Iniciação Cientifica

Machine Learning e Quantum Computing

Lucas A. Leite[a,1]

[a]*Grupo SAMPA - Instituto de Física da USP*

Professor/Grupo SAMPA

Abstract—Escrever alguma coisa como abstract desse relatório e estudo.

Keywords—*Machine Learning, Quantum Computing, Artigo*

1. Introdução

Vamos começar nosso estudo embarcando em Machine Learning buscando um overview de técnicas e com uma quantidade balanceada de teoria de Statistical Learning e uma abordagem um pouco mais matemática, pretendo também deixar citações de onde as discussões e teoria estão sendo tiradas e podem ser aprofundadas ou lidas na integra, ao mesmo tempo, vou buscar trazer códigos com aplicações dos conceitos estudados.

Vamos começar com o estudo de Machine Learning clássico mas sempre que possível buscar conceitos de Quantum Computing para podermos trabalhar com essa infraestrutura no futuro.

2. Inicio

De forma bem vaga, definirei Machine Learning (ML) como um método de aproximação de uma função preditora do comportamento de um sistema especifico, que é treinada com dados relativos a tal sistema. Eu vou supor que o leitor já tenha uma certa familiaridade com a finalidade do estudo de ML.

Vou passar, então, desse tópico de apresentação e motivação do estudo, direto para trabalhar com as principais estruturas de ML que temos: Regressão, Classificação e Clusterização. Trabalharemos com Redes Neurais também.

3. Regressão Linear

3.1. Modelo

Costumeiramente o tópico e o framework de ML é introduzido a partir de Classificação e exemplificações. com kNN, porém não farei desta foram e vou introduzir o conteúdo de uma forma um pouco mais crua e direta, pulando motivações, como já foi dito.

Sendo assim, vamos direto ao assunto. Quando estamos trabalhando com ML a ideia e a construção de uma função que sera otimizada e que em Regressão Linear (RL) é descrita como:

$$f(\mathbf{X}) = \hat{\mathbf{Y}} = W^T\mathbf{X} \approx Y$$

Veja, nos temos um conjunto de dados $\mathbb{D} = \{(\mathbf{X}, \mathbf{Y})\}$ no qual $\mathbf{X}$ são nossos $inputs$ e $\mathbf{Y}$ que são nossos $outputs$, ambos dados reais do nosso sistema/problema, outras nomenclaturas são usadas na literatura como (feature,label), (preditor, predição). Nossos grandes Romeu e Julieta são o foco principal quando tratamos de aprendizado supervisionado, que é quando temos justamente o *output* para podermos ter um parâmetro de comparação de erro ou acerto. É valido mencionar que o nome linear é sobre a linearidade nos parâmetros W e não nos inputs, onde podemos fazer transformações.

Bem, voltando a nossa função definida acima, ela pode ser chamada de ***Função de predição*** **($\hat{Y}$)**, veja bem $\hat{\mathbf{Y}}$ e $\mathbf{Y}$ são diferentes, uma é a nossa função predição, que vem da relação linear descrita, o outro é o valor de output real que temos nos dados, respectivamente.

3.2. Otimização

Regessão Linear vem com um chute inicial para a função preditora que depois sera otimizada para que cheguemos num valor de coeficientes $\mathbf{W}$ que melhor descrevam o comportamento dos dados que temos.

Assim, com esse conceito de comparação, vamos criar uma função de perda, chamaremos de Loss function ou Loss. O conceito geral é que essa função deve medir a diferença entre os dados preditos pelo nosso modelo e os valores reais que temos.

Um dos exemplo mais comuns de Loss function são os *Resíduos Quadrados* que tem uma forma de:

$$l = (y - \hat{y})^2$$

Porém podemos considerar o conjunto $\mathbb{D}$ e somar o Loss considerando todos os dados que possuímos, isso nós chamaremos de Cost Function ou Função de custo, a otimização do processo esta realmente nesta função.

$$J = (Y - \hat{Y})^2$$

De forma inocente vamos assumir que o melhor modelos que podemos ter é aquele com que cometeremos o menor número de erro, dessa forma minimizando a função de custo J. O unico parâmetro que podemos levar em conta para a minimização da função de custo é o W, dessa forma vamos derivar com respeito a ele:

$$\begin{aligned}\frac{\partial J}{\partial W} &= \frac{\partial}{\partial W}(Y - WX)(Y - WX)^T \\ &= \frac{\partial}{\partial W}[Y^2 + WXX^TW^T - YX^TW^T - WXY^T] \\ &= XX^TW^T + WXX^T - YX^T - XY^T = 2(WXX^T - YX^T) \\ &= 0\end{aligned}$$

Assim podemos achar o valor de W:

$$W = \frac{YX^T}{XX^T}$$

Porém esse valor só sera possível de obter dessa forma, se e somente se, XX^T for inversível.

3.3. Gradiente Descendente

Isso nem sempre é o que acontece, sendo assim, vamos criar uma técnica que é mais geral, chamamos essa técnica de **Gradiente Descendente**, e ele funciona dessa forma:

1º Teremos um chute inicial de parâmetros W_0 e a partir desse chute inicial iremos buscar convergir para o mínimo em pequenos passos e atualizações, assim a cada interação o W atualizado sera o seguinte:

$$W_{n+1} = W_n - \eta\nabla_W J$$

Onde o η é o que chamamos e step size, porém essa é uma denominação confusa, visto que esse hiperparâmetro seria mais como um step weight. O termo do gradiente é essencialmente o que fizemos anteriormente para o caso de XX^T inversível. Bem e qual é o peso ideal de cada passo de otimização? Isso é algo bem nebuloso, a única informação relevante referente a essa questão é que esse hiperparâmetro não deve ser muito grande, porém também não muito pequeno, sweet spot geralmente é $\eta = 0,01$, porém ´ um chute. Outra possibilidade quando tratando desse problema é utilizar um step size ajustável a

alguma premissa que você queira, por exemplo, se o gradiente for muito grande, quero que η seja grande, ou pequeno, premissas desse tipo.

Bem, agora pode restar a dúvida, quando devemos parar de otimizar o nosso parâmetro W? Em geral as principais formas usadas são número de interações ou ganho negligenciável de otimização, dando um retorno para que o sistema para de otimizar o parâmetro.

3.4. Gradiente Descendente Estocástico

Bem, vimos sobre alguns aspectos do GD, porém não discutimos sobre um caso que pode ser limitante para o uso dele... A quantidade de dados!

Caso tenhamos uma quantidade muito grande de dados calcular o GD será uma tarefa muito cara computacionalmente e levará muito tempo. Sendo

3.5. Verossimilhança

Até agora, trabalhamos com a ideia de Machine Learning como uma função a ser melhorada para melhor descrever os dados. Vamos agora discutir um contexto um pouco mais geral de Machine Learning, vamos tratar ele de forma probabilística. Primeiro, vamos tomar o nosso valor de Y do nosso dataset $\mathbb{D}$, e vamos descreve-lo segundo a seguinte equação:

$$Y = W^T X + \epsilon$$

Onde o ϵ seria um erro gaussiano normal, com média 0.

Podemos pensar, como exemplo ilustrativo dessa nova filosofia, que o nosso conjunto de dados é inicialmente produzido segundo o modelo $W^T X$ e depois é adicionado um ruido nesse valor obtido, e esse processo se repete para todos os pontos, cada um com um erro gaussiano próprio. O nosso termo de erro serve para justificar o erro que não conseguimos controlar e os erros aos quais decidimos não abordar na complexidade do modelo.

Por fim, temos a função de probabilidade para os dados, onde a premissa são os parâmetros usados:

$$p(\mathbb{D}|W^T) = \mathcal{L}(W^T|\mathbb{D})$$

que nos informa quão provável é que os dados sejam explicados pelos parâmetros W^T que decidimos usar ($p(\mathbb{D}|W^T)$), ou numa medida inversa qual a probabilidade de W^T ao explicar os dados, $\mathcal{L}(W^T|\mathbb{D})$.

No nosso caso em especial, estamos considerando o erro inerente como gaussiano, logo

$$p(\mathbb{D}|W^T) = \mathcal{L}(W^T|\mathbb{D}) = \mathcal{N}(Y|W^T X, \sigma^2)$$

Além disso, sabemos que maximizar $\mathcal{L}(W^T|\mathbb{D})$ chegara no mesmo resultado que maximizar $log\mathcal{L}(W^T|\mathbb{D})$, então vamos trabalhar com o log, por ser algebricamente mais facil. Teremos algo como:

$$log\mathcal{L}(W^T|\mathbb{D}) = \frac{N}{2\sigma^2}\frac{1}{N}(Y - W^T X)^2 + cte$$

Se prestarmos atenção notaremos que temos uma expressão muito familiar a de Mínimos Quadrados. Uma propriedade do erro quadrático é que ele pune muito mais os valores que estão muito longe do fit suposto, logo ele é muito mais sensível para outliers. Além disso, o log da probabilidade do modelo é concavo e mais faceis de achar os melhores valores de fitting. Para este modelo especifico que estamos usando minimizar os erros quadráticos é a mesma coisa que maximizar a probabilidade com erro gaussiano, porém isso pode não ser verdade no caso de modelos que os erros não são Gaussianos.

3.6. Métricas de análise

Como podemos saber qual a incerteza que temos sobre certos parâmetros que estamos considerando no nosso modelo. Aqui nós começamos a discutir como melhor selecionar um modelo e como saber se nosso modelo é adequado como solução. Para isso existe duas filosofias para abordar esta tarefa.

3.6.1. Goodness of Fit

Aqui usamos parâmetros que levam em consideração

3.6.2. Cross-Validation

3.7. Generalized Linear Model (GLM)

4. Tratamento dos dados e divisão

Este capitulo é totalmente dedicado à praticas que são essenciais quando tratamos do conjunto de dados que possuímos. Bem quando temos um conjunto de dados do tipo:

$$\mathbb{D} = \{(\mathbf{X}, \mathbf{Y})\}$$

E queremos usar ele para treinar um modelo de descrição do nosso problema, nós utilizamos técnicas para o fitting dos dados segundo nossa função de predição. Porém, isso nos gera a dúvida sobre 'o que garante que o nosso fit esta realmente certo?', 'será que se eu treinar meu modelo para ele cometer o mínimo de erros possíveis é a melhor solução?'. Para responder a primeira pergunta e acabando respondendo a segunda também, podemos fazer um teste, quebramos nosso conjunto de dado $\mathbb{D}$ em dois pedaços, um $\mathbb{D}_{treino}$ e o outro $\mathbb{D}_{teste}$ segunda uma proporção digamos 80% e 20% respectivamente. Podemos então treinar o modelo com apenas os 80% e depois testar nosso modelo para predizer os dados no conjunto de 20%. Dessa forma, podemos ter um feedback de como o modelo vai atuar quando ele tiver diante de dados reais, não antes vistos.

Essa prática é o que chamamos de divisão Treino/Teste, e o modelo deve ser usado no conjunto de dados de teste apenas quando tiver fundamentalmente pronto, e é preciso ter cuidado para que não exista *vazamento* das informações de resposta para dentro do dataframe de treino do modelo, além do cuidado de que os dados sejam selecionados aleatoriamente.

Além desta técnica comentada, existem diversas outras, falarei em seguida de algumas outras:

4.1. Treino/Validação/Teste

Essa quebra é muito similar a que comentamos agora acima, porém invés de quebrar apenas em dois subconjuntos, quebraremos em três. O de **Treino**, que é usado para treinar o modelo, o de **validação** que é usado para analisar as métricas e para fazer mudança no modelo segundo os resultados obtidos nessas métricas e, por fim, o subconjunto de **Teste**, que tem o mesmo efeito da outra quebra explicada acima, que é usada apenas para testar o modelo num ambiente similar a realidade onde performará o modelo.

4.2. K-Fold

Essa abordagem é um pouco diferente das ultimas duas comentadas, aqui nós iremos pegar nosso dataset $\mathbb{D}$ e quebra-lo em K subdatasets de mesmo tamanho, e faremos um revesamento onde pegamos um desses K subconjuntos e chamaremos de subconjunto de Teste e os demais serão os subconjuntos de Treino. Faremos isso até que todos os subconjuntos tenham sido utilizados como conjuntos de Teste e tivermos consequentemente gerado K modelos. Por fim, tomamos a média desses K modelos gerados como o modelo que vai descrever o comportamento do problema que temos.

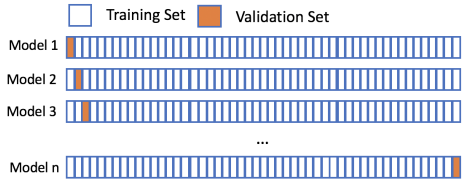

Figure 1. Esboço da estratégia de K-fold e Leave One Out.

4.3. Leave One Out

Na mesma linha em que pensamos na estratégia do K-fold, podemos levar ao limite, onde invés de quebrarmos nosso data set $\mathbb{D}$ em K partes, vamos levar ao máximo possível e tomar cada um dos dados, Deixando um dos dados para Teste e os demais como Treino. Faremos depois o mesmo processo que fizemos no K-Fold, treinaremos diversos modelos e pegaremos a média deles.

Logo num conjunto com N dados, treinaremos N modelos e tomaremos a média deles.

5. Classificação

6. Rede Neurais

O que são rede neurais? De forma geral e abstrata são um conjunto de operações matemáticas aplicadas em um conjunto de inputs de forma a gerar um output. Essas operações matemáticas são mutáveis e escolhidas de acordo com o que você quer estruturar, os inputs são tratados adicionando a eles um fator de peso, de forma que um vetor de input pode ter mais impacto na sua segunda ou primeira entrada, etc. Além disso, esse conjunto de pesos usados podem e devem ser otimizados através de um sistema de aprendizado supervisionado para que possam mais se aproximar do sistema do problema em questão. Essa explicação é uma síntese do que seria uma rede neural, sem levar em conta sua diferentes formas. Porém, vamos entrar em mais detalhes sobre essa ferramenta agora.

Dentro de Redes Neurais, possuímos um vetor de input x, um conjunto de pesos w e um conjunto de valores de Intercepto/Bias/Viés b, todos esses vetores são importantes para alimentar um perceptron, que é uma única unidade da rede neural. Vamos explicar como o perceptron funciona e generalizar para redes maiores.

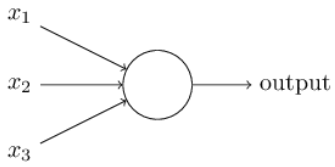

Figure 2. Esboço de como é um sistema de perceptron.

7. Equation

Equation 1, shows the Schrödinger equation as an example.

$$\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi = -i\hbar\frac{\partial\Psi}{\partial t} \tag{1}$$

The *amssymb* package was not necessary to include, because stix2 font incorporates mathematical symbols for writing quality equations. In case you choose another font, uncomment this package in tau-class/tau.cls/math packages.

If you want to change the values that adjust the spacing above and below the equations, play with `\setlength{\eqskip}{8pt}` value until the preferred spacing is set.

8. Adding codes

This class[1] includes the *listings* package, which offers customized features for adding codes in LaTeX documents specifically for C, C++, LaTeX and Matlab.

You can customize the format in tau-class/tau.cls/listings style.

```
function fibonacci_sequence(num_terms)
    % Initialize the first two terms of the sequence
    fib_sequence = [0, 1];

    if num_terms < 1
        disp('Number of terms should be greater than or equal to 1.');
        return;
    elseif num_terms == 1
        fprintf('Fibonacci Sequence:\n%d\n', fib_sequence(1));
        return;
    elseif num_terms == 2
        fprintf('Fibonacci Sequence:\n%d\n%d\n', fib_sequence(1), fib_sequence(2));
        return;
    end

    % Calculate and display the Fibonacci sequence
    for i = 3:num_terms
        fib_sequence(i) = fib_sequence(i-1) + fib_sequence(i-2);
    end

    fprintf('Fibonacci Sequence:\n');
    disp(fib_sequence);
end
```

Code 1. Example of Matlab code.

If line numbering is defined at the beginning of the document, I recommend placing the command `\nolinenumbers` at the start and `\linenumbers` at the end of the code.

This will temporarily remove line numbering and the code will look better.

9. References

The default formatting for references follows the IEEE style. You can modify the style of your references. See appendix for more information.

10. Appendix

10.1. Alternative title

You can make the following modification in tau-class/tau.cls/title preferences section to change the position of the title.

```
\newcommand{\titlepos}{\centering}
```

Code 2. Alternative title.

This will move the title to the center.

10.2. Info environment

An example of the info environment declared in the 'tauenvs.sty' package is shown below. Remember that *info* and *note* are the only packages that translate their title (English or Spanish).

Information

Small example of info environment.

10.3. Equation skip value

With the `\eqskip` command you can change the spacing for equations. The default *eqskip* value is 8pt.

```
\newlength{\eqskip}\setlength{\eqskip}{8pt}
  \expandafter\def\expandafter\normalsize\expandafter{%
    \normalsize%
    \setlength\abovedisplayskip{\eqskip}%
    \setlength\belowdisplayskip{\eqskip}%
    \setlength\abovedisplayshortskip{\eqskip-\
     ↪ baselineskip}%
    \setlength\belowdisplayshortskip{\eqskip}%
  }
```

Code 3. Equation skip code.

[1] Hello there! I am a footnote :)