A lightweight approach for predicting errors in chess matches

Giovanni Comarela[1], **Davi Hagap Emanuel da Silva**[2]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
29.075-910 – Vitória – ES – Brazil

[2]Departamento de Informática – Universidade Federal de Viçosa (UFV)
36.570-900 – Viçosa – MG – Brazil

gc@inf.ufes.br, davi.emanuel@ufv.br

Abstract. *Chess is becoming more popular and accessible by the day. For instance, online chess enables matches between players from different parts of the world, bringing new ways of learning the game and interacting with other Web users. With this growth in popularity, there is a possibility to empower amateur players with rich computer analysis and tools, which may assist them in their learning process. One of the ways to analyze chess matches is through the study of errors. In this context, we present a new approach for the task of error prediction in chess. Our motivation is that knowing when players are likely to make a mistake is knowing what types of situations lead to difficulties in making the right decision. To that end, we add an abstraction layer to the already studied error prediction task, providing graph-based features to the machine learning models. Our results show a increase in the accuracy of the tested models, improving the results obtained in recent studies.*

Resumo. *O xadrez vem se tornando cada vez mais popular e acessível. O xadrez online permite que jogadores se desafiem de diferentes partes do mundo, possibilitando novas formas de aprendizagem do jogo e interação entre usuários na Web. Com o crescimento de popularidade, existe a possibilidade de empoderar jogadores casuais com ricas análises computacionais, que podem auxiliar no processo de aprendizagem do jogo. Uma das formas de analisar partidas de xadrez é através do estudo de erros. Nesse contexto, uma nova abordagem para a tarefa de predição de erros no xadrez é apresentada. A motivação desse trabalho é que saber quando jogadores tem mais chances de errar, é saber que tipos de situações apresentam mais dificuldades no processo de tomada de decisão. Para esse fim, foi adicionada uma camada de abstração no já estudado problema de predição de erro, acrescentando* features *baseadas em grafos aos modelos de aprendizagem de máquina. Os resultados indicam um aumento na acurácia dos modelos testados, melhorando os resultados obtidos em estudos recentes.*

1. Introdução

A popularização do xadrez online vem trazendo cada vez mais admiradores e tornando o jogo mais acessível e popular. A plataforma gratuita `lichess.org`, por exemplo, acumula atualmente mais de um milhão e meio de partidas semanais nos diversos modos de jogo disponíveis. O jogo também vem se popularizando no ramo do entretenimento. Um

dos maiores jogadores da atualidade, Hikaru Nakamura, reúne cerca de 18 mil espectadores simultâneos em suas transmissões ao vivo na `twich.tv` e foi contratado pela equipe de *e-sport* TSM para gerar conteúdo especializado para a organização, gerando receita através de anúncios [Statt].

Com uma grande quantidade de partidas sendo jogadas e disponibilizadas nessas plataformas, o jogo se torna um campo fértil para estudar o comportamento humano e como usuários interagem uns com os outros na Web. O uso de ferramentas computacionais no estudo do xadrez não é novidade, mas vem se tornando cada vez mais acessível e comum, não somente para jogadores profissionais mas também dos jogadores casuais. Jogadores de nível avançado buscam o aperfeiçoamento máximo de suas capacidades, desvendando variantes e utilizando o computador para analisar posições, tirando vantagem da capacidade computacional para aprofundar seu conhecimento em posições que eles julgam importantes. As análises realizadas com auxílio das *engines* acabam se tornando parte importante das preparações desses jogadores para competições.

Por outro lado, jogadores amadores geralmente encontram nas análises do computador um auxílio na hora de rever suas partidas e estudar seus erros e acertos. As próprias plataformas de jogo, como por exemplo `chess.com` e `lichess.org`, tornam essas análises acessíveis disponibilizando reportes das partidas, onde os jogadores podem revisitar os erros cometidos e tentar encontrar uma jogada melhor nessas posições. As plataformas também disponibilizam ambientes de treino onde uma posição é apresentada ao jogador e ele tem que encontrar o melhor movimento possível.

O conceito de erro nas partidas está presente tanto nas análises profissionais quanto nos estudos amadores. Geralmente, os erros são encontrados através das análises das *engines* nas posições. O movimento do jogador é comparado ao que o computador julgou ser o melhor lance, um erro é apontado quando a jogada escolhida pelo enxadrista for muito inferior ao que o computador sugeriu.

Uma das vertentes exploradas nos últimos anos pelos pesquisadores na área de ciência da computação é a análise de tomada de decisões e previsão de erros [McIlroy-Young et al. 2020b]. A tarefa de previsão de erros pode trazer diversos insumos no campo de aprendizagem do jogo, por exemplo, prever quando erros ocorrerão em partidas de xadrez pode ajudar na tarefa de sugerir treinos personalizados para diferentes jogadores, e isso pode ser incorporado nos programas de treinos que as plataformas de treino proporcionam para os jogadores. Além disso, o xadrez é um bom modelo de estudo nesse sentido porque é um domínio onde as tomadas de decisões podem ser satisfatoriamente avaliadas por um computador, visto que as *engines* há vários anos já ultrapassaram os melhores jogadores do mundo.

Esse trabalho apresenta uma nova abordagem no contexto de previsão de erros em partidas de xadrez, usando aspectos básicos do domínio do jogo como forma de representação das posições e de suas características. Essa abordagem combina uma estratégia de representação de posições por meio de grafos, que já é existente na literatura, com o desenvolvimento de novas *features* voltadas a previsão de erros. Essas novas *features* vão ser usadas como entrada para modelos de previsão que serão treinados para prever quando erros ocorrerão nas partidas.

Os resultados obtidos mostraram que as *features* tornam o processo de previsão

de erros mais simples, do ponto de vista computacional, se comparado ao estado da arte, resultando em melhores acurácias nas predições.

2. Trabalhos Relacionados

O xadrez é usado como modelo de estudo em diversos trabalhos recentes. [Brown et al. 2017] desenvolveu uma ferramenta para o auxílio no estudo de partidas de xadrez, através do uso de aprendizagem supervisionada e não supervisionada, a ferramenta desenvolvida foi capaz de agrupar partidas parecidas, com o objetivo de facilitar o estudo diminuindo o número de partidas a serem estudadas, e também reconhecer posições que levam a uma probabilidade alta de vitória.

O uso de *engines* no estudo do xadrez é amplamente empregado na literatura. [Biswas and Regan 2015] as utilizou para desenvolver métricas quantitativas de conceitos como dificuldade, complexidade, profundidade e discriminação. [McIlroy-Young et al. 2020b] as utilizou para desenvolver modelos de previsão de comportamento personalizado. [Zegners et al. 2020] estudou o processo de tomada de decisões de jogadores profissionais, também com o uso de *engines*. [Chabris 2017] sugere que através de ferramentas computacionais pode-se não só apontar os erros cometidos, mas também analisar que condições levam a eles, e isso pode ser usado para prover treinos que auxiliem no desenvolvimento do jogo.

Assim como neste trabalho, [Anderson et al. 2017] também teve como objetivo prever erros em partidas de xadrez, mas especificamente nos finais das partidas. Os autores preferiram utilizar somente finais de jogos com 6 ou menos peças, para fazer uso das tabelas de finais já completamente desvendadas pelos computadores. Essa estratégia difere da apresentada neste trabalho, onde serão utilizadas avaliações de *engines* onde não há garantia de verdade absoluta, entretanto, existe a vantagem prática de poder abranger o jogo em todas as suas fases: aberturas, meios e finais.

[Anderson et al. 2017] levaram em conta três aspectos nos modelos de previsão: tempo disponível, dificuldade da tomada de decisão e a habilidade do tomador de decisões. O trabalho dos autores se restringe a essas três variáveis para que possa ser facilmente extrapolado a outros domínios. Neste trabalho não haverá essa limitação e o domínio específico do objeto de estudo será explorado com o objetivo de obter resultados melhores nas previsões. O presente trabalho considera a hipótese de que ter significativamente maior assertividade na previsão de erros é compensação suficiente para ter que adaptar a solução encontrada a outros domínios.

Na área de inteligência artificial, [McIlroy-Young et al. 2020a] estudou não só a previsão de erros mas também a previsão de movimentos de acordo com o nível de habilidade do jogador. Os autores utilizaram a estratégia de modificar *engines* de código aberto para prever os movimentos, já para prever erros foram utilizadas redes neurais. Outros trabalhos também tiveram os erros em partidas de xadrez como objeto de estudo, como [Biswas and Regan 2015], que descobriram uma correlação entre o nível de habilidade do jogador e o quão fundo analisam uma posição, fazendo o uso de *engines* para comparar o movimento sugerido pelo computador em determinada profundidade com o movimento escolhido pelo jogador.

Os trabalhos relacionados à previsão de erros em jogos de xadrez utilizam diferentes abordagens na tarefa de representar as posições do jogo, seja através do desenvolvi-

mento de métricas [Anderson et al. 2017] ou da representação de cada uma das peças para cada casa do tabuleiro, conforme praticado por [McIlroy-Young et al. 2020a]. Estratégias mais simples também podem ser adotadas, como a notação FEN, que é capaz de sintetizar a posição em uma cadeia de menos de 100 caracteres e que é a forma como a maioria dos programas de computador a entende. O problema do uso da notação FEN é que, apesar de ser muito eficiente na gravação e recuperação das posições, sua interpretação depende das regras definidas em sua concepção, o que geralmente não está presente nos modelos de previsão.

O uso de métricas como entrada para modelos de previsão é a alternativa adotada por [Anderson et al. 2017] para cumprir a tarefa de extrair características das posições de xadrez. Os autores obtiveram resultados significativos na tarefa de predição de erros, mas em seu trabalho os autores consideraram apenas posições de jogo simplificadas, mais especificamente nas finais com 6 ou menos peças. Essa abordagem adotada, além de possibilitar a prerrogativa de jogador perfeito, uma vez que essas posições já foram totalmente desveladas por computadores, também facilita o cálculo das métricas desenvolvidas, que possivelmente teriam um alto custo computacional em posições mais complexas, ou seja, com maior número de peças e, portanto, possibilidades de movimentos. Por outro lado, a representação adotada por [McIlroy-Young et al. 2020a] gera uma entrada complexa e o custo computacional da etapa de treino dos modelos pode aumentar.

O trabalho de [Farren et al. 2013] apresentou uma nova forma de representação das posições, os autores utilizaram grafos com o objetivo de prever o resultado final do jogo. A representação das posições de xadrez por meio de grafos possibilitou o treinamento de modelos capazes de prever o resultado do jogo. Neste trabalho será usada essa representação em grafos de uma forma diferente da apresentada pelos autores, principalmente porque os objetivos são diferentes. Para este trabalho é mais interessante capturar nuances de cada posição do que saber que lado tem mais chances de vitória, dessa forma, serão desenvolvidas novas *features* que serão extraídas dos grafos gerados.

3. Métodos

Nesta seção serão apresentados os métodos utilizados para o desenvolvimento da estratégia proposta, realização dos testes e obtenção dos resultados. Primeiro, será apresentada a definição de erro e o objetivo do trabalho, depois será descrito o processo de obtenção dos dados. A sessão seguinte detalha as etapas de pré-processamento e logo em seguida a sessão de *Features* elucida como os grafos foram criados e como as características das posições foram extraídas deles. Por último, a sessão Modelo descreve os modelos de predição utilizados e suas configurações.

3.1. Erro em partidas de xadrez

Existem diferentes maneiras de se definir o que é um erro em uma partida de xadrez, nesse trabalho será definido como erro um movimento que diminui significativamente a chance de vitória de um jogador, sendo assim, movimentos podem ser erros, mas posições não. Entretanto, por simplificação, a terminologia utilizada é a de ”posições erro”e ”posições não-erro”, isto por que o objetivo do trabalho é prever quais posições originam erros em partidas de xadrez e, para cumprir esse objetivo, serão estudadas as posições nas quais os movimentos são realizados, visto que, para a etapa de predição, não existem informações do movimento, pois ele ainda não aconteceu.

Para classificar as posições em erros e não-erros, serão utilizadas as avaliações das *engines*. Uma avaliação da posição é um valor decimal que expressa quem está melhor no jogo, se positivo as peças brancas têm vantagem, se negativo as pretas têm melhor posição, o zero significa igualdade. É importante ressaltar que não há comprovação de que a análise da *engine* é uma verdade absoluta, mas considerou-se a hipótese de que, para os fins deste trabalho, a avaliação é suficiente para identificar erros nas partidas de jogadores amadores.

A estratégia utilizada para essa classificação é a mesma adotada por [McIlroy-Young et al. 2020a] e consiste em calcular previamente a probabilidade empírica de vitória de cada avaliação da *engine* (arredondando em duas casas decimais), ou seja, dada uma avaliação da *engine*, obteve-se a proporção de vitória baseada em todas as ocorrências dessa avaliação em nossa base de partidas. Com a probabilidade empírica de vitória de cada avaliação, definimos como erro as posições que decrescem essa probabilidade em 10%. A Figura 1 mostra a probabilidade empírica de vitória por valor de avaliação de *engine*.

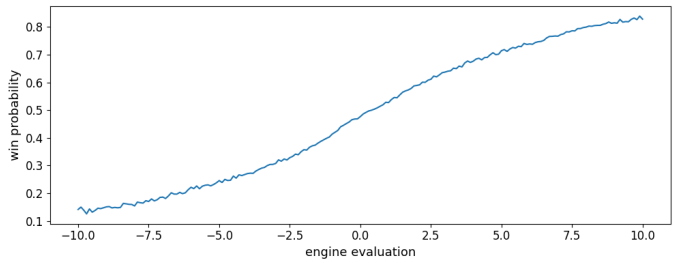


Figura 1. Probabilidade empírica de vitória na base de partidas utilizada.

Através da Figura 1 é possível verificar que uma avaliação de *engine* igual a 0.00 tem uma probabilidade de vitória de aproximadamente 50%. Uma posição de tabuleiro com essa avaliação é classificada como ”erro”se a avaliação da próxima posição do tabuleiro após o movimento decrescer essa probabilidade em 10 pontos percentuais ou mais.

3.2. Coleta de dados

A coleção de dados utilizada nessa pesquisa foi obtida em `database.lichess.org`. Esta coleção é aberta e contém todas as partidas jogadas na plataforma separadas por período. Para esse trabalho, considerou-se o período do mês de novembro de 2019. As partidas obtidas estão codificadas em PGN (*Portable Game Notation*). Essa notação possibilita não somente a gravação dos movimentos, ou seja, a partida em si, mas também informações como tempo no relógio, avaliação da *engine*, pontuação dos jogadores, entre outras. A base de dados obtida tem um total de 40.357.832 partidas que correspondem a 80 GB de dados (se descompactada).

3.3. Pré-processamento

Após baixar os arquivos PGN da base do *lichess*, algumas etapas de pré-processamentos foram realizadas, a primeira delas diz respeito às modalidades de tempo de jogo. Por se tratarem de modos de jogo onde as tomadas de decisão são realizadas com pouco tempo disponível, partidas das modalidades *bullet* e *ultra bullet* não foram consideradas neste trabalho.

Com o objetivo de realizar as análises dos erros cometidos pelos jogadores, também foi necessário avaliar posição por posição de cada uma das partidas da base, o que traz um custo computacional relativamente alto. Como é inviável realizar todo o pré-processamento em toda a base de partidas, foi utilizado o *rating*, que é uma forma de ranquear os enxadristas de acordo com o seu desempenho em torneios e partidas, para separar conjuntos de partidas que abrangessem diversos níveis de jogadores. Foi realizada uma amostra aleatória de dez mil partidas em cada faixa de *rating*, sendo cada faixa dividida em 100 pontos de *rate*, começando da faixa de 1100-1199 até 1900-1999. Depois da amostragem, as faixas de rate foram agrupadas novamente e embaralhadas em uma mesma base de partidas, que foi posteriormente usada para obter os resultados.

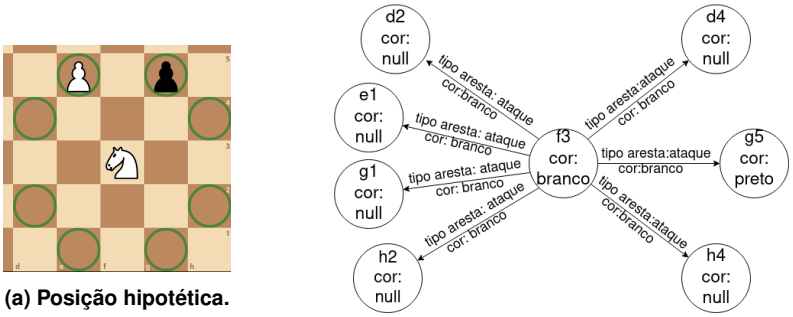
Com as amostras das partidas realizadas, a avaliação das posições foi realizada pela *engine* `stockfishchess.org`, que é de código aberto. Foram adicionadas, em cada posição, comentários com informações como: a avaliação da *engine* na posição atual; o melhor movimento segundo a *engine* e o tempo disponível para realização da jogada, sendo ele relativo e também em segundos. Foi utilizada a profundidade 12 nas análises realizadas, ou seja, a *engine* só avaliou as posições em até 12 movimentos futuros.

Após realizada a avaliação das posições, foi realizado o processo de classificação das posições em ”erro”e ”não-erro”conforme descrito na sessão anterior. Foi identificada a necessidade de balancear a base de acordo com a classificação, para que cada classe represente 50% do número total de posições, para tal, foi aplicada a mesma técnica utilizada por [McIlroy-Young et al. 2020a], que consistiu em realizar *downsample* nas posições ”não-erros”de forma a se obter a base de dados balanceada. Esse balanceamento pode tornar o problema de previsão mais fácil e os efeitos dele em cenários reais de previsão precisam ser investigados com mais profundidade em trabalhos futuros, mas como essa estratégia já é utilizada em trabalhos semelhantes na literatura, trabalha-se com a hipótese de que o balanceamento não invalida os resultados apresentados nesse trabalho.

3.4. *Features*

Com as partidas já amostradas e avaliadas, iniciou-se o processo de extração de *features*. Essa etapa tem como objetivo auxiliar o modelo de previsão de erros com conhecimentos específicos do jogo de xadrez já abstraídos. A ideia é traduzir as posições em conceitos básicos presentes nos estudos teóricos do jogo, como por exemplo o desenvolvimento das peças, influência central e peças sobrecarregadas. Para auxiliar nesse processo de abstração, adaptou-se a representação proposta por [Farren et al. 2013], onde cada posição do jogo pode ser representado por grafos direcionados, que por sua vez representam as peças, posições do tabuleiro e suas interações. Dois dos grafos apresentados pelos autores serão utilizados nessa pesquisa, o de posição e o de suporte, que aqui são denominadas mobilidade e atividade, respectivamente.

A Figura 2b representa o grafo de mobilidade extraído da posição da Figura 2a. Este grafo é direcionado e representa as peças e as possibilidades de movimento. Assim, o grafo pode ser entendido como o quão restritas ou não estão as peças. No exemplo da Figura 2b, o cavalo pode movimentar-se para 7 casas e é representado pelo nó “f3”, algumas dessas casas estão ocupadas por peças e possuem a propriedade “cor” preenchida de acordo com a cor da peça inimiga. Note que nunca haverá, no grafo de mobilidade, um nó conectado a outro da mesma cor, visto que seria um movimento ilegal. Já as



(a) Posição hipotética.

(b) Exemplo de grafo de mobilidade.

Figura 2. Exemplo de posição e grafo de mobilidade. Jogam as peças brancas. As casas que o cavalo ataca estão marcadas com um círculo.

casas vazias têm essa propriedade nula. O nó que representa a casa atual do cavalo está conectado as 7 possíveis casas de destino através das arestas, as arestas sempre têm a propriedade "tipo_aresta" com o valor "ataque" e a "cor" da mesma cor da peça de origem.

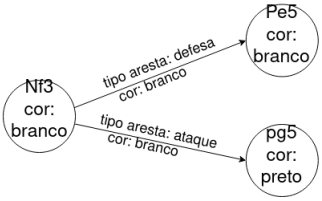


Figura 3. Exemplo de grafo de atividade.

Já o grafo de atividade trata somente da relação das peças com outras peças. Este grafo é útil para representar aspectos como, por exemplo, quantas peças inimigas a rainha está atacando ou quantos peões estão defendendo o rei. A Figura 3 exemplifica esse grafo na mesma posição da Figura 2a. Nela, pode-se observar que o cavalo, representado pelo nó "kf3" interage com duas outras peças, um peão amigo e um inimigo, "Pe5" e "pg5". É possível observar que além da propriedade "cor", também usou-se a caixa alta pra distinguir a cor das peças. A relação entre as peças é definida pela propriedade "tipo_aresta", que nesse grafo pode ter os valores "defesa" e "ataque".

Para cada posição foi efetuada a construção do grafo de mobilidade e atividade, e o seguinte conjunto de *features* foi extraído deles:

- "peões atacados": soma dos graus de entrada dos nós do tipo peão no grafo de atividade, somente arestas do tipo ataque;
- "peças atacadas": soma dos graus de entrada dos nós do grafo de atividade, somente arestas do tipo ataque;
- "bispos atacados": soma dos graus de entrada dos nós do tipo bispo no grafo de atividade, somente arestas do tipo ataque;
- "influência central": soma da quantidade de nós nas casas D4, D5, E4 e E5, com a soma dos graus de entrada dos nós que representam essas casas no grafo de mobilidade;
- "atividade dos cavalos": soma dos graus de saída dos nós do tipo cavalo do grafo de atividade;

- ”número de defensores do rei”: soma dos graus do nó que representa o rei no grafo de atividade, somente arestas de defesa;
- ”peças sobrecarregadas”: soma dos nós que possuem grau de saída em arestas do tipo defesa maior que 1, cujos nós defendidos tem grau de entrada do tipo defesa igual a 1.
- ”atividade dos peões”: soma dos graus de saída dos nós do tipo peão no grafo de atividade;
- ”mobilidade dos peões”: soma dos graus de saída dos nós do tipo peão no grafo de mobilidade;
- ”atividade das peças”: soma dos graus de saída dos nós no grafo de atividade;
- ”mobilidade das peças”: soma dos graus de saída dos nós no grafo de mobilidade;
- ”peças sem defensores”: quantidade de nós no grafo de atividade com grau de entrada igual a 0, somente arestas do tipo defesa;
- ”atividade das rainhas”: soma dos graus de saída dos nós do tipo rainha no grafo de atividade;
- ”mobilidade das rainhas”: soma dos graus de saída dos nós do tipo rainha no grafo de mobilidade;
- ”mobilidade das torres”: soma dos graus de saída dos nós do tipo torre no grafo de mobilidade;
- ”peças sem desenvolvimento”: número de nós que não sejam do tipo peão que permanecem nas casas iniciais;
- ”peões em jogo”: número de peões no tabuleiro;

Cada uma dessas *features* extraídas dos grafos são computadas duas vezes, do ponto de vista do jogador com a vez e do ponto de vista do seu oponente, gerando assim *features* como ”mobilidade das peças ativo”e ”mobilidade das peças oponente”. A única exceção a isso é a *feature* ”peões em jogo”, que é computada somente uma vez. Também são retiradas *features* de metadado das posições, essas *features* foram desenvolvidas sem o auxílio dos grafos:

- ”*rate*”: nível do jogador ativo;
- ”*rate* oponente”: nível de habilidade do oponente;
- ”avaliação média profundidade 1”: avaliação média da *engine* nos 5 melhores movimentos;
- ”melhor movimento”: peça a ser movida sugerida pela *engine*;
- ”relógio”: tempo restante em segundos;
- ”proporção relógio”: fração de tempo restante em segundos;
- ”horário do dia”: horário da partida;
- ”tempo *engine*”: tempo necessário para *engine* avaliar a posição (tentativa de capturar a dificuldade da posição);
- ”avaliação”: avaliação da *engine* na posição atual;
- ”avaliação jogada passada”: avaliação da *engine* antes do último movimento;
- ”avaliação 2 jogadas passadas”: avaliação da *engine* antes dos dois últimos movimentos;
- ”avaliação 3 jogadas passadas”: avaliação da *engine* antes dos três últimos movimentos;
- ”avaliação 4 jogadas passadas”: avaliação da *engine* antes dos quatro últimos movimentos;

- ”ply”: contador de movimentos;

É importante observar que os conceitos abstraídos nas posições não se tratam de teorias complexas que requerem conhecimento profundo do jogo, são na verdade uma maneira diferente, e simples, de representar as posições de xadrez. O resultado do processo de extração aqui descrito é um arquivo no formato CSV (*Comma Separated Values*) onde cada posição em nossa amostragem de partidas é uma linha e cada *feature* extraída, uma coluna. No total, nosso processo de extração gerou 47, 33 *feature* do tabuleiro e 14 metadados.

3.5. Modelo

O modelo escolhido para os testes foi uma rede neural, treinada com o *framework* Keras (`https://keras.io/`). Testes preliminares obtiveram melhores resultados com a configuração de rede contendo três camadas ocultas com 1024, 512 e 256 neurônios respectivamente, redes com mais neurônios e/ou camadas não resultaram em acurácias significativamente maiores. Cada uma das camadas da rede foi sucedida por uma camada de *Dropout* para evitar *overfitting* durante o treinamento.

Seguindo as boas práticas da literatura de aprendizado de máquina, os dados rotulados foram divididos em conjuntos de treino (50%), validação (25%) e teste (25%). Os resultados da próxima seção foram obtidos com o conjunto de testes, o qual não foi considerando durante o processo de treinamento do modelo.

4. Resultados

Nessa sessão será apresentada uma caracterização das *features* resultantes para analisar sua eficiência. Também serão apresentados os comparativos realizados com o estado da arte.

4.1. Caracterização das *features*

Para avaliar a qualidade das *features* desenvolvidas, a Informação Mútua (MI) [Ross 2014] entre as *features* e as classes foi calculada utilizando a base de partidas obtida.

Analisando a Tabela 1a, é possível observar altos valores de MI nas *features* provenientes dos metadados. Comparando a importância das *features* de metadados com os demais, é possível medir a qualidade das *features* desenvolvidos. Para entender quais tipos de *feature* podem trazer melhores ganhos, também serão apresentados os resultados de forma crescente. A tabela 1b mostra as *features* que possuem pior ganho de MI, a maioria delas não está na lista da sessão anterior por terem sido consideradas fracas. Essas *features* fracas foram testadas e avaliadas mas não fizeram parte do conjunto que gerou os resultados comparativos mostrados mais adiante. Algumas *features* obtiveram MI inferior a 0,01 % e foram ocultados da tabela e descartadas.

Pode-se observar comparando as Tabelas 1a e 1b que as melhores *features* desenvolvidas na verdade são mais simples e estão presentes de forma mais abrangente no jogo de xadrez. Tomemos como exemplo a *feature* ”raio-x torre-rei oponente”que foi uma tentativa de representar um conceito um pouco mais avançado do jogo chamado ”cravadura”, ela faz parte de uma série de *features* desenvolvidas para representar o mesmo conceito,

Feature	*Mutual Information* (%)
avaliação	10,57
avaliação média profundidade 1	9,14
avaliação jogada passada	7,18
avaliação 2 jogadas passadas	6,47
avaliação 3 jogadas passadas	5,77
avaliação 4 jogadas passadas	5,54
horário do dia	4,59
tempo *engine*	2,91
atividade das peças ativo	2,79
mobilidade das peças ativo	2,76
atividade das peças oponente	2,54
atividade das torres ativo	2,50

(a) Top-12 *features* ordenadas por *Mutual Information*.

Feature	*Mutual Information* (%)
cavalos atacados ativo	0,03
melhor movimento rainha	0,05
torres atacadas ativo	0,06
rei atacado oponente	0,10
torres atacada oponente	0,10
peças sem defensores oponente	0,10
peões isolados oponente	0,10
bispos atacados oponente	0,10
raio-x torre-rei oponente	0,11
defensores do rei atacados oponente	0,12
raio-x bispo-rainha oponente	0,12
defensores do rei atacados ativo	0,14

(b) Top-12 piores *features* ordenadas por *Mutual Information*.

Tabela 1. Comparação entre *features* por MI

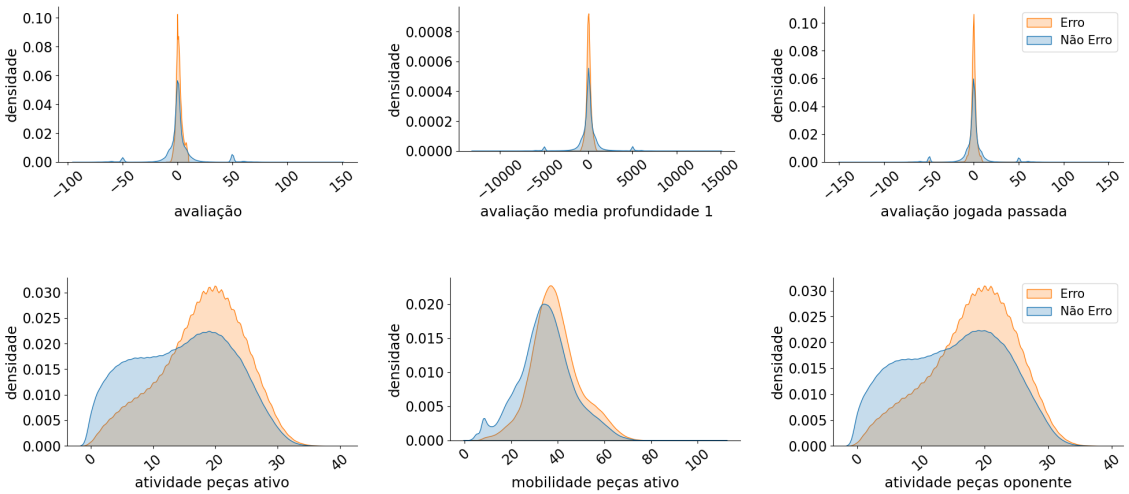


Figura 4. Densidade das *Features*

mas nenhuma delas foi de grande valia e tiveram pouco ganho de MI. Outro exemplo é o conjunto de *features* que representam o conceito de ”peças atacadas”, embora as que representam o conjunto de peças atacadas como um todo foram positivas, as *features* que representam peças atacadas específicas não foram tão bem, como ”torres atacadas”e ”bispos atacados”, que acabaram tendo pouco ganho de MI. Dessas análises foi elaborada a hipótese de que *features* muito específicas e pouco abrangentes como as comentadas anteriormente não são de grande valia, mas as que representam conceitos mais abrangentes acabam tendo um ganho significativo de MI e são úteis no problema a ser resolvido.

Para validar a hipótese de que as *features* desenvolvidas com base nos grafos são úteis em separar as posições ”erro”das posições ”não-erro”, a Figura 4 apresenta a diferença da densidade das três melhores *features* de metadados (primeira linha) e das três melhores *features* baseadas em grafos (segunda linha) segundo o percentual de MI, comparando-se as classes ”erro”e ”não-erro”.

A partir da Figura 4 podemos observar que tanto as *features* de metadados quanto as *features* desenvolvidas baseadas em grafos, apresentam diferenças na distribuição ao se comparar posições ”erro”e ”não-erro”, corroborando a hipótese de que as *features* baseadas em grafos são importantes e podem ser úteis na tarefa de prever erros em jogos de xadrez.

4.2. Predições

Para avaliar as *features* desenvolvidas, utilizamos o modelo descrito nas sessões anteriores, para comparação de desempenho, também processamos a mesma base de dados utilizada em nosso trabalho no código desenvolvido e disponibilizado por [McIlroy-Young et al. 2020a]. A Tabela 2 mostra a comparação entre as acurácias obtidas com as *features* desenvolvidas aplicadas no modelo descrito anteriormente, e também as acurácias obtidas com o modelo de [McIlroy-Young et al. 2020a]. As bases de dados utilizadas foram as mesmas, e a configuração do modelo dos resultados da Rede Neural Convolucional Residual (Residual CNN) de [McIlroy-Young et al. 2020a] foram as mesmas descritas em seu trabalho.

Estratégia	Tabuleiro e metadados	Somente Tabuleiro
Residual CNN	0,67	0,64
Estratégia desenvolvida	0,74	0,65

Tabela 2. Comparação com o estado da arte

A Tabela 2 mostra os resultados obtidos com *features* dos metadados e do tabuleiro e também com o uso somente de *features* do tabuleiro. Podemos observar significativa melhora na acurácia da estratégia desenvolvida, principalmente no conjunto com *features* de metadados e tabuleiro.

Além das acurácias obtidas demostradas nas tabelas é importante ressaltar que o tempo gasto na estratégia desenvolvida para a etapa de construção dos grafos e treinamento do modelo foi significativamente menor do que o tempo necessário para treinar a Residual CNN nas configurações propostas por [McIlroy-Young et al. 2020a]. A etapa de construção dos grafos da estratégia desenvolvida demorou em média uma hora e quarenta minutos pra nove bases de dez mil partidas cada em um processador de 40 núcleos, já a etapa de treinamento dos modelos foi executada no mesmo hardware (RTX 1650). O modelo desenvolvido levou cerca de 5 minutos para ser treinado, totalizando menos de 2 horas, contando a construção dos grafos, extração de *features* e treinamento do modelo. Por outro lado, a Residual CNN levou cerca de 12 horas totais nas configurações propostas pelos autores.

Os resultados obtidos mostram que a estratégia de extração de *features* apresentada além de tornar o modelo mais simples, melhorando o custo computacional com a diminuição da dimensionalidade, também o torna significativamente mais acurado.

5. Conclusão

Este estudo apresentou uma nova abordagem na tarefa de previsão de erros em partidas de xadrez. Foram desenvolvidas *features* baseadas em grafos que se mostraram eficazes em abstrair das posições conceitos básicos do jogo. Essa estratégia pode ser empregada em outros domínios com o mesmo objetivo de simplificar e abstrair conceitos específicos do objeto de estudo.

Foram realizados testes em diferentes base de dados, separados por nível de habilidade dos jogadores e também em bases unificadas. Os resultados encontrados foram tão bons quanto os de trabalhos recentes, mas a estratégia desenvolvida tem a vantagem de ter menos complexidade computacional.

Como trabalhos futuros, pretende-se expandir o conjunto de *features* e os modelos utilizados, afim de melhorar os resultados obtidos. Também há interesse em mesclar as diferentes abordagens propostas pelos trabalhos relacionados com proposta deste trabalho para avaliar o desempenho obtido. Foi identificada também a necessidade de entender como o balanceamento pode afetar as predições no mundo real, pois apesar dessa ser a estratégia adotada na literatura, é interessante procurar por melhores alternativas para não perder qualidade nas predições práticas.

Referências

Anderson, A., Kleinberg, J., and Mullainathan, S. (2017). Assessing human error against a benchmark of perfection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):1–25.

Biswas, T. and Regan, K. (2015). Measuring level-k reasoning, satisficing, and human error in game-play data. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 941–947. IEEE.

Brown, J. A., Cuzzocrea, A., Kresta, M., Kristjanson, K. D., Leung, C. K., and Tebinka, T. W. (2017). A machine learning tool for supporting advanced knowledge discovery from chess game data. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 649–654. IEEE.

Chabris, C. F. (2017). Six suggestions for research on games in cognitive science. *Topics in cognitive science*, 9(2):497–509.

Farren, D., Templeton, D., and Wang, M. (2013). Analysis of networks in chess. Technical report, Stanford University, Tech. Rep.

McIlroy-Young, R., Sen, S., Kleinberg, J., and Anderson, A. (2020a). Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1677–1687.

McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J., and Anderson, A. (2020b). Learning personalized models of human behavior in chess. *arXiv preprint arXiv:2008.10086*.

Ross, B. C. (2014). Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357.

Statt, N. The Verge esports giant tsm signs hikaru nakamura, its first pro chess player.

Zegners, D., Sunde, U., and Strittmatter, A. (2020). Decisions and performance under bounded rationality: A computational benchmarking approach.