Universidade de Brasília
Departamento de Estatística

Interpretação de redes neurais

Davi Guerra Alves

Projeto apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2023

Davi Guerra Alves

Interpretação de redes neurais

Orientador(a): Thais Carvalho Valadares Rodrigues

Projeto apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2022

1 Referencial teórico

1.1 Regressão logística

A regressão logística é um método estatístico utilizado para modelar a probabilidade de uma variável dependente categórica. É comumente utilizada para problemas de classificação binária, onde a variável dependente possui apenas duas categorias, como sim/não, positivo/negativo, 0/1.

A regressão logística é baseada na combinação linear entre as variáveis explicativas e os coeficientes lineares, cuja fórmula é:

$$Y = \beta_0 + X_1\beta_1 + X_2\beta_2 + \ldots + X_k\beta_k, \tag{1.1.1}$$

onde cada variável explicativa $(X_1, X_2, ..., X_k)$ tem um parâmetro β correspondente, influenciando o resultado de Y.

O cálculo da regressão logística é baseado na probabilidade do Y ser igual a 1 $(P(Y = 1))$, logo para se realizar esse cálculo, é passado uma função logística no Y, cuja forma é dada por:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + X_1\beta_1 + X_2\beta_2 + ... + X_k\beta_k)}} \tag{1.1.2}$$

A estimação dos coeficientes $(b_0, b_1, b_2, ..., b_k)$ na regressão logística é geralmente realizada por meio do método da máxima verossimilhança. O objetivo é encontrar os valores dos coeficientes que maximizam a função de verossimilhança, representando a probabilidade de observar os dados observados dado o modelo.

A função de verossimilhança (L) para a regressão logística é dada pelo produto das probabilidades condicionais de observar os eventos(valores da variável dependente) dados os valores das variáveis independentes. Para facilitar o cálculo, geralmente trabalhamos com o logaritmo natural da função de verossimilhança, conhecido como log-verossimilhança(l).

A log-verossimilhança para a regressão logística é:

$$l(\beta) = \sum_{i=1}^{n} [y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})] \tag{1.1.3}$$

- Né o número total de observações.
- y_i é a variável dependente binária da i-ésima observação (0 ou 1).

- p_i é a probabilidade predita de $Y = 1$ para a i-ésima observação, dada pela função logística.

A ideia é encontrar os valores de $(b_0, b_1, b_2, ..., b_k)$ que maximizam essa função. Isso geralmente é feito usando métodos computacionais, como o algoritmo de otimização Newton-Raphson ou o Gradiente Descendente.

Um ponto a ser destacado é a função log odds ou logíto. Ela é uma função que calcula o log da razão do evento ser acontecer e dele não acontecer, cuja formulação é dada por:

$$\text{logit}(P(Y = 1)) = \log\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) \tag{1.1.4}$$

Onde esse resultado nada mais é do que $\beta_0 + X_1\beta_1 + X_2\beta_2 + \ldots + X_k\beta_k$.

Portanto, a utilizade de se analisar o log-odds é justamente um ponte entre olhar coeficiente e a probabilidade final, pois um coeficiente positivo indica que o aumento na variável está associado a um aumento nas log-odds (e, portanto, na probabilidade), enquanto um coeficiente negativo está associado a uma diminuição nas log-odds (e na probabilidade).

1.2 Redes neurais artificiais

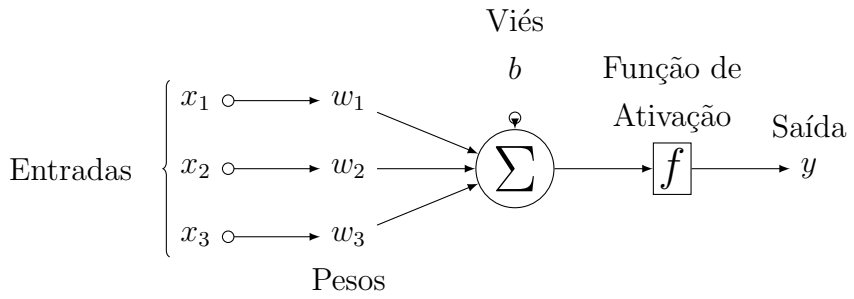
Redes Neurais Artificiais (ou *Deep Learning*) é uma técnica preditiva presente no campo de Inteligência Artificial. As redes neurais tem sido amplamente utilizadas devido ao seu alto poder preditivo e também à flexibilidade de se aplicar esse método em diversos contextos, permitindo ser um modelo com menos restrições que os modelos tradicionais estatísticos.

1.2.1 Neurônio

Uma rede neural tem esse nome devido à tentativa de se reproduzir o comportamento do cérebro humano. Sua arquitetura é composta por um conjunto de unidades denominadas neurônios, e cada neurônio é responsável por receber informações, fazer o tratamento do que foi recebido, e repassar o resultado disso para frente. A Figura **??** ilustra a estrutura de 1 neurônio. Quando as informações x_i entram no neurônio, acontece primeiramente um processo onde é ponderada cada informação que foi recebida, os chamados **pesos**. Logo em seguida ocorre a soma dessa ponderação. Feito isso, é realizado mais um processo de soma, agora adicionando uma informação própria daquele neurônio nesse resultado. Essa informação é chamada de **bias** (ou Viés). Antes desse resultado

ser repassado para outro neurônio, ele passa por uma função que vai definir a natureza daquela informação, chamada de **função de ativação**, retornando assim uma saída y.

Figura 1: Neurônio da Rede Neural

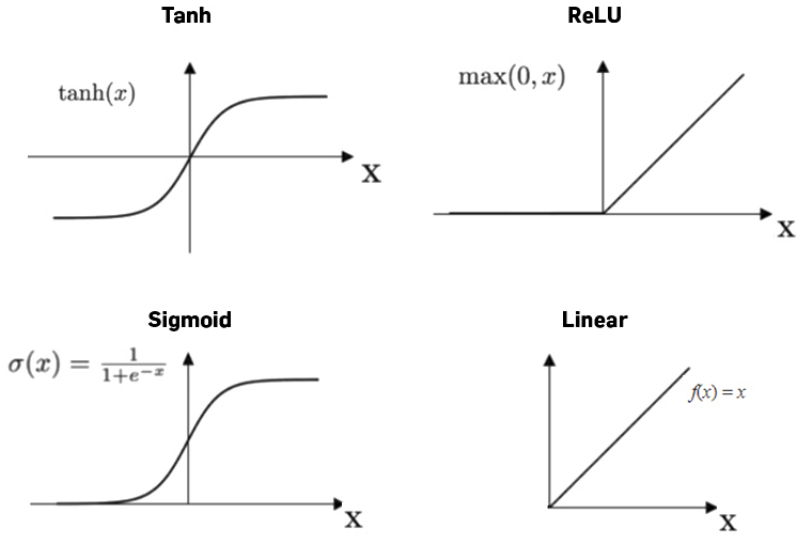


A Figura **??** pode ser representada matematicamente da seguinte maneira:

$$y = f(\beta + \sum_{i=1}^{d_x} w_i x_i) \tag{1.2.1}$$

onde d_x é o número de entradas.

Figura 2: Tipos de Função de Ativação.



Fonte: https://machine-learning.paperspace.com/wiki/activation-function

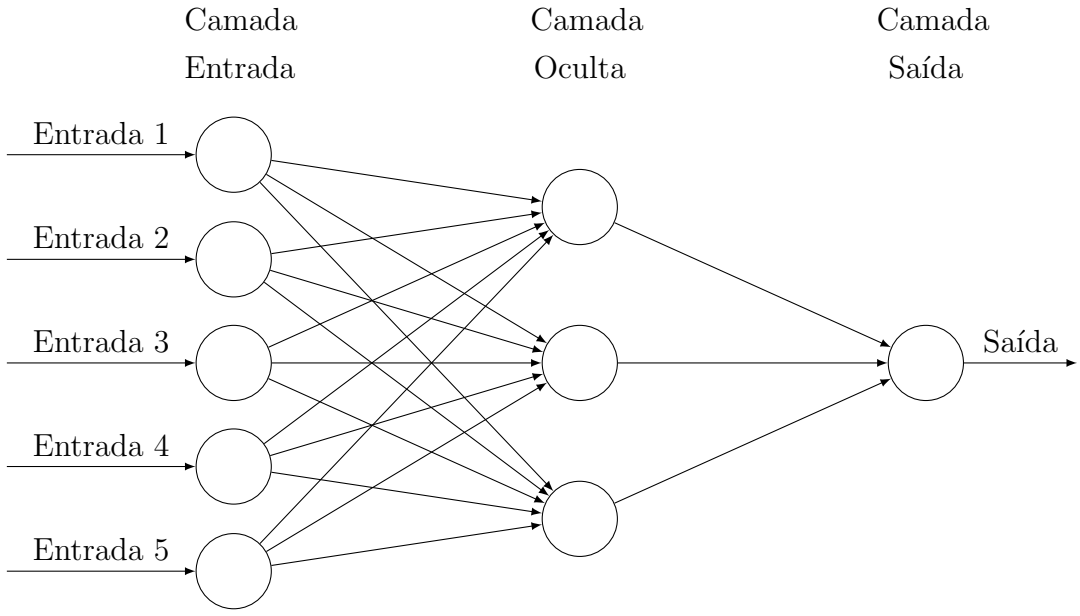
A Figura **??** mostra alguns tipos de funções de ativação que um neurônio pode ser atribuído. Note como a maioria dessas funções restringe o valor de x (nesse caso o valor calculado no neurônio), no caso da função Sigmoide e da Tangente hiperbólica (tanh) limitando o valor do neurônio em um intervalo, a ReLU, que é bastante utilizada, desconsidera os valores negativos e existe a função Linear que basicamente só vai repassar

a informação do neurônio para frente.

1.2.2 Arquitetura

Uma rede neural é estruturada em camadas formadas por um conjunto de neurônios. Conforme ilustrado na Figura **??**, temos as camadas de entrada, as camadas ocultas e a camada de saída. A camada de entrada é o ponto de partida da rede neural, pois é onde as informações das variáveis entram. Logo em seguida encontram-se as camadas ocultas, que são as principais responsáveis por criar redes mais complexas, pois o número de camadas e o número de neurônio dentro dessas camadas podem ser moldados ou adicionados dependendo do objetivo empregado pela rede, conforme ilustrado na Figura **??**. E por fim existe a camada de saída que contém o(s) valor(es) predito(s) pela rede.

Figura 3: Rede Neural com uma camada oculta



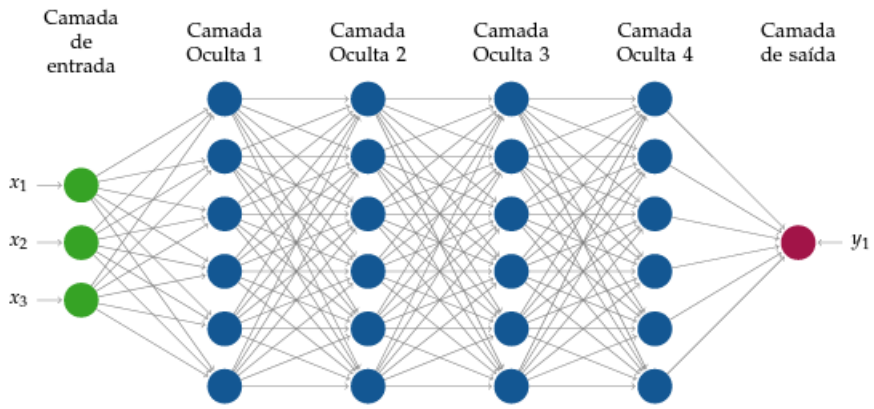


Figura 4: Arquitetura padrão de um rede neural *feedforward* **?**

Note que as Figuras **??** e **??** evidenciam um potencial muito grande de crescimento da rede e naturalmente esse aumento pode acabar gerando um custo computacional elevado quando a rede estiver em treinamento.

1.2.3 *Forward propagation*

O processo *Forward propagation* (ou propagação direta) é o responsável por transmitir as informações, desde a camada de entrada, passando pelas camadas ocultas, até chegar na camada de saída. O *Forward propagation* utiliza da generalização a Equação **??** para cada neurônio presente nas camadas internas da rede neural. Por isso temos que, para cada j-ésimo neurônio, da camada l:

$$z_j^{(l)} = b_j^{(l)} + \sum_{i=1}^{d_{(l-1)}} w_{ij} a_i^{(l-1)}$$

onde:

- w_{ij} é o peso associado à conexão entre o neurônio i na camada $l-1$ e o neurônio i na camada l;
- $a_{(i)}^{(l-1)}$ é a saída do neurônio i na camada anterior $(l-1)$;
- $b_j^{(l)}$ é o viés (bias) associado ao neurônio j na camada l.

Logo em seguida é aplicada uma função de ativação g em $z_i^{(l)}$ que vai ser a responsável por gerar o resultado final $a_i^{(l)}$, do o i-ésimo neurônio na l-ésima camada.

$$a_i^{(l)} = g(z_i^{(l)})$$

Esse processo vai ser realizado camada a camada, sequencialmente. Logo, supondo que uma rede neural tenha l camadas ocultas e, cada camada contendo d_l neurônios, considerando também w_{ij} como o peso presente no i-ésimo neurônio com a j-ésima saída na camada seguinte $(l+1)$, onde $l = 0, ..., H$. Temos que o resultado final da propagação é igual a:

$$f(\mathbf{x}) = \mathbf{a}^{H+1} = g(b_j^{(H+1)} + \sum_{i=1}^{d_H} w_{ij} a_i^H) \tag{1.2.2}$$

Note que a previsão da rede vem diretamente do resultado obtido da última camada oculta, e esse depende da camada que o antecede e assim sucessivamente até chegar na camada de entrada.

1.2.4 Função de perda

Para se obter informações sobre o desempenho do modelo, é escolhida uma função de perda. Uma função bastante utilizada é a do erro do quadrático médio:

$$EQM(f) = \frac{1}{n} \sum_{k=1}^{n} (f(\mathbf{x}_k) - y_k)^2$$

Essa função é uma indicadora do quão longe, em média, os valores preditos estão distantes dos valores reais. Note que o resultado da função f depende exclusivamente dos parâmetros da rede (viés e pesos), por isso, se essa função de perda tende a 0, significa que os parâmetros dessa rede alcançaram um ponto mínimo global. Entretanto, devido a complexidade desse modelo, acabam-se escolhidos pontos locais mínimos, que, dependendo do contexto, acabam satisfazendo o objetivo. A Figura **??** ilustra esse comportamento:

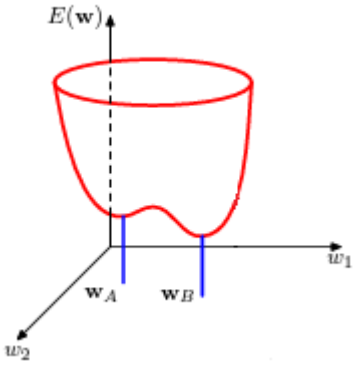


Figura 5: Comportamentos dos pesos em relação à função de perda. O ponto w_A representa um ponto local mínimo e w_B representa um ponto global mínimo. **?**

1.2.5 *Backpropagation*

Como a função de perda está relacionada com os parâmetros (θ) da rede, para se minimizar a função de perda R(θ), é necessário encontrar os valores de θ que resolvam esse problema de otimização. Para fazer isso, é necessário calcular o gradiente de R(θ) em relação à θ **?**,

$$\nabla R(\theta) = \frac{\partial R(\theta)}{\partial \theta} \tag{1.2.3}$$

A rede neural, durante todo o treinamento, aplica esse processo do cálculo do gradiente de R(θ) em relação à θ. Esse é um processo iterativo, com o objetivo de mudar o valor de θ afim de conseguir minimizar a função de perda. Com isso a Equação **??** pode ser descrita nesse processo iterativo como:

$$\nabla R(\theta^m) = \left.\frac{\partial R(\theta)}{\partial \theta}\right|_{\theta=\theta^m},$$

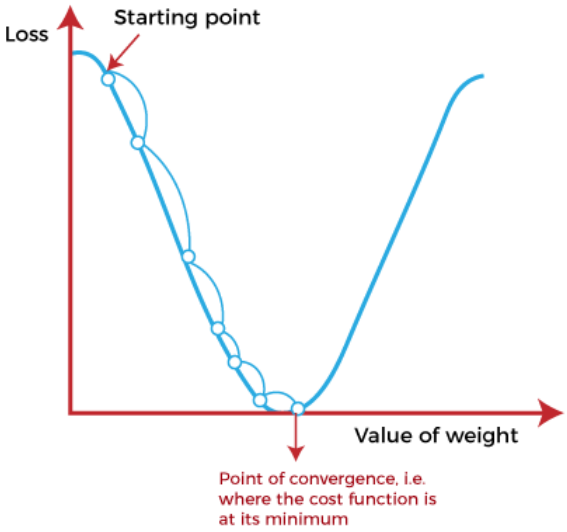
onde $\theta = \theta_m$ significa que o cálculo do gradiente está sendo realizado na iteração m.

E, para conseguir atualizar esse θ, conforme é calculado o gradiente durante as iterações, é utilizada a técnica de gradiente descendente, que pode ser descrita como:

$$\theta^{m+1} \leftarrow \theta^m - \lambda \frac{\partial R(\theta^m)}{\partial \theta^m}$$

sendo λ o parâmetro que vai definir a magnitude de influência da derivada $\frac{\partial R(\theta^m)}{\partial \theta^m}$ em θ^m.

Figura 6: Representação do método do gradiente descendente para a estimação de um parâmetro.



Fonte: https://www.javatpoint.com/gradient-descent-in-machine-learning

A Figura **??** demonstra o processo do gradiente descendente. Os parâmetros são iniciados com algum valor e, conforme ocorre os processos iterativos de aprendizado, o parâmetro converge para um mínimo da função de perda. Note que a distância entre cada ponto é definida pelo λ ou taxa de aprendizado.

Todo esse processo é realizado em cada parâmetro que existe na rede neural. Assim como as informações das variáveis são passadas camada a camada, saindo da camada de entrada, passando pelas camadas ocultas e chegando na camada de saída, visto anteriormente como *Forward propagation*, a informação do resultado da rede na função de perda é passada de forma contrária. O gradiente de cada parâmetro é calculado primeiro nas camadas mais próximas da saída, e essa informação é repassada para trás, chegando até os parâmetros próximos aos da camada de entrada. Esse processo é chamado de *Backpropagation* **?**.

1.3 SHAP

A estrutura de uma rede neural, por mais que proporcione bons resultados, mostra uma deficiência na parte interpretativa. Conhecida por ser uma "caixa-preta"pelo fato de sua estrutura ser muito complexa, existe a necessidade de se entender as predições feitas. Para isso, existem técnicas que abordam o tema de interpretação de modelos de redes neurais e dentro delas existe a técnica SHAP, que através dela é possível entender como as variáveis de entrada influenciam as previsões do modelo, fornecendo *insights* sobre sua lógica e permitindo uma explicação clara e confiável. Isso contribui para a

transparência, confiabilidade e aceitação dos modelos, além de auxiliar na detecção de viéses e discriminação.

1.3.1 Valores de Shapley

Os valores de Shapley foram desenvolvidos por Lloyd Shapley **?** no contexto da teoria de jogos, e essa técnica ganhou força na área de inteligência artificial pela sua capacidade de conseguir interpretar modelos preditivos tidos como "caixa-preta". No método criado por Shapley, existiam uma quantidade de jogadores que exerciam juntos determinada atividade, e o intuito era observar o ganho que um jogador (ou um conjunto de jogadores), obtinha ao ser adicionado para realizar a mesma tarefa, sem a presença do restante do grupo.

Podemos definir $\mathbf{F}$ como o conjunto de jogadores (ou as variáveis explicativas) presentes na atividade, logo $\mathbf{F} = \{1, 2, ..., \mathbf{M}\}$, onde $\mathbf{M}$ é o número de variáveis. Definindo $\mathbf{S}$ como uma coligação do conjunto $\mathbf{F}$ ($\mathbf{S} \subseteq \mathbf{F}$), temos, por exemplo, as seguintes possibilidades de $\mathbf{S}$, quando $\mathbf{M}$ é igual a 3:

$$\{\{\emptyset\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Podemos definir também ν como uma função que vai mapear um conjunto de valores e retornar um número real. Com isso, o retorno de $\nu(\mathbf{S})$ é um número real que pode ser definido como o "trabalho da coligação $\mathbf{S}$ ". Esse valor é equivalente ao total ganho que os jogadores podem obter caso trabalhem juntos em uma determinada coligação.

Para calcular o ganho ao adicionar uma variável i ou a importância do jogador em específico, pode-se calcular o ganho quando é adicionada aquela variável na coligação menos a coligação sem a adição daquela variável, ficando da seguinte maneira:

$$\nu(\mathbf{S} \cup \{i\}) - \nu(\mathbf{S})$$

No exemplo acima, caso queiramos calcular o efeito da coligação $\{3\}$, poderíamos fazer:

$$\textit{Contribuição de } \{3\} = \nu(\{1, 2, 3\}) - \nu(\{1, 2\})$$

Mas suponha que as variáveis (ou jogadores) $\{2\}$ e $\{3\}$ sejam extremamente semelhantes. Quando é calculado o ganho após inserir $\{2\}$ na coligação $\{1,2\}$, é possível notar um aumento substancial, mas quando é adicionado $\{3\}$ na coligação $\{1,2,3\}$, o ganho

obtido é muito pouco. Como as variáveis exercem um papel parecido, o ganho maior ficou sujeito à variável que foi adicionada primeiro na coligação, não necessariamente porque uma é mais importante que a outra. Por isso, para calcular o real ganho da variável {i}, é necessário testar todas as permutações de **F** (conjunto de jogadores) e obter a contribuição de {i} em cada uma delas, para então fazer a média dessas contribuições. Por exemplo, definindo **F** = {1,2,3,4}, suponha que estamos interessados em calcular a contribuição de {3}, logo, podemos obter a seguinte permutação de **F**:

$$[3, 1, 2, 4]$$

Calculando a contribuição de {3}, temos:

$$\nu(\{3\}) - \nu(\emptyset)$$

Outra permutação poderia ser :

$$[2, 4, 3, 1]$$

Calculando a contribuição de {3}, nessa permutação temos:

$$\nu(\textit{coligação de } [2, 4, 3]) - \textit{coligação de } [2, 4])$$

Uma observação deve ser feita: a função ν considera a coligação como argumento, não a permutação. A coligação é um conjunto, com isso a ordem dos elementos não importa, mas a permutação é uma coleção ordenada de elementos. Na permutação do tipo [3,1,2,4], 3 é a primeira variável adicionada e 4 é a última. Por isso, para cada permutação a ordem dos elementos pode mudar a contribuição do total ganho, contudo o total ganho da permutação somente depende dos elementos, não da ordem. Logo:

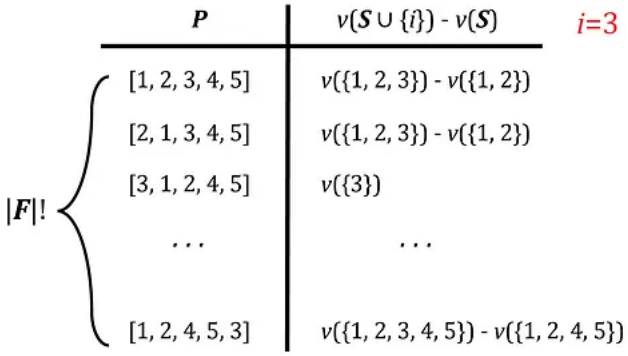
$$\nu(\textit{coligação de } [3, 1, 2, 4]) = \nu(\{1, 4, 2, 3\})$$

Sendo assim, para cada permutação **P**, é preciso primeiro calcular o ganho da coligação das variáveis que foram adicionadas antes de {i}, e esse conjunto pode ser chamado de coligação **S**. Feito isso, agora é preciso calcular o ganho das coligações que são formadas ao adicionar {i} em **S**, e podemos chamar isso de $\mathbf{S} \cup \{i\}$. Com isso, a contribuição da variável {i}, denotada por ϕ_i, é:

$$\phi_i = \frac{1}{|\mathbf{F}|!} \sum_{\mathbf{P}} [\nu(\mathbf{S} \cup \{i\}) - \nu(\mathbf{S})] \tag{1.3.1}$$

O número total de permutações de **F** é |**F**|!. Logo, podemos dividir a soma das contribuições por |**F**|! para encontrar o valor esperado de contribuição de {i}. A Figura **??** mostra como é feito esse calculo para um determinado jogador $\{i\}$.

Figura 7: Ganho do jogador 3 em relação à todas as permutações de jogadores.



$$\phi_i = \frac{1}{|F|!}\sum_{P}(v(S \cup \{i\}) - v(S))$$

Fonte: https://towardsdatascience.com/introduction-to-shap-values-and-their-application-in-machine-learning-8003718e6827

É possível perceber que algumas permutações possuem a mesma contribuição, desde que suas coligações $\mathbf{S} \cup \{i\}$ e **S** sejam as mesmas. Com isso, para reduzir o processo do cálculo de contribuição de cada permutação, pode-se identificar quantas vezes a permutação gerada vai resultar em uma contribuição que seja igual a outra.

Para fazer isso, é necessário descobrir quantas permutações podem ser formadas de cada coligação. Podemos definir $\mathbf{F} - \{i\}$ como o conjunto de todas as variáveis excluindo a variável {i}, e **S** como uma das coligações de $\mathbf{F} - \{i\}$ ($\mathbf{S} \subseteq \mathbf{F}$ - {i}).

Logo, para cada coligação **S** temos |**S**|! possíveis permutações, que corresponde às possibilidades de variáveis e suas respectivas ordens antes de adicionar a variável {i}.

Tendo os conjuntos $\mathbf{S} \cup \{i\}$ e **S** definidos, resta agora achar as possíveis permutações das variáveis restantes. E para saber o valor restante é preciso calcular o tamanho do conjunto gerado por: **F** - ($\mathbf{S} \cup \{i\} + 1$) que basicamente é o que resta das variáveis para completar o conjunto **F**.

A Figura **??** mostra o que acontece quando se escolhe o jogador i, nesse caso $i = 3$. Note que, na linha das coligações, é definido as possíveis coligações de **S**, que seria as permutações dos jogadores 1 e 2, temos a coligação de um único elemento na coluna $\{i\}$, que sempre vai ser o próprio elemento, em seguida a coligação dos jogadores restantes. Na linha das permutações é definida todas as possíveis permutações para **S**, para $\{i\}$ e

para $\mathbf{F} - \mathbf{S} - \{i\}$. E na última linha é representado o tamanho do conjunto formado pela permutação/coligação descrita anteriormente.

Figura 8: Relação entre permutações e coalizões.

Coalitions	***S***	+	$\{i\}$	+	***F*-*S*-$\{i\}$**	= ***F***
	$\{1, 2\}$	+	$\{3\}$	+	$\{4, 5\}$	= $\{1, 2, 3, 4, 5\}$
Permutations	$[1, 2]$ $[2, 1]$	+	$[3]$	+	$[4, 5]$ $[5, 4]$	= $[1, 2, 3, 4, 5]$ = $[1, 2, 3, 5, 4]$
					$[4, 5]$ $[5, 4]$	= $[2, 1, 3, 4, 5]$ = $[2, 1, 3, 5, 4]$
Number of Permutations	$\lvert S\rvert!$	+	1	+	$(\lvert F\rvert\text{-}\lvert S\rvert\text{-}1)!$	= $\lvert S\rvert!(\lvert F\rvert\text{-}\lvert S\rvert\text{-}1)!$

Fonte: https://towardsdatascience.com/introduction-to-shap-values-and-their-application-in-machine-learning-8003718e6827

Com isso, podemos reescrever a Equação **??** da seguinte maneira:

$$\phi_i = \sum_{\mathbf{S} \subseteq \mathbf{F} - \{i\}} \frac{|\mathbf{S}|!(|\mathbf{F}| - |\mathbf{S}| - 1)!}{|\mathbf{F}|!} [\nu(\mathbf{S} \cup \{i\}) - \nu(\mathbf{S})],$$

onde ϕ_i é o valor de shapley para a variável {i}.

1.3.2 Shapley Additive Explanations

Fazendo a relação do valor de Shapley para o SHAP (Shapley Additive Explanations), temos que a função característica ν é equivalente à função $f(x)$ responsável por fazer as predições. E os valores de SHAP são calculados a partir das observações que entram no modelo. Com isso, a fórmula do valor de SHAP, para cada conjunto de observação e variável especificada, se dá por:

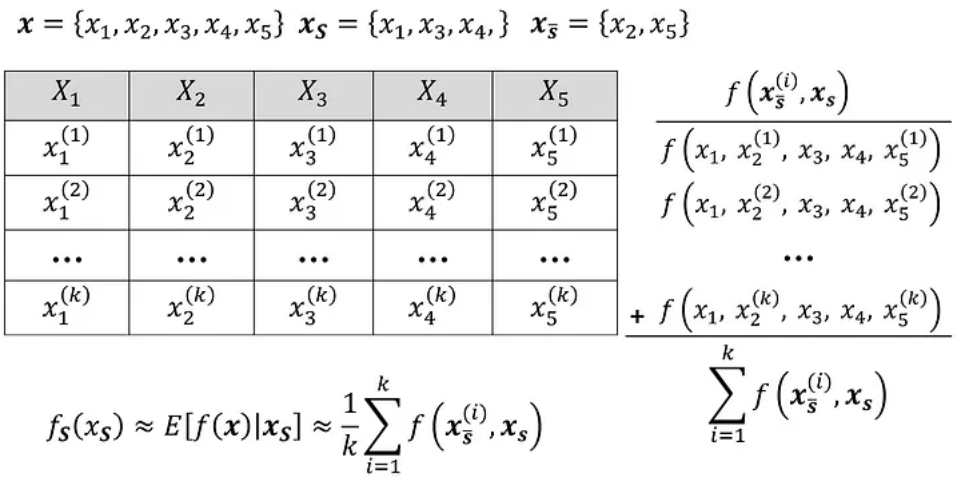
$$\phi_i(f, \mathbf{x}) = \sum_{\mathbf{S} \subseteq \mathbf{F} - \{i\}} \frac{|\mathbf{S}|!(|\mathbf{F}| - |\mathbf{S}| - 1)!}{|\mathbf{F}|!} [f_{\mathbf{S} \cup \{i\}}(\mathbf{x}_{\mathbf{S} \cup \{i\}}) - f_{\mathbf{S}}(\mathbf{x}_{\mathbf{S}})]$$

Perceba que $f_{\mathbf{S}}(\mathbf{x}_S)$ representa o resultado do modelo com somente as variáveis que estão na coligação $\mathbf{S}$, algo que na realidade não é permitido na maioria dos modelos.

Por isso, uma aproximação desse resultado é a seguinte:

$$f_S(\mathbf{x}_S)) \approx E[f(\mathbf{x}|\mathbf{x}_S)] \approx \frac{1}{k}\sum_{i=1}^{k} f(\mathbf{x}_{\bar{S}}^{(i)}, \mathbf{x}_S) \tag{1.3.2}$$

Figura 9: Cálculo de $f_{\mathbf{S}}$, sendo S o conjunto de variáveis X_1, X_3, X_4, dentre as observações de um conjunto de dados.



Fonte: https://towardsdatascience.com/introduction-to-shap-values-and-their-application-in-machine-learning-8003718e6827

A Figura **??** representa, para cada observação do conjunto de dados, a Equação **??**. Neste caso, as variáveis X_1, X_3 e X_4 representam o conjunto **S**, e escolhendo um valor x_1, x_3 e x_4 dessas variáveis, respectivamente, calcula-se f para cada observação do conjunto de dados, travando x_1, x_3 e x_4 na função e utilizando o valor das variáveis complementares, que neste caso são os valores de X_2 e X_5, em suas respectivas observações. Feito isso, é calculada média desses valores que corresponde justamente com o resultado da função $f_{\mathbf{S}}(x_{\mathbf{S}})$.

2 Conclusão

3 Anexo