**FFBUni**

**GRADUANDO EM CIENCIA DA COMPUTAÇÃO**

**JOSUE SOUZA LOPES**

**ERNESTO GURGEL VALENTE NETO**

**MATEUS DO CARMO DO VALE DE JESUS**

**JULIO OLIVEIRA**

**ANALISE DE ALGORITIMOS**

**ALGORITIMOS DE MACHINE LEARNING**

**FORTALEZA – CE**

**2020**

Josué Souza Lopes

Ernesto Gurgel Valente Neto

Mateus do Carmo do Vale de Jesus  
Julio Oliveira

**ANALISE DE ALGORITIMOS**

**ALGORITIMOS DE MACHINE LEARNING**

Esse artigo é apresentado para obtenção de nota V2 na matéria, analise de algoritmos. Centro Universidade Farias Brito – FFBUni.

Professor, Maikol Magalhaes Rodrigues

Fortaleza – CE

2020

**ANALISE DE ALGORITIMOS**

**ALGORITIMOS DE MACHINE LEARNING**

Josué Souza Lopes

Ernesto Gurgel Valente Neto

Mateus do Carmo do Vale de Jesus  
Julio Oliveira

**Resumo:** Este documento apresenta uma exposição de ideias sobre as analise de algoritmos e fundamentos a se considerar em sua avaliação e mensuração utilizando de técnicas, analises e provas aplicadas em *Machine Learning* e sobre a realização de uma comparação funcional e pragmática baseada em seus conceitos.

# SUMÁRIO

**1.0 – Introdução .................................................................................................**

**2.0 – Analise de Algoritimos.............................................................................**

**2.1 – Linguagem Python: Introdução...............................................................**

**2.2 – Linguagem R: Introdução.........................................................................**

**3.0 Machine Learning........................................................................................**

**3.1 Algoritimos....................................................................................................**

**Aprendizagem...........................................................................................  
Supervisionada.........................................................................................**

**Regressão Linear ....................................................................................**

**Naive Bayes..............................................................................................**

**Regressão Linear......................................................................................**

**4.0 Analise de Algoritmos de Machine Learning..............................................**

**5.0 Conclusão.......................................................................................................**

**Bibliografia ...........................................................................................................**

# 1.0 INTRODUÇÃO

Com o advento da popularidade em que os computadores se desenvolveram em meio a sociedade contemporânea e uma alta demanda a sistemas capazes de gerenciar operações complexas para a implementação de ambientes sofisticados de trabalho, assim também surgem uma grande necessidade de desenvolvimento de softwares.

A alta demanda de desenvolvimento gerou a necessidade de buscar-se mais metodologias e práticas que pudessem categorizar, analisar e mensurar valores que para agregar conhecimento e qualificar os códigos gerados nas várias linguagens emergentes que posteriormente surgiriam, destes conhecimentos que complementam os estados gerais são; fundamentos de programação, programação orientada objetos, modelos funcionais, paradigmas de linguagens, gestão de projetos, analise de algoritmos dentre tantos outros.

Segundo Dijkstra, “Computer science is no more about computers than astronomy is about telescopes”, tradução livre pelo autor: “A ciência da computação não é mais sobre computadores do que a astronomia é sobre telescópios ”. — (Edsger W. Dijkstra).

Tendo em vista a importância dada através desses conhecimentos esse documento tem como um objetivo apresentar conceitos e fundamentos dos para apresentar ideias e conceitos discutidos em sala de aulas, suas visões históricas, teóricas e práticas.

**2.0 ANALISE DE ALGORITIMOS**

A análise de algoritmos surgiu a partir das ideias de D.E. Knuth (The Art of Computer Programming), sendo Knuth o percursos do conceito ao qual seria dado como a previsão do tempo de execução e o consumo de espaço de um algoritmo.

Estudos sobre a análise de algoritmos foram ensaiados em livros de autores comumente reconhecidos pela comunidade computacional tais como Cormen, Leiserson, Rivest e Stein, Kleinberg e Tardos, de Brassard e Bratley. Ambos realizaram seus estudos e exprimiram ideias baseando-se em conceitos clássicos ao qual seriam utilizados para analisar desempenho e a correlação do mesmo. Sendo assim, a área de Análise de Algoritmos passa a tratar de importantes ferramentas matemáticas como a comparação assintótica de funções e a resolução de recorrências.

Análise de algoritmos estuda a correção e o desempenho de algoritmos. Em outras palavras, através de comparações matemáticas e resolução de funções de recorrência permite-se responder perguntas, primeiramente se o algoritmo é capaz de solucionar os problemas aos quais é especificado e em segundo quanto o algoritmo consome para processar uma 'entrada' de tamanho n.

S. S. Skiena, “Um bom algoritmo, mesmo rodando em uma máquina lenta, sempre acaba derrotando (para instâncias grandes do problema) um algoritmo pior rodando em uma máquina rápida. Sempre.” –”. —( S. S. Skiena, The Algorithm Design Manual).

Outro ponto importante na Análise de Algoritmos advém da complexidade do seu tempo de execução, ao qual se define por aproximação da previsão do tempo de execução, dito isto vale denotar que se encontram duas dificuldades:

1 – O tempo gasto depende dos dados de entrada.

2 – O tempo também depende da capacidade de hardware do computador utilizado.

Na análise da complexidade de algoritmos definimos os mesmos para os possíveis casos que são segregados em; o melhor cenário, ou melhor caso, o pior cenário, ou pior caso, e o caso médio para mensurar o custo computacional ao qual o algoritmo está tratando.

**Melhor caso:** é quando o algoritmo representa o menor tempo de execução sobre todas as entradas de tamanho n.

**Pior caso:** maior tempo de execução sobre todas as entradas de tamanho n.

**Caso médio (ou caso esperado):** é a média dos tempos de execução de todas as entradas de tamanho n.

Assim, como uma das disciplinas tratadas em ciência da computação percebemos que a análise de algoritmos se trata de definir o algoritmo mais eficiente em determinados problemas, permitindo criação de códigos mais saudáveis e eficientes.

Abaixo trataremos da linguagem de programação em Pyhton utilizando premissas e um conjunto de ideias discutidas no capítulo 1º do livro SEBESTA, R. W. Conceitos de Linguagens de Programação. 5. ed. Editora Bookman, 2013 (ou edição mais atualizada) para então descrever a linguagem usando os critérios desse capítulo.

**2.1 Linguagem Python: Introdução.**

Python é uma linguagem de programação, também ocasionalmente chamada em alguns meios de linguagem de uso geral, possuindo uma biblioteca padrão ao qual possui vários módulos para processamento de texto e expressões regulares e protocolos de rede (HTTP, FTP, SMTP, POP, XML-RPC, IMAP), acesso aos serviços do sistema operacional, criptografia, interface gráfica dentre outros.

Além da biblioteca padrão, existe uma extensão adicional para todo tipo de aplicação criada pela vasta comunidade. Python é tipicamente usado em aplicações web e como linguagem de scripting para administração de sistemas sendo uma linguagem também possui uma facilidade de integração com C faz de Python uma linguagem embutida atrativa em aplicações de maior porte.

Python é uma linguagem de propósito geral de alto nível e multiparadigma interpretada, script, de alto nível, imperativa, orientada a objetos, funcional com tipagem dinâmica.

• Interpretada: O código fonte é executado por um interpretador e conseguinte pelo sistema operacional ou processador.

• Linguagem Script: Significa que a mesa suporta scripts, programas que poderiam alternativamente ser executadas.

• Linguagem de Alto nível: Sendo dito que uma linguagem é de alto nível quando possui forte abstração.

• Linguagem Imperativa: É dito que uma linguagem é imperativa quando sua programação descreve a computação como ações indicando ações especificas a serem seguidas.

• Linguagem Orientada a Objetos: A mesma pode deter informações na forma de códigos, atributos, métodos e outros procedimentos.

• Linguagem Funcional: A mesma trata os dados na forma de funções e que evita a transição de estados ou em contraste da forma de aplicação imperativa.

• Linguagem de tripagem dinâmica: Não exigem declarações de tipos de dados, pois é permitido alterar o tipo de dado durante a sua execução.

**2.2 Linguagem R: Introdução.**

Desenvolvida por Ross e Robert R é uma linguagem de programação multi-paradigma, dinâmica, fracamente tipada, voltada à manipulação, análise e visualização de dados. R é um ambiente e uma linguagem de programação similar ao S, contudo, é uma implementação distinta do S. Por conta do contato dos desenvolvedores do ambiente R e linguagem R com a linguagem Scheme boa parte do interpretador da linguagem foi desenvolvido baseado nessas características permitindo até mesmo códigos executados em ambas serem compatíveis.

A primeiras versões do interpretador consistiam em códigos capazes de interpretar a linguagem Scheme e código em C ao qual permitia que a linguagem adquirisse novas funcionalidades ao qual seriam desenvolvidas em versões futuras focadas em análise, tratamento e manipulação de dados. Para tornar o interpretador útil eles precisavam adicionar estruturas de dados para suportar uso estatístico e escolher uma interface de usuário. Optaram por uma interface de linha de comando e como ambos estavam bem familiarizados com a sintaxe de S, foi natural usarem uma parecida.

Atualmente a linguagem R é amplamente aplicada por matemáticos, estatísticos e analistas devido ao seu escopo voltado para o mesmo fim, ganhando uma ampla gama de arquitetura que facilita a aplicação dos métodos e práticas de programação ao qual permitem prover um estilo mais límpido ao programar e focado diretamente nas análises estatísticas e matemáticas das operações que suportam o escopo ao qual foi desenvolvido.

**3.0 Machine Learning.**

Algoritmos de aprendizado de máquina compostos por fragmentos de códigos dos quais permitem os cientistas de dados e analistas de dados explorar, analisar, observar padrões complexos dos conjuntos e amostras populacionais de grandes montantes de informações e apresentar resultados significativos sobre essas explorações. Em Machine Learning cada algoritmo é considerado um conjunto finito de instruções inequívocas representadas por expressões matemáticas ao qual uma máquina utiliza para atingir um determinado objetivo. Assim, em Machine Learning um modelo de aprendizado de máquina, ou modelo de I.A, tem como o objetivo é estabelecer e descobrir padrões ao qual permita as pessoas categorizar informações ou realizar previsões sobre amostras.

Segundo Arthur Samuel, “ Aprendizado de máquina é o subcampo da ciência da computação que dá aos "computadores a capacidade de aprender sem serem explicitamente programados". — (Arthur Samuel).

Utilizando matemática e estatística para realizar a função necessária os algoritmos de Machine Learning tentam minimizar a perda de precisão nas previsões ao aplicar um determinado padrão e, então, nos devolvem o melhor padrão que poderiam aprender com os dados utilizando parâmetros baseados em dados de treinamento ou subconjuntos de dados populacionais para realizar treinamento ao qual expandem para representar o mundo de forma mais realista.

Os algoritmos de Machine Learning são agrupados pelas técnicas de aprendizado de máquina como trataremos a seguir dos mais comuns:

**1 – Aprendizagem supervisionada**

Os algoritmos fazem previsões sob conjunto de exemplos fornecidos.

**2 – Aprendizagem não supervisionada**

O algoritmo os rotula organizando os dados ou descrevendo estruturas.

**3 – Aprendizagem por reforço**

O algoritmo recebe feedback que o ajuda a determinar escolhas, considerando-as como correta, neutra ou incorreta. Normalmente usada em sistemas automatizados sem orientação humana.

Pontuando os algoritmos de aprendizado citados são mais comumente usados ​​usam regressão e classificação para prever categorias e grupos finais, encontrar pontos de dados incomuns, prever valores e descobrir semelhanças.

**3.1 Algoritmos.**

O interesse no aprendizado de máquina disparou nos anos desde que o artigo da Harvard Business Review intitulou 'Data Scientist' o 'trabalho mais sexy do século 21, assim surgiram em conseguinte vários modelos matemáticos que pudessem expressar a resolução desses, tais quais são citados a seguir;

Os conjuntos de algoritmos de aprendizado de máquina mais importantes para ciência de dados:

1 - Técnicas de aprendizagem supervisionada - Regressão Linear, Regressão Logística, CART, Naïve Bayes, KNN.

2 - Técnicas de aprendizagem não supervisionadas - a priori, K-médias, PCA.

3 - Técnicas de ensembling- Bagging with Random Forests, Boosting with XGBoost.

**1. Aprendizagem Supervisionada**

*O aprendizado supervisionado usa dados de treinamento rotulados para aprender a função de mapeamento que transforma as variáveis ​​de entrada (X) na variável de saída (Y). Em outras palavras, ele resolve f na seguinte equação:*

1. *Y = f (X)*

Isso nos permite gerar saídas com precisão ao recebermos novas entradas. Assim, dado um conjunto N de exemplos de treinamento da formula:

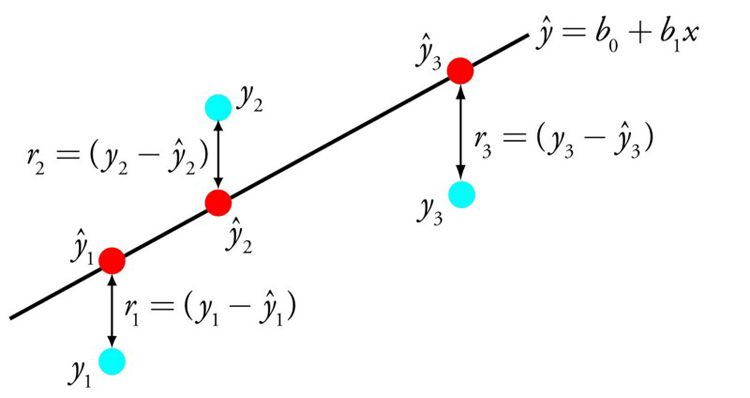
*{(x1,y1),....,(xN, yN)}* de tal modo que xi é o vetor do i-ésimo exemplo de *Yi*, e seu rotulo de aprendizagem seja uma função de busca de *g: X => Y*, onde *X* é o espaço de entrada e *Y* a saída. A função g é um elemento de espaços possíveis G, chamado de hipótese representada *por f: X \* Y => R* de tal modo que *G* é retornado como *Y* de maior pontuação: *g(x) = argmax(x,y)*.

Existem também formulas que são aplicáveis para reduzir o risco empírico e o risco estrutural do modelo supervisionado, além da formulação de Generative ao qual aplica métodos de treinamentos indiscriminados para encontrar funções G que discrimine possíveis valores de saída, porem ambas não serão tratadas nessa análise.

1. **2. Regressão Linear**

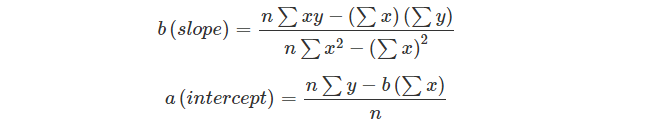
A regressão linear é talvez um dos algoritmos mais conhecidos e compreendidos em estatística e aprendizado de máquina existindo há mais de 200 anos e sendo amplamente estudada. Algumas boas regras básicas ao usar essa técnica são remover variáveis que são muito semelhantes (correlacionadas) e remover ruído de seus dados, se possível. É uma técnica rápida e simples e um bom algoritmo inicial para tentar.

A modelagem preditiva se preocupa principalmente em minimizar o erro de um modelo ou em fazer as previsões mais precisas possíveis, em detrimento da explicabilidade. A representação da regressão linear é uma equação que descreve uma linha que melhor se ajusta à relação entre as variáveis de entrada (x) e as variáveis de saída (y), encontrando ponderações específicas para as variáveis de entrada chamadas coeficientes (B).



**Figura 1: Regressão Linear**

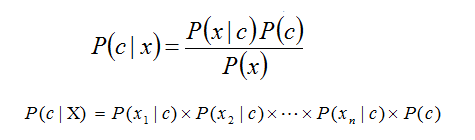
* + - * 1. Matematicamente, podemos escrever uma equação de regressão linear como:

1. what-is-linear-regression-2
   
2. Onde a e b são dados pelas fórmulas:
3. 
4. Tais quais; x e y são duas variáveis ​​na linha de regressão.
5. b = Inclinação da linha.
6. a = interceptação em y da linha.
7. x = variável independente do conjunto de dados
8. y = variável dependente do conjunto de dados

### **3. Naive Bayes**

Baseada no Teorema de Bayes com o pressuposto de independência entre preditores. Em termos simples, um classificador Naive Bayes assume que a presença de um recurso específico em uma classe não está relacionada à presença de qualquer outro recurso.

O teorema de Bayes fornece uma maneira de calcular a probabilidade posterior P (c | x) de P (c), P (x) e P (x | c). Observe a equação abaixo:



Assim temos que,

* *P* ( *c | x* ) é a probabilidade posterior da *classe*( *alvo* ) dado *preditor*( *atributo* ).
* *P* ( *c* ) é a probabilidade anterior da *classe*.
* *P* ( *x | c* ) é a probabilidade que é a probabilidade do *preditor*dada *classe*.
* *P* ( *x* ) é a probabilidade anterior do *preditor*.

### **4. Regressão Logística**

Na implementação da regressão logística, devemos estar cientes das seguintes suposições sobre o mesmo;

• no caso de regressão logística binária, as variáveis alvo devem ser sempre binárias e o resultado desejado é representado pelo fator nível 1.

• não deve haver nenhuma multicolinearidade no modelo, o que significa que as variáveis independentes devem ser independentes umas das outras.

• devemos incluir variáveis significativas em nosso modelo.

• devemos escolher um tamanho de amostra grande para a regressão logística.

A regressão logística é um algoritmo de classificação de aprendizagem supervisionada usado para prever a probabilidade de uma variável de destino. A natureza da variável alvo ou dependente é dicotômica, o que significa que haveria apenas duas classes possíveis.

Em palavras simples, a variável dependente é de natureza binária, com dados codificados como 1 (significa sucesso / sim) ou 0 (significa falha / não). Matematicamente, um modelo de regressão logística prevê P (Y = 1) como uma função de X. É um dos algoritmos de ML mais simples que pode ser usado para vários problemas de classificação, como detecção de spam, previsão de diabetes, detecção de câncer, etc.

A base da regressão logística é a função logística, também chamada de função sigmóide, que pega qualquer número de valor real e o mapeia para um valor entre 0 e 1, apresentada pela equação abaixo:

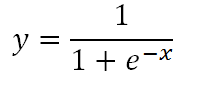
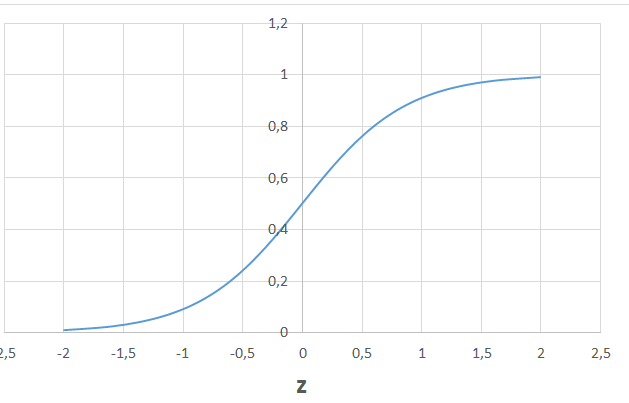


Image for post

Considerando que temos a seguinte equação linear e a reta representa a seguir:



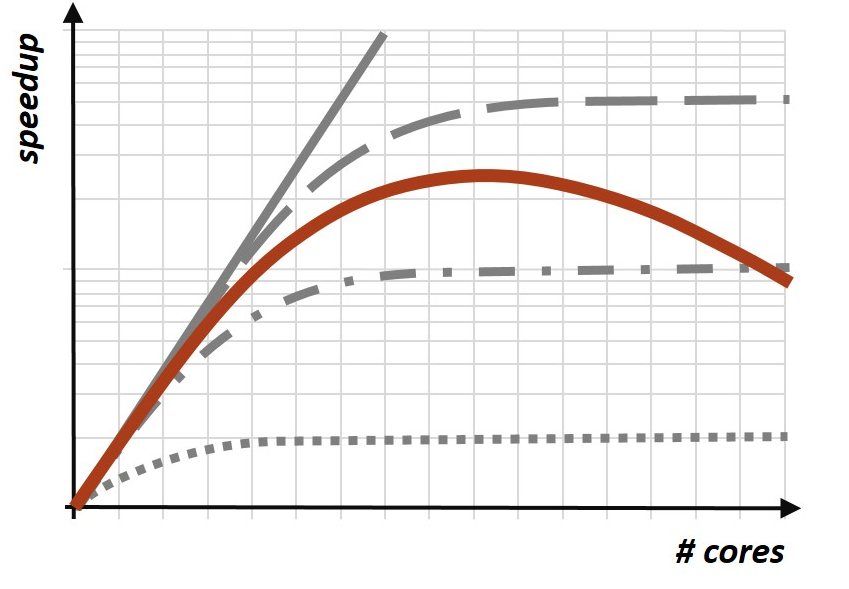
**Figura 2: Equação Linear Z**

O modelo de regressão logística é usado uma equação linear como entrada e uma função logística para medir as chances de log para realizar uma tarefa de classificação binária.

**4.0 ANALISE DE ALGORITIMOS DE MACHINE LEARNING**

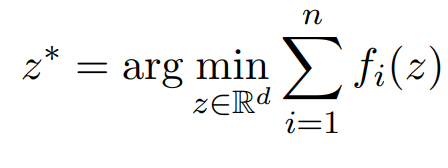
Segundos estudos da universidade de Boston, Departamento de Engenharia Elétrica e de Computadores e Divisão de Engenharia de Sistemas, “*os métodos de otimização de primeira ordem, que vão desde a descida do gradiente vanilla até a aceleração de Nesterov e suas variantes surgiram na última década como a principal forma de treinar modelos de aprendizado de máquina (Machine Learning). Assim passa a existir uma grande necessidade de técnicas para treinar tais modelos de forma rápida e confiável de forma distribuída em redes onde os processadores ou GPUs podem estar espalhados pelo mundo e se comunicar por meio de um rede que pode sofrer perdas de mensagens, atrasos e assincronia*”.

Ao trabalharmos com processamento em GPUs costuma haver aumento de processadores aos quais executando algoritmos costumeiramente são desperdiçados pelo custo de coordenação, compartilhado memória, perda de mensagens e latência. Esse fato costuma ocorrer a medida crescente aos quais os processadores encontram-se espalhados por data centers, fato amplamente reconhecido pela comunidade de sistemas distribuídos, pois é dito que independente de se aumentar consideravelmente o número de processadores em um problema não resultará, depois de certo ponto, em melhor desempenho, dito que também, a partir de um determinado ponto limite ao se acrescentar números adicionais de processadores paralelamente não irá efetivamente impactar positivamente depois de determinado ponto como expresso no gráfico a seguir;



**Figura 3: Taxa de decaimento de processamentos em GPU’s**

Essa representação é refletida devido aos limites de tempo de convergência obtidos para otimização distribuída na literatura. Dado então pela literatura acadêmica a formulação do problema:

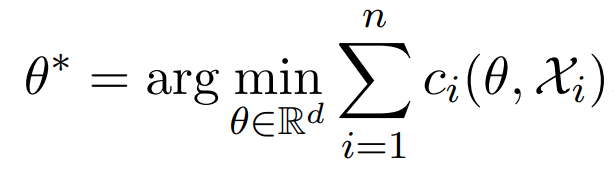


Dito que em uma rede de *n* nós apenas o nó *i* tem conhecimento de a função *fi (z)*, e a suposição padrão é que, a cada passo quando está acordado, o nó *i* pode calcular o gradiente de sua própria função local *fi(z)*.

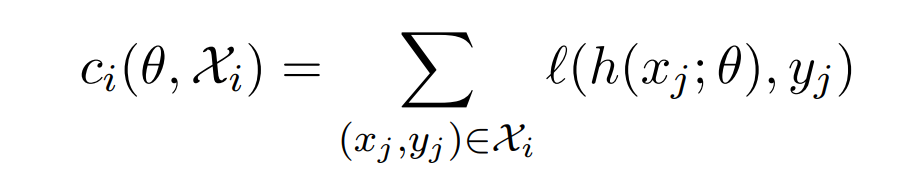
Representada pelas funções *fi(x), tais quais são assumidos como* convexas o problema se torna em calcular esses mínimos de forma distribuída pela rede com base na comunicação ponto a ponto, possíveis perdas de mensagens, atrasos e assincronia ao qual minimize os impactos.

Temos então que, supondo cada agente *i* irá armazenar pontos de dados de treinamento *Xi = {(xj, yj)}*, onde *xj ∈ R p* são vetores de características e *yj ∈ R* são as respostas associadas a variáveis discretas ou contínuas concluísse a necessidade de desenvolver um modelo preditivo *h (x; θ)*, parametrizado pelos parâmetros *θ ∈ R d*, de modo que *h (xj; θ) ≈ yj* para todos *j*.

Em outras palavras, estamos procurando um modelo que se adapte a todos os dados em toda rede ao qual deve ser minimização de risco empírico como expresso a seguir;

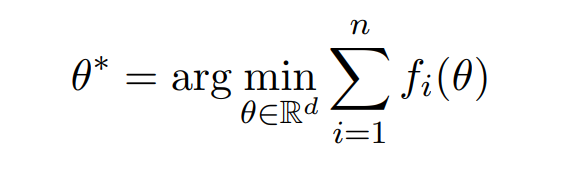


Onde temos que a equação mede o quão bem o parâmetro θ se ajusta aos dados em cada respectivo nó *i*, com *(h (xj; θ), yj)* representando a taxa de perda ao qual entre *h (xj; θ) e yj*.

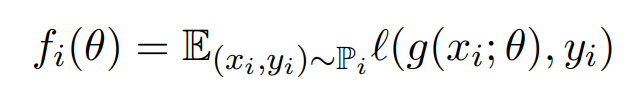


Muito do aprendizado de máquina moderno é construído em torno dessa formulação matemática ao qual é construída, incluindo regressão, classificação e variantes. Tornando possível para modelos dinâmicos aos quais agentes *i* permitem a coleta pontos de transmissão de dados *(xi , yi) ∼ Pi* repetidamente ao longo do tempo, onde *Pi* representa um distribuição desconhecida representada por *(xi , yi)*.

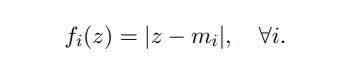
Neste caso, podemos encontrar θ∗ através da minimização de risco esperada, demonstra por;



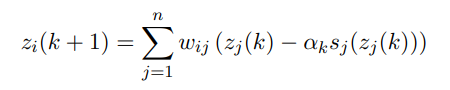
Onde temos que;



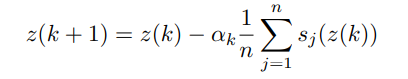
Em ciência da computação para ilustrar a preocupação com as limitações da distribuição e otimização dos modelos de Machine Learning são levados em consideração o método de subgradiente na configuração mais simples possível, ou seja, o problema de calcular o mediana de uma coleção de números de forma distribuída onde cada agente *i* na rede tem o valor *mi> 0,* e o objetivo global é encontrar a *mediana de (m1, m2, ..., mn)*, torna-se um problema expresso que pode ser incorporado na estrutura escolhendo o modelo a seguir;



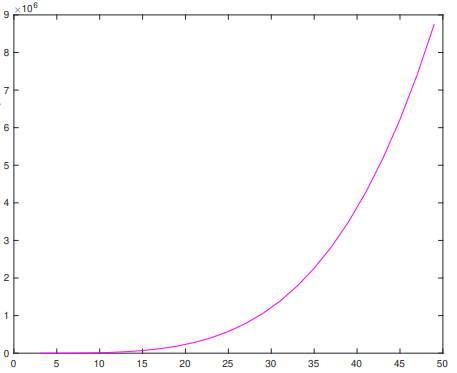
O método de subgradiente usa os subgradientes *si(z) de fi(z)* em qualquer ponto *z*, para que o agente atualize como;



Onde *αk> 0* denota o tamanho do passo na iteração *k, e wij ∈ [0, 1]* são os pesos do agente *3i* aos quais é atribuída as soluções do agente *j:* dois agentes *i* e *j* são capazes de trocar informações se e apenas se *wij, wji> 0 (wij = wji = 0*). Os pesos *wij* são assumidos como sendo simétrico onde, o método de subgradiente centralizado atualiza a solução em iteração *k* de acordo com



Conclui-se que o desempenho do Algoritmo em função do tamanho da rede n aos quais os agentes comunicam-se através de uma rede em anel para tamanhos de passo *αk = √1/k*, e *mi* são uniformemente distribuídos em *[−10, 10] e o* tempo *k* chegar a *1 n Pni = 1 | yi (k) |* <é plotado em gráfico, onde *yi (k) = 1 k Pk – 1 `= 0 zi (`) e = 0,1*.



**Figura 4: Reta do Gráfico Plotado N**

Claramente, esta é uma propriedade indesejável. Olhando para a figura, vemos que a distribuição computação em 50 nós pode resultar em um tempo de convergência de ordem consideravelmente elevado. Poucos ficariam entusiasmados com a otimização distribuída se o efeito final for muito aumento do tempo de convergência.

Notasse que este fenômeno demonstrado para o problema de computação não será válido para os problemas de otimização mais sofisticados na literatura Machine Learning (ML). Infelizmente, a maioria dos trabalhos de otimização de algoritmos em sistemas distribuídos replica isso fenômeno indesejável.

**5.0 CONCLUSÃO**

Por mais essa nova tecnologia apresente características de um sistema notável com uma infraestrutura robusta ainda apresenta também muitos desafios, sendo a ciência de dados uma área nova em desenvolvimento, porém devido a muitos das premissas matemáticas usadas em probabilidade e estática serem discutidas amplamente a décadas ou séculos muitos destes modelos anteriormente apresentados na resolução desse documento são amplamente aceitos pela comunidade matemática e acadêmica.

Nesse contexto a análise de algoritmos apresentada como disciplina permite medir o custo computacional e analisar com cautela e destacar todas as peculiaridades em modelos mais complexos de experimentos para que possam surgir metodologias e algoritmos capazes de melhor utilizar da arquitetura de hardware utilizada em processamentos matriciais destas respectivas operações.

### **6.0 REFERÊNCIAS**

Analyticsvidhya, https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

Data do Acesso 20 de Novembro de 2020.

Eureka, https://www.edureka.co/blog/machine-learning-algorithms/

Data do Acesso 20 de Novembro de 2020.

Towardsdatascience, https://towardsdatascience.com/11-most-common-machine-learning-algorithms-explained-in-a-nutshell-cc6e98df93be

Data do Acesso 20 de Novembro de 2020.

Medium, https://medium.com/swlh/63-machine-learning-algorithms-introduction-5e8ea4129644

Data do Acesso 21 de Novembro de 2020.

Simplilearn, https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article

Data do Acesso 21 de Novembro de 2020.

Medium, https://medium.com/analytics-steps/top-10-machine-learning-algorithms-77704f259638

Data do Acesso 21 de Novembro de 2020.

Kdnuggets, https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html

Data do Acesso 22 de Novembro de 2020.

Dataquest, https://www.dataquest.io/blog/top-10-machine-learning-algorithms-for-beginners/

Data do Acesso 22 de Novembro de 2020.

Machinelearningmastery, https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Data do Acesso 21 de Novembro de 2020.

Builtin, https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies

Data do Acesso 23 de Novembro de 2020.

Ufabc, http://professor.ufabc.edu.br/~jair.donadelli/aa/index.html

Data do Acesso 23 de Novembro de 2020.

Treinaweb, https://www.treinaweb.com.br/blog/introducao-a-analise-de-algoritmos/

Data do Acesso 23 de Novembro de 2020.

IC, https://www.ic.unicamp.br/~lee/mc448/

Data do Acesso 21 de Novembro de 2020.

Azure, https://azure.microsoft.com/en-us/overview/machine-learning-algorithms/

Data do Acesso 20 de Novembro de 2020.

Wikipedia, https://pt.wikipedia.org/wiki/An%C3%A1lise\_de\_algoritmos

Data do Acesso 20 de Novembro de 2020.

IME, https://www.ime.usp.br/~pf/analise\_de\_algoritmos/lectures.html

Data do Acesso 20 de Novembro de 2020.

w3schools, https://www.w3schools.com/python/python\_try\_except.asp

Data do Acesso 21 de Novembro de 2020.

wikipedia, https://en.wikipedia.org/wiki/Supervised\_learning

Data do Acesso 23 de Novembro de 2020.

Universiadade de Boston, <https://www.researchgate.net/profile/Shi_Pu6/publication/334129675_Asymptotic_Network_Independence_in_Distributed_Optimization_for_Machine_Learning/links/5d1c1eeea6fdcc2462bad8b7/Asymptotic-Network-Independence-in-Distributed-Optimization-for-Machine-Learning.pdf>

Data do Acesso 24 de Novembro de 2020.