser obtido através do estado anterior no instante de tempo $k-1$ da seguinte forma:

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{3.31}$$

Na equação acima, F_k representa o modelo de transição de estados aplicado ao estado anterior x_{k-1}, B_k é o modelo de controle aplicado ao sinal de controle u_k (quando presente) e w_k representa o ruído do processo, assumido como amostrado de uma distribuição normal de média zero e covariância Q_k ($w_k \sim \mathcal{N}(0, Q_k)$). Como neste trabalho o filtro de Kalman será usado apenas para estimar os dados reais de localização, reduzindo o erro da medição do *GPS*, não havendo portanto qualquer tipo de mecanismo de controle, iremos desconsiderar o modelo de controle B_k para a revisão do filtro de Kalman.

O filtro de Kalman contudo, consegue modelar o estado real x_k a partir da medição ruidosa z_k como:

$$z_k = H_k x_k + v_k \tag{3.32}$$

Aqui, H_k representa o modelo de medição que realiza uma transformação do espaço de estados verdadeiro para o espaço de estados medidos e v_k representa o ruído da medição,

assumido como amostrado de uma distribuição normal de média zero e covariância R_k ($v_k \sim \mathcal{N}(0, R_k)$).

Para podermos analisar o funcionamento do filtro de Kalman, vamos denotar $\hat{x}_{n|m}$ como a estimativa de x no tempo n, dada as observações até o tempo m. Assim, na fase de predição, a estimativa *a priori* pode ser calculada a partir da própria estimativa do passo anterior $\hat{x}_{k-1|k-1}$ ponderada pelo modelo de transição de estados F_k como:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} \tag{3.33}$$

Para que, na etapa de atualização, o algoritmo consiga realizar uma correção na estimativa *a priori*, minimizando o erro entre os valores reais e os estimados, há a necessidade de calcular a matriz de covariância do erro *a priori* $P_{k|k-1}$, ou seja, uma predição da acurácia da estimação do estado, sendo baseada na própria a matriz de covariância do erro atual $P_{k-1|k-1}$ e no ruído Q_k de processo inerente ao sistema, como:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{3.34}$$

A etapa de predição de no instante de tempo k do filtro de Kalman se resume a estas duas equações anteriores, realizando a predição tanto do estado real atual $\hat{x}_{k|k-1}$ quanto da matriz de covariância do erro atual $P_{k|k-1}$.

3.4.2 Atualização

A etapa de atualização do instante de tempo k do filtro de Kalman é feita comparando a estimativa $\hat{x}_{k|k-1}$ com a medição ruidosa z_k, e atualizando-a a partir do produto entre essa diferença com o ganho de Kalman K_k. Este ganho de Kalman K_k é calculado a partir da minimização da matriz de covariância do erro $P_{k|k-1}$, que também deverá ser atualizado para a próxima iteração $k+1$.

Assim, primeiramente é calculado o resíduo $\tilde{y}_k$ entre a medição e a estimação como:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \tag{3.35}$$

Fazendo com que a estimativa atualizada de $\hat{x}_{k|k}$ seja igual a:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{3.36}$$

O ganho de Kalman K_k é calculado como:

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{3.37}$$

onde S_k pode ser interpretado como a covariância do resíduo $\tilde{y}_k$ caso o modelo esteja correto, sendo calculado como:

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{3.38}$$

Olhando as equações 3.37 e 3.38 é possível observar os dois possíveis casos que tornarão a estimativa *a posteriori* $\hat{x}_{k|k}$ mais próxima à medição z_k ou à estimativa *a priori* $\hat{x}_{k|k-1}$. No primeiro caso, se a covariância da medição tender a zero, o ganho de Kalman se aproximará de H_k^{-1} (a inversa da transformação do espaço de estados verdadeiro para o espaço de estados de medição). Ou, mais especificamente:

$$\lim_{R_k \to 0} K_k = H_k^{-1} \tag{3.39}$$

Neste caso, o resíduo $\tilde{y}_k$, ao ser multiplicado pelo ganho K_k fará com que a atualização da estimativa *a posteriori* $\hat{x}_{k|k}$ pondere mais ao valor da medição z_k (neste caso transformado para o espaço de estados verdadeiro através de H_k^{-1}), já que sua covariância seria próxima de zero, tornando-a mais confiável.

Porém, se a covariância do erro *a priori* $P_{k|k-1}$ tender a zero, o ganho de Kalman K_k também o faz, assim:

$$\lim_{P_k \to 0} K_k = 0 \tag{3.40}$$

Aqui, a atualização da estimativa *a posteriori* $\hat{x}_{k|k}$ quase que desconsideraria o valor da medição z_k, ponderando mais ao valor da estimativa *a priori* $\hat{x}_{k|k-1}$, já que a covariância do erro *a priori* é próxima de zero, tornando-a mais confiável.

Por fim, a covariância do erro *a posteriori* $P_{k|k}$ é calculada como:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{3.41}$$

4 Desenvolvimento da solução proposta

Com os objetivos traçados no capítulo 2 em mente, utilizando os métodos apresentados no 3 para atingi-los, utilizou-se a linguagem de programação Python [30], que permite a implementação de algoritmos de aprendizado de máquina com uma sintaxe simples e documentação vasta, juntamente com as bibliotecas Pandas [31], auxiliando no tratamento e organização dos dados, além da Scikit-learn [32], que auxilia na implementação de algoritmos de redes neurais e de validação de modelos de aprendizado de máquina, além de outras bibliotecas auxiliares. Este captítulo apresenta o processo de desenvolvimento realizado para a implementação da solução proposta.

4.1 Dados disponíveis

Foi disponibilizado pelo Figueirense Futebol Clube dez arquivos de formato JSON, cada um contendo registros de GPS de uma atividade de futebol específica de duração média de 1 hora, seja de treino ou de jogo, realizada por um jogador da equipe. Estes arquivos foram gerados por dispositivos Ksport. Cada registro de GPS, como visualizado em 1, possui valores diferentes de tempo, latitude, longitude, uma variável de acurácia do sinal de GPS (valor em metros do raio da circunferência ao redor do ponto de latitude e longitude do qual a verdadeira localização está com 95% de certeza) e uma variável de velocidade instantânea vinda do satélite calculada a partir do Efeito Doppler.

```
{
    "input-type": "gps",
    "datetime": 1510047861933,
    "latitude": -27.604333226,
    "longitude": -48.518829577,
    "accuracy" : 24.0,
    "speed": 0.22224,
}
```

Listing 1: Exemplo de registro de GPS em formato JSON

Como definido no Capítulo 2, foi realizada uma amostragem destes dados com dez períodos de amostragem distintos, sendo eles 0.25 s, 0.5 s, 0.75 s, 1.0 s, 1.25 s, 1.5 s, 1.75 s, 2.0 s, 2.5 s e 3.0 s. A decisão da variação dos períodos de amostragem foi feita para não limitar o modelo a aprender as diferenças entre dados de 1 Hz e de 10 Hz, mas sim que generalize para dados de variadas frequências, sendo útil para nosso projeto considerando a possibilidade de haver falha na captura de dados por mais de 1 s. Assim, para cada

atividade registrada, foram criadas outros 10 arquivos com dados de GPS de diferentes frequências, resultando em 100 arquivos distintos, totalizando mais de 360 mil registros diferentes de GPS.

4.2 Tratamento dos Dados

4.2.1 Transformação de Coordenadas

Dados de latitude e longitude não são propícios para o cálculo de distância em metros. Assim, é necessário realizar uma conversão destes valores do sistema de coordenadas geodésico para cartesiano, onde cada unidade é representada por um metro. Este cálculo é simples de ser feito. Consideraremos o primeiro registro de GPS como referência (λ_0, φ_0), onde λ_0 é a sua longitude e φ_0 a sua latitude. Este ponto será a origem do diagrama cartesiano do novo sistema de coordenadas.

Lembrando que latitudes e longitudes são medidas em graus, uma latitude φ é fácil de ser convertida para o eixo y do novo sistema cartesiano, calculando o arco criado pela diferença entre ela e a latitude de referência φ_0 através da medida do arco criado por 1^{o}. A longitude λ é calculada comparando-a com a longitude de referência λ_0, multiplicando sua diferença pelo arco da circunferência que passa pela latitude de referência φ_0.

Por fim, considerando R como o raio esférico da Terra em metros, para qualquer coordenada de GPS (λ,ϕ), seu respectivo ponto (x, y) no sistema de coordenadas cartesiano será calculado baseando-se na projeção equiretangular [33] como:

$$x = \frac{2\pi R}{360} \cdot cos(\varphi_0 \frac{\pi}{180}) \cdot (\lambda - \lambda_0) \tag{4.1}$$

$$y = \frac{2\pi R}{360} \cdot (\varphi - \varphi_0) \tag{4.2}$$

4.2.2 Aplicação do Filtro de Kalman

Após a conversão do sistema de coordenadas geodésico para o sistema cartesiano, aplicamos o Filtro de Kalman nos dados em coordenadas cartesianas a fim de minimizar o efeito de ruídos das mensurações do GPS. É nesta etapa do tratamento de dados que a variável de acurácia do sinal do GPS torna-se importante. Utilizando a notação definida em 3.4, a matriz de covariância do ruído da medição R será composta pela variância dos valores de acurácia do sinal de GPS.

Por outro lado, não temos informações suficientes para definir a covariância do ruído do processo Q. Para fins de desenvolvimento, a cada iteração k a matriz de covariância Q_k foi definida por um modelo de ruído branco discreto com variância igual a 1 e com o parâmetro de intervalo de tempo definido pela diferença em segundos dos registros da

iteração atual k e a anterior $k-1$. Esta etapa foi feita com o auxílio da biblioteca FilterPy, especializada na implementação de filtros matemáticos.

4.2.3 Preparação dos Dados

Com o ruído de medição suavizado nos dados espaciais pelo Filtro de Kalman, podemos extrair as métricas de interesse para a construção do modelo de rede neural. A primeira métrica de interesse é a distância. Considerando os dados espaciais no sistema de coordenadas cartesianas, a distância d, em metros, entre dois pontos (x_A, y_A) e (x_B, y_B) é dada pela distância euclidiana entre eles:

$$d_{A,B} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{4.3}$$

Para cada registro i de atividade amostrada definida em 4.1, foi calculada a distância $\hat{d}_i$ entre os pontos i e $i-1$, além da distância d_i entre eles considerando os dados reais (calculada pela soma das distâncias entre os pontos que foram ignorados pela amostragem). Definimos enfim r_i como a razão entre d_i e $\hat{d}_i$. Podemos visualizar a diferença espacial entre $\hat{d}_i$ e d_i na figura 11.

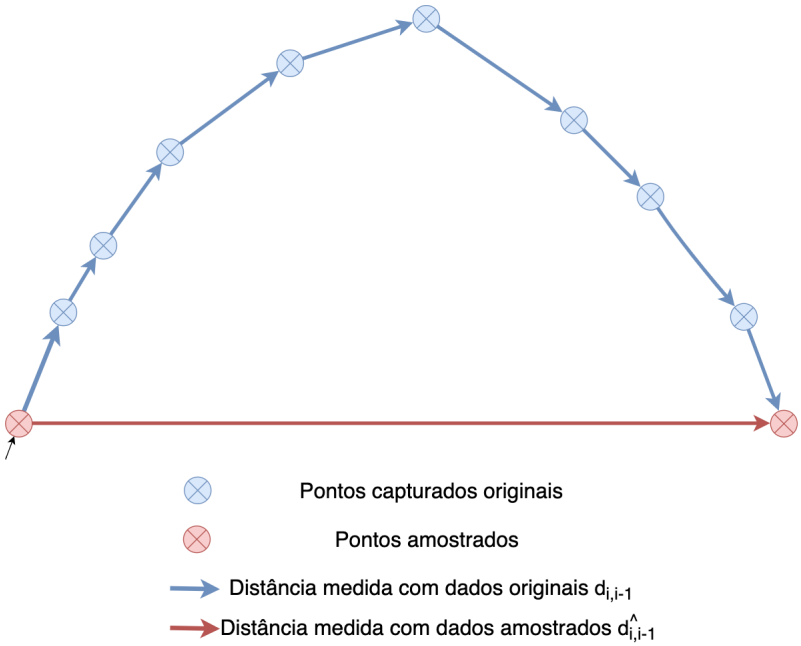

Figura 11 – Visualização da diferença entre as distâncias $\hat{d}_i$ e d_i de um par de pontos.

Além disso, para cada registro i foi calculado o ângulo θ_i da mudança de direção do jogador ocorrida naquele instante, considerando a hipótese de que, caso a mudança de direção não seja tão brusca, a movimentação torna-se mais linear, fazendo com que a perda de informação por amostragem não seja tão grande.

Para a construção de um dataframe para realizar o aprendizado, foi considerado também o valor da velocidade instantânea v_i provida pelo satélite de GPS, que embora

não seja igual ao cálculo da distância divido pelo intervalo de tempo entre dois pontos consecutivos, fornece uma informação relevante para a construção do modelo. Assim, para cada registro i temos:

- Razão r_i entre a distância calculada com dados reais e com dados amostrados;
- Intervalo de tempo dt_i entre o registro i e $i-1$;
- Velocidade instantânea v_i;
- Ângulo θ_i de mudança de direção;

Todos os dados foram normalizados de acordo com a normalização de mínimo-máximo. O objetivo do aprendizado foi então a estimação da razão r_i que servirá como fator de correção para a distância $\hat{d}_i$ calculada entre dois registros de GPS quaisquer, com base nas métricas de movimentação listadas acima. Ademais, foi levantada a hipótese de que a razão r_i seria melhor estimada se o modelo compreendesse de que parte de uma movimentação específica o registro em questão representa (i.e. início ou fim de uma arrancada). Assim, para a estimação da razão r_i, o modelo considerará também as métricas de movimentação dos instantes $i-2$, $i-1$, $i+1$ e $i+2$. Por fim, podemos simplificar o modelo f da rede neural como uma função da seguinte da maneira:

$$r_i = f(dt_{i-2:i+2}, \theta_{i-2:i+2}, v_{i-2:i+2}) \tag{4.4}$$

4.3 Conjuntos de Treinamento, Validação e Teste

Para evitar a ocorrência de *over-fitting*, o conjunto de dados de treinamento S_{train} definida em 4.1 foi dividido de acordo com sua respectiva atividade para que fosse feita uma validação cruzada de 10-*fold*.

Em cada iteração i das dez iterações do 10-*fold*, os dados amostrados de uma das atividades foram separados do conjunto total para que servissem como conjunto de validação S_{ival} do modelo treinado com o conjunto de treino S_{itrain} composto pelas outras nove atividades. Nesta etapa de validação cruzada foi feito o ajuste de parâmetros do modelo, como taxa de aprendizado e número de camadas e de neurônios. Cada registro possui quinze diferentes variáveis como definido em 4.2.3.

Visto que aqui estamos criando um modelo que aproxima as métricas extraídas de dados amostrados das métricas extraídas de dados reais, há a necessidade de validar a hipótese de que essa amostragem se assemelha a dados de GPS de outros dispositivos, como o Samsung Gear Fit utilizado pela empresa. Assim, há a necessidade de testar o modelo criado nos dados do *smartwatch* definidos em 2.4. Seguindo a nomenclatura definida pela

literatura em 3, o conjunto de testes S_{test} foi escolhido como os dados capturados com o Samsung Gear Fit de atividades que também foram capturadas pelo dispositivo Ksport, fornecendo então uma métrica de comparação. A 12 representa graficamente a divisão aplicada nos dados.

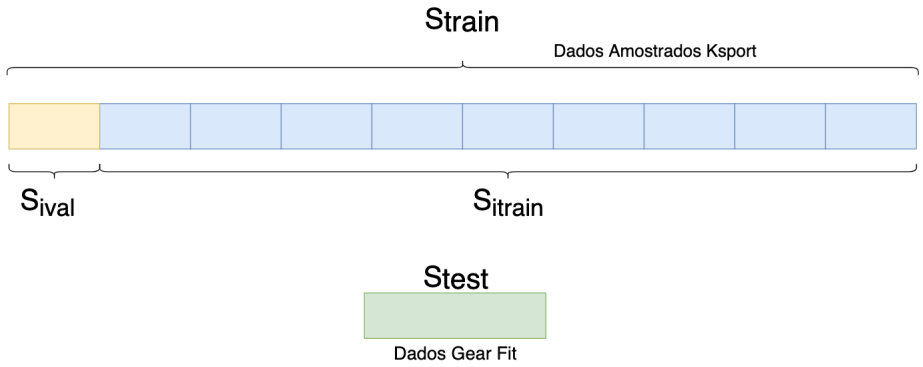

Figura 12 – Divisão dos dados do conjunto de exemplos S para o modelo de rede neural.

4.4 Construção da Rede e Validação Cruzada

Foi criado então um modelo de rede neural de regressão para estimar o fator de correção de uma dada distância medida com dados de certa frequência de captura. O modelo foi treinado com os exemplos rotulados definidos em 4.2.3. As entradas são quinze variáveis de métricas (tempo, ângulo e velocidade) de uma janela de cinco instantes de tempo enquanto que a saída é o fator de correção que a distância do instante central da janela deverá ser corrigida para se assemelhar à distância mensurada com dados não amostrados.

Utilizando a biblioteca do Scikit-learn, temos alguns parâmetros no código para configurar o algoritmo de regressão:

- Número de neurônios nas camadas internas;
- Taxa de aprendizado η do algoritmo de gradiente descendente;
- Função de ativação ϕ dos neurônios;
- Limite máximo de iterações até convergência;

A definição destes parâmetros foi feita ao longo da etapa de validação cruzada. Foi testado diferentes combinações de parâmetros e mensurado o erro médio de cada iteração da validação cruzada. A variação de arquitetura (número de neurônios nas camadas internas) foi feita seguindo regras de boas práticas utilizadas em projetos de redes neurais.

O conjunto de parâmetros que resultou no menor erro médio de distância total (9.65%) das atividades foi:

- 7 neurônios na segunda camada e 3 neurônios na terceira camada;
- $\eta = 0.001$;
- Função de ativação ϕ retificadora (ReLU);
- Limite de 300 iterações até convergência.

Por fim, com a rede construída, podemos visualizar a integração final do Filtro de Kalman com a rede neural na figura 13.

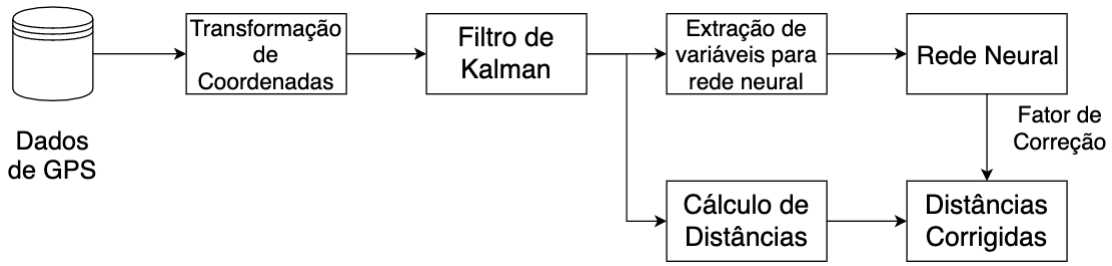

Figura 13 – Visualização da integração do Filtro de Kalman com a Rede Neural no cálculo das distâncias.

5 Resultados

5.1 Avaliação nos Dados de Teste

Como o objetivo definido em 2.4 era a minimização das diferenças entre as métricas extraídas dos dados da Ksport e os do Samsung Gear Fit, aplicamos o filtro de Kalman e o modelo de rede neural definidos em 4 nos dados de atividades capturadas com ambos os dispositivos. Os resultados podem ser visualizados nas tabelas abaixo.

Atividade	Distância Total K-sport (m)	Distância Total Gear Fit + Modelo (m)	Diferença Relativa
1	1171	1475	25.96%
2	1038	1145	10.31%
3	3438	3412	0.76%
4	3007	2426	19.32%
5	2004	2151	7.34%
6	2680	2848	6.27%
7	1076	1182	9.85%
8	982	1263	28.62%
9	1533	1429	6.78%
10	1259	1513	20.17%
11	999	1128	12.91%
		Média	13.48%

Tabela 4 – Diferenças entre as distâncias totais mensuradas pelo K-sport (10 Hz) e o Samsung Gear Fit (1 Hz) com filtro de Kalman e modelo de rede neural

Podemos ver que os objetivos traçados em 2.4 foram cumpridos. A diferença média relativa da distância total das atividades capturadas por ambos os dispositivos caiu de 19.42% para 13.48%. Já nas distâncias em alta intensidade, a diferença média relativa caiu de 41.77% para 33.79%, um valor considerado aceitável considerando os experimentos feitos pela literatura [11].

Este resultado mostra que o objetivo inicial do projeto foi alcançado. Isto é, que utilizando um filtro de Kalman para minimização de ruídos e de uma rede neural para estimar as diferenças de distâncias entre dados capturados com frequências diferentes, é possível ter, utilizando um dispositivo de GPS com frequência de captura de 1 Hz, dados de performance com qualidade similar a dados capturados por um dispositivo de 10 Hz. Além disso, a rede neural foi aplicada aos dados do Samsung Gear Fit sem a aplicação do filtro de Kalman, a fim de avaliar a influência do ruído de medição nos dados de GPS. A diferença relativa da distância total neste caso ficou em torno de 15.70%, provando que a minimização do ruído foi crucial para o sucesso deste projeto.

Atividade	Distância em Alta Intensidade K-sport (m)	Distância em Alta Intensidade Gear Fit + Modelo (m)	Diferença Relativa
1	200	309	54.50%
2	60	151	151.67%
3	312	342	9.62%
4	471	324	31.21%
5	235	220	6.38%
6	393	319	18.83%
7	266	259	2.63%
8	276	374	35.51%
9	338	250	26.04%
10	223	279	25.11%
11	284	255	10.21%
		Média	33.79%

Tabela 5 – Diferenças entre as distâncias em alta intensidade mensuradas pelo K-sport (10 Hz) e o Samsung Gear Fit (1 Hz) com filtro de Kalman e modelo de rede neural

Contudo, a premissa fundamental deste projeto é que as métricas extraídas dos dados da K-sport representam fielmente a realidade. Antes de produtizar o modelo desenvolvido aqui, foi necessário realizar um experimento controlado onde as distâncias eram conhecidas para avaliar de forma mais objetiva a eficácia do modelo construído aqui.

5.2 Avaliação com Experimento Controlado

Baseando-se no experimento feito por [9], foi realizado um experimento controlado no CEFID-UDESC, em conjunto com o professor Lorival Caminatti e 7 alunos que jogam futebol nos times da universidade. Foi criado um circuito a ser percorrido pelos alunos, que pode ser visualizado na figura 14 com variação de movimentos, contendo: caminhada, trote, velocidade, sprint e agilidade.

Cada aluno executou duas voltas contínuas no circuito utilizando três dispositivos JogaPro (Samsung Gear Fit com dados sendo aplicados ao modelo desenvolvido no projeto) e três dispositivos GPSports (de frequência de captura de 15 Hz, maior do que os 10 Hz da Ksport), os quais foram ligados e desligados no mesmo local e momento. O tamanho total de cada sequência foi de 260 metros (2 circuitos de 130 metros). O tamanho total do experimento foi de 6430 metros (1 circuito caminhando para reconhecimento + 7 alunos executando 3 sequências cada + deslocamento de 40 metros após cada sequência). Os trechos do circuito do qual foi exigido que o atleta corresse em alta velocidade foi contabilizado como a distância em alta intensidade a ser mensurada por nossos algoritmos, totalizando em 2730 m.

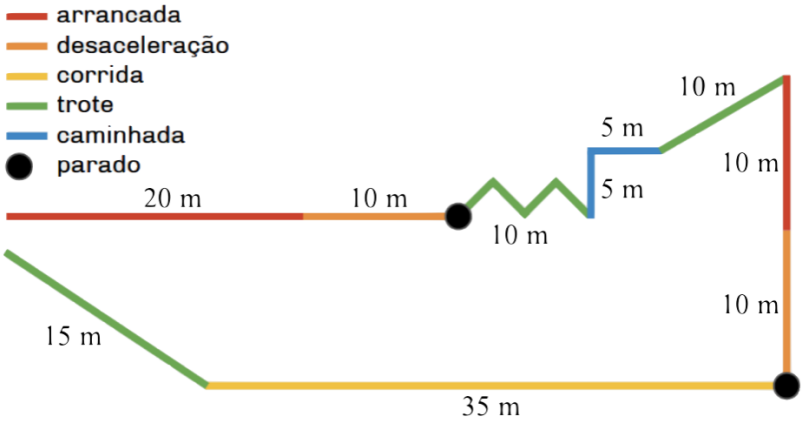

Figura 14 – Ilustração do circuito do experimento controlado.

Importante ressaltar estas medidas de 6430 m para distância total do experimento e de 2730 m para distância em alta intensidade são medidas aproximadas, devido a dois motivos: a existência de uma margem de "desobediência"natural dos atletas durante os movimentos e a incerteza quanto ao momento em cada sprint que o atleta atinge a velocidade de 14 km/h. Os resultados podem ser visualizados na tabela 6.

	Distância real	Joga Pro (GPS 1Hz)	GPSport (GPS 15Hz)
Distância total (com desvio padrão)	6430	6321±84	6193±445
Distância em Alta Intensidade (com desvio padrão)	2730	2492±140	2264±199
Distância média por sequência (com desvio padrão)	260	247±7	243±18

Tabela 6 – Resumo do experimento controlado

Este resultado superou as expectativas do time de desenvolvimento, já que o erro das distâncias mensuradas pelo produto da empresa foi menor do que o erro de um dispositivo de GPS de 15 Hz, uma frequência de captura muito superior a 1 Hz. Considerando o projeto como sucesso, o filtro de Kalman e o modelo de rede neural desenvolvidos aqui foram produtizados e já estão sendo utilizados tanto por clubes profissionais quanto por atletas amadores que utilizam o app *mobile* durante suas partidas.

6 Conclusões e Perspectivas

Este documento apresentou uma aplicação de técnicas de aprendizado de máquina com o objetivo de oferecer métricas de performance de atletas de futebol com dados capturados por dispositivos de baixo custo de mesma qualidade que dados capturados por dispositivos de alto custo. A aplicação de tecnologia no futebol ainda está muito limitada a clubes profissionais com alto poder aquisitivo, algo que não é realidade para a maioria dos clubes brasileiros. O projeto desenvolvido aqui conseguiu aproximar clubes e atletas de análises de performance que exigiriam um alto custo para serem obtidas caso se optasse pela aquisição de dispositivos de GPS de alta frequência.

Redes neurais são poderosos algoritmos de classificação e de estimação de valores que nos últimos anos têm entrado cada vez mais em evidência no mundo tecnológico. Contudo, ficou claro no desevolvimento deste projeto que a criação de uma rede neural ocupa uma pequena parte do tempo exigido por um projeto de aprendizado de máquina. A coleta ou a criação de dados catalogados é crucial para qualquer projeto deste tipo. Neste projeto, devido à incapacidade de comparar fielmente, registro a registro, dados de dispositivos distintos, foi necessária a amostragem de dados de GPS de 10 Hz, a fim de simular dados capturados de uma mesma atividade com frequências menores. Portanto, o treinamento do modelo foi realizado com dados que não são do mesmo tipo que seriam utilizados em produção, algo que não é tão comum em projetos de aprendizado de máquina. Devido à este contraste entre dados de aprendizado e dados de produção, foi necessária a realização de um experimento controlado a fim de avaliar melhor a eficácia do modelo projetado aqui. O resultado superou as expectativas da empresa, visto que os algoritmos desenvolvidos aqui aplicados a dados de 1 Hz resultaram em métricas mais próximas à realidade do que métricas extraídas de dados de frequência de 15 Hz.

Assim, concluiu-se que, em face à incapacidade de se obter uma base de dados ideal, sempre é possível pensar em alternativas que possam superar os obstáculos do projeto. A escolha pela aplicação do filtro de Kalman provou ser crucial para o sucesso do projeto, visto que, se o ruído de medição não fosse minimizado, o resultado final não teria sido tão satisfatório. Isso mostra que a escolha de qual modelo de aprendizado de máquina utilizar, ou que a escolha de parâmetros de modelo não são suficientes se os dados dos quais o modelo será aplicado estiverem afetados por ruídos de medição.

Como possibilidade de trabalho futuro, para o problema de mensuração de métricas de performance com dispositivos de baixo custo, é interessante ressaltar que *smartphones* e *smartwatches* possuem sensores de acelerômetro com frequência na faixa de 100 Hz que trazem relevantes informações sobre a direção de uma movimentação do atleta. Há

uma variedade de estudos na literatura sobre *sensor fusion*, a combinação de dados de diferentes fontes que resultam em informações menos incertas do que se apenas uma fonte fosse considerada. Integrar dados de GPS com acelerômetro, além de magnetômetro e giroscópio é um desafio complexo, mas que pode reduzir ainda mais a incerteza de dados espaciais capturados com dispositivos de baixo custo, podendo até extrair métricas de perfomance de atividades realizadas sem qualquer sinal de GPS, como nas modernas arenas cobertas e quadras fechadas, criando a possibilidade das análises realizadas na empresa serem aplicadas em esportes diferentes como futebol de salão, handebol, vôlei e basquete.

Por fim, percebe-se que a engenharia tem muito a acrescentar em diversas áreas de trabalho que exigem algum nível de análise matemática para otimizar seus processos internos. Com o custo de dispositivos eletrônicos reduzindo a cada ano, há a perspectiva de que cada vez mais projetos de aprendizado de máquina serão aplicados em uma maior variedade de áreas.

Referências

1 KUNZ, M. 265 million playing football. *FIFA magazine*, v. 7, p. 11–5, 2007. Citado na página 19.

2 ROSS, C. et al. *Football Money League*. [S.l.], 2019. Citado na página 19.

3 ANDERSON, C.; SALLY, D. *The numbers game: Why everything you know about soccer is wrong*. [S.l.]: Penguin, 2013. Citado na página 19.

4 BRADLEY, P. S. et al. High-intensity running in english fa premier league soccer matches. *Journal of sports sciences*, Taylor & Francis, v. 27, n. 2, p. 159–168, 2009. Citado na página 20.

5 DAVIS, J. A. Anaerobic threshold: review of the concept and directions for future research. *Medicine and Science in sports and Exercise*, v. 17, n. 1, p. 6–21, 1985. Citado na página 20.

6 BANGSBO, J.; NØRREGAARD, L.; THORSOE, F. Activity profile of competition soccer. *Canadian journal of sport sciences= Journal canadien des sciences du sport*, v. 16, n. 2, p. 110–116, 1991. Citado na página 20.

7 RAMPININI, E. et al. Variation in top level soccer match performance. *International journal of sports medicine*, © Georg Thieme Verlag KG Stuttgart· New York, v. 28, n. 12, p. 1018–1024, 2007. Citado na página 20.

8 DUFFIELD, R. et al. Accuracy and reliability of gps devices for measurement of movement patterns in confined spaces for court-based sports. *Journal of Science and Medicine in Sport*, Elsevier, v. 13, n. 5, p. 523–525, 2010. Citado na página 20.

9 JENNINGS, D. et al. The validity and reliability of gps units for measuring distance in team sport specific running patterns. *International journal of sports physiology and performance*, v. 5, n. 3, p. 328–341, 2010. Citado 2 vezes nas páginas 20 e 54.

10 VARLEY, M. C.; FAIRWEATHER, I. H.; AUGHEY1 2, R. J. Validity and reliability of gps for measuring instantaneous velocity during acceleration, deceleration, and constant motion. *Journal of sports sciences*, Taylor & Francis, v. 30, n. 2, p. 121–127, 2012. Citado na página 20.

11 JOHNSTON, R. J. et al. Validity and interunit reliability of 10 hz and 15 hz gps units for assessing athlete movement demands. *The Journal of Strength & Conditioning Research*, LWW, v. 28, n. 6, p. 1649–1655, 2014. Citado 2 vezes nas páginas 23 e 53.

12 DRAGLAND, Å. *Big Data, for better or worse: 90% of world's data generated over last two years. ScienceDaily. May 22, 2013*. 2013. Citado na página 27.

13 STEWART, D.; SCHATSKY, D.; SALLOMI, P. *Machine learning: things are getting intense*. [S.l.], 2018. Citado na página 27.

14 CHUI, M. Artificial intelligence the next digital frontier? *McKinsey and Company Global Institute*, v. 47, 2017. Citado na página 27.

15 STUART, R.; PETER, N. *Artificial intelligence-a modern approach 3rd ed.* [S.l.]: Berkeley, 2016. Citado na página 27.

16 DOMINGOS, P. M. A few useful things to know about machine learning. *Commun. acm*, v. 55, n. 10, p. 78–87, 2012. Citado 2 vezes nas páginas 28 e 31.

17 SATHYA, R.; ABRAHAM, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, Citeseer, v. 2, n. 2, p. 34–38, 2013. Citado na página 28.

18 CAWLEY, G. C.; TALBOT, N. L. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, v. 11, n. Jul, p. 2079–2107, 2010. Citado na página 31.

19 KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, v. 160, p. 3–24, 2007. Citado na página 32.

20 RIPLEY, B. D.; HJORT, N. *Pattern recognition and neural networks*. [S.l.]: Cambridge university press, 1996. Citado na página 32.

21 TOM, M. Machine learning. *Hill, McGraw*. Citado 2 vezes nas páginas 33 e 34.

22 CRAVEN, M. W.; SHAVLIK, J. W. Using neural networks for data mining. *Future generation computer systems*, Elsevier, v. 13, n. 2-3, p. 211–229, 1997. Citado na página 34.

23 FAUSETT, L. V. et al. *Fundamentals of neural networks: architectures, algorithms, and applications*. [S.l.]: prentice-Hall Englewood Cliffs, 1994. v. 3. Citado 2 vezes nas páginas 34 e 37.

24 HODGKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, Wiley Online Library, v. 117, n. 4, p. 500–544, 1952. Citado na página 35.

25 DALTON, A. D. J. Artificial neural networks, an approach to increasing machine intelligence. *IEEE POTENTIALS*, v. 1, p. 33–36, 2000. Citado na página 36.

26 HAYKIN, S.; NETWORK, N. A comprehensive foundation. *Neural networks*, v. 2, n. 2004, p. 41, 2004. Citado 2 vezes nas páginas 36 e 37.

27 SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: *Fifteenth annual conference of the international speech communication association*. [S.l.: s.n.], 2014. Citado na página 37.

28 WICKERT, M.; SIDDAPPA, C. Exploring the extended kalman filter for gps positioning using simulated user and satellite track data. 2018. Citado na página 44.

29 KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960. Citado na página 44.

30 ROSSUM, G. van. Python language website. *http://www. python. org/*, 2007. Citado na página 47.

31 MCKINNEY, W. pandas: a python data analysis library. *see http://pandas. pydata. org*, 2015. Citado na página 47.

32 PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, v. 12, n. Oct, p. 2825–2830, 2011. Citado na página 47.

33 KENNEDY, M.; KOPP, S. et al. *Understanding map projections*. [S.l.]: ESRI, 2000. Citado na página 48.