Fundamentos de Sistemas Inteligentes
Projeto 1: Experimentos de classificação com dois algoritmos

Gabriel Correia de Vasconcelos
Departamento de Ciência da Computação
Universidade de Brasília
Campus Universitário Darcy Ribeiro, DF
gcvasconcelos98@gmail.com

Raphael Luís Souza de Queiroz
Departamento de Ciência da Computação
Universidade de Brasília
Campus Universitário Darcy Ribeiro, DF
rapha95@gmail.com

***Resumo*—Primeiro projeto da matéria de Fundamentos de Sistemas Inteligentes com o objetivo de fazer experimentos de classificação com dois algoritmos: um linear, o LDA (Linear Discriminant Analysis) e outro não-linear, o Knn (K Nearest Neighbors). O projeto utiliza como base de dados, um dataset do MNIST onde há o registro de 70.000 dígitos manuscritos em imagens em escala de cinza. Para a implementação dos experimentos foi utilizada a linguagem Python.**

***Index Terms*—classification, python, lda, knn, mnist**

I. INTRODUÇÃO

O projeto consiste numa série de experimentos utilizando dois algoritmos de classificação. Um é O LDA, um classificador linear normalmente usado para redução de dimensões na etapa de pre-processamento de aplicações de reconhecimento de padrões supervisionadas. O outro é o Knn, classificador não paramétrico, considerado um dos mais simples entre os algoritmos de aprendizagem e se baseia na similaridade de características.

Os dados a serem usados como testes pertencem ao National Institute of Standards and Technology e correspondem a uma base de dados de 70.000 dígitos manuscritos, em forma de imagens em escala de cinza 28x28, que variam de 0 a 9. No experimento 60.000 destas imagens serão usadas para o treino dos algoritmos e 10.000 para o teste da acurácia destes. O objetivo dos algoritmos é classificar, dada um imagem, o digito ela representa de forma correta.

Para a realização dos experimentos a equipe escolheu a linguagem Python devido a sua conhecida facilidade para manipular com dados e sua ampla disponibilidade de bibliotecas para nos auxiliar na aplicação dos algoritmos. As principais bibliotecas usadas foram: numpy, para a utilização de seu array e algumas funções matemáticas; tensorflow, para o download do MNIST dataset; matplotlib, para a visualização de alguns gráficos relacionados ao experimento; sklearn, para utilizar seus algoritmos de aprendizado de máquina; entre outras, para usos mais pontuais no código.

Apesar das bibliotecas possuírem funções fazerem a maior parte do trabalho de classificação, é extremamente necessário o conhecimento dos conceitos teóricos que rodam por trás do algoritmo, para entender o que são os argumentos da função e o que esperar de seu retorno.

II. CONCEITOS TEÓRICOS

A. Classificação e classificadores

O problema da classificação consiste na identificação de qual grupo pré-definido uma observação pertence, baseado na similaridade de seus critérios com os do grupo. Esses critérios são um conjunto de propriedades quantitativas pertencentes a cada observação e podem ser categóricos (e.g. ”vermelho”, ”amarelo”, ”lilás”), ordinais (e.g ”ruim”, ”médio”, ”bom”) ou números, sejam eles valores inteiros ou reais.

Outro nome para a classificação é reconhecimento de padrões, pois o classificador (F(x)) é função matemática que diz a qual grupo pertence a observação (y) baseado em padrões identificados em seus critérios (x), que são organizados em grupos. Nesse modelo, o Y é sempre discreto.

$$y = F(x) \tag{1}$$

A classificação ainda pode ser divida entre binária ou multi-classe. Na binária, a observação só possui duas classificações, ou é ou não é, enquanto na multi-classe ela pode ter várias categorias.

Na linguagem de aprendizado de máquina, algoritmos de classificação necessitam de um aprendizado supervisionado, ou seja, é necessária a existência de um conjunto de dados de treino, com os valores dos critérios e da classificação da observação já conhecidos, para a predição da classificação de novos dados.

Por convenção, sempre uma pequena parte da base de dados, chamada de dados de teste, é separada para o teste da acurácia do classificador escolhido. Outra forma de visualizar a corretude do modelo estatístico escolhido e ainda mais identificar mais facilmente falsos positivos e a distribuição das predições é a utilização de matrizes de confusão. Esse é uma matriz de valores reais e valores preditos onde a diagonal corresponde a predições acertadas e as outras posições foram falsos positivos, classificações que o classificador achou que

estavam certas, mas ao comparar com o Y do dado de treino elas se mostraram erradas.

Análise de crédito em bancos dado os dados do cliente e proposta de empréstimo. Diagnóstico de doenças dado os sintomas e exames do paciente. Reconhecimento de faces, dado diversas fotos digitais da pessoa. Todos esses são problemas muito conhecidos no mundo do aprendizado de máquina e que suas soluções são algoritmos de classificação.

B. LDA

O LDA (Linear Discriminant Analysis) é um método usado para descobrir combinações lineares dos critérios que classificam duas ou mais observações diferentes. Ele é linear pois envolve a determinação de uma equação linear, como por exemplo uma regressão linear, que terá a função de estimar a que grupo um nova observação pertence, e é discriminante pois utiliza funções discriminantes que são as que fazer as combinações lineares.

$$D = v_1X_1 + v_2X_2 + ... + v_iX_i + a \tag{2}$$

Na equação 2], D é a função discriminante, v são os critérios da observação, X é a observação e a é uma constante arbitrária.

A discriminação possui algumas regras e por isso assume que as observações são de amostras randômicas e que estas estão normalmente distribuídas, com uma variância comum. Como a média e variância são estimados, os critérios são escolhidos de forma a maximizar a densidade da população e a distancia entre as médias dos grupos. Normalmente, bons preditores tem os pesos da função polinomial alto.

A particularidade do LDA é que ele modela a distribuição dos preditores separadamente em cada grupo e então usa o teorema de Bayes [3] para estimar a probabilidade. Supondo que queremos classificar um observação em K grupos. Assim, π_k é a probabilidade de um observação ser de kaésimo grupo. Então $f_k(x)$ representa a função de densidade de X para uma observação do kaésimo grupo. Isso significa que quanto maior a probabilidade de uma observação ser do kaésimo grupo, maior o valor da função de densidade.

$$P(X) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{3}$$

O funcionamento do LDA pode ser resumido em cinco passos principais:

1) Computa os vetores de média dos grupos do dataset.
2) Computa as matrizes de dispersão, dentro e entre os grupos
3) Computa os autovetores e autovalores correspondentes das matrizes de dispersão
4) Escolhe n autovetores com os maiores autovalores para formar uma matriz
5) Usa essa matriz transformar os n critérios em k critérios.

Como a classificação utilizando o teorema de Bayes tem a menor taxa de erro, o problema sempre está em como o classificador vai estimar a função de densidade. Em casos que a variância entre os grupos é muito diferente, como o algoritmo LDA estima essas variâncias com o uso de um dataset de treino (aprendizagem supervisionada), seu uso não é muito indicado e pode apresentar problemas. Nessas situações, outros modelos são mais indicados, como o QDA (Quadratic Discriminant Analysis).

C. KNN

KNN (K-Nearest Neighbors) é um método usado no campo de mining e machine learning para classificação. O algoritmo KNN é considerado de fácil implementação e o mais simples entre os outros algoritmos de aprendizado, sendo assim, pertencente ao grupo dos algoritmos de aprendizado preguiçoso (lazy learning).

O KNN pode ser vantajoso, pois é uma técnica simples e de fácil implementação, resultando em uma alta porcentagem de acurácia para objetos mais simples. Porém possui a desvantagem de ter alto tempo de processamento por causa do cálculo da distância entre cada elemento ou por causa do conjunto de treinamento grande.

O aprendizado no KNN consiste em quão similar uma instância do objeto é de outra. São encontrado os K vizinhos mais próximo do padrão de consulta e calculado a distância entre eles. A distância é calculada do elemento a ser classificado até seus vizinhos e então são ordenados em ordem de mais próximo até o de maior distância. Destes elementos ordenados, são selecionados os k primeiros que serão utilizados para a regra de classificação.

Para o KNN, dois pontos são importantes: a regra de classificação e a função que calcula a distância entre duas instâncias.

1) Regra de Classificação: A regra de classificação do KNN diz como o algoritmo vai tratar a importância dos k elementos selecionados. Para k=1, a regra é irrelevante, pois só há um elemento de escolha, logo só há um resultado para a classificação. Para k¿1, é preciso levar em conta duas regras clássicas: maioria dos votos e peso da distância. Na maioria dos votos, os elementos são tratados como iguais e a classe que aparecer mais na votação dos vizinhos será escolhida. Já no peso da distância, este peso é inversamente proporcional a distância.

2) Função de Cálculo da Distância: Existem várias formas de calcular a distãncia entre as intâncias do objeto. A mais simples utilizada é a distância euclidiana:

$$d(p, q) = \sqrt{\sum (p_i - q_i)^2} \tag{4}$$

Outras formas calcular a distância podem ser: distância de Mahalanobis, distância de Minkowsky, Hamming Distance, entre outras. Para todas as formas de calcular a distância, é necessário normalizar os dados.

III. EXPERIMENTOS E RESULTADOS

A. Pré-processamento e escolha dos critérios

Inicialmente o dados foram coletados e armazenados em arrays da biblioteca numpy, chamados de 'ndarray'. Como o

próprio dataset já separou os grupos de treino e test, isso não foi necessário no pré-processamento. Foram escolhidos dois critérios:

1) Soma dos valores em em escala de cinza, calculados na função *split_and_sum_image()*
2) Média dos valores em escala de cinza do centro da imagem, calculados na função *get_enter_mean()*

Também foi criada uma função chamada *split()*, que é utilizada dentro da *split_and_sum_image()*. Ela recebe uma matriz, um 'M' e um 'N', e retorna um vetor de submatrizes de tamanho MxN. Ela serve para dividir a imagem em setores de mesmo tamanho e seu uso será justificado mais a frente. O número de divisões tem que ser um múltiplo de 28 (tamanho da imagem).

Todas as funções de pré-processamento foram agrupadas em uma função chamada *preprocessing()* que recebe a imagem dos dígitos pré-processada e retorna um array de critérios retirados dessas imagens.

Agora com os dados tratados, foram desenvolvidas duas funções que para a aplicação do LDA e Knn no dataset.

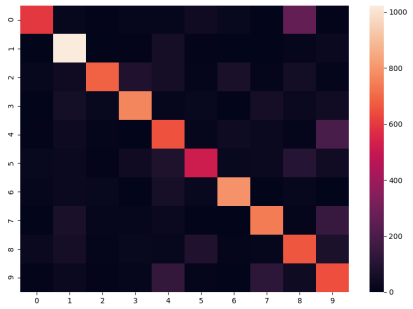
O *lda_classifier()*, que recebe como argumento a array de critérios e array dos grupos dos dados de treino e teste e número de dimensões do retorno da LDA. A função treina o preditor LDA com os dados (grupos e critérios) de treino e prediz os resultados com base nos critérios dos dados de teste. No final, calcula a acurácia com uma função do sklearn 'accuracy_score' e mostra uma matriz de confusão dos reusltados, utilizando a biblioteca seaborn.

O *knn_classifier()* que recebe como argumento a array de critérios e array dos grupos dos dados de treino e teste e o K do Knn. A função treina o preditor Knn com os dados (grupos e critérios) de treino e prediz os resultados com base nos critérios dos dados de teste. No final, assim como no LDA, calcula a acurácia e a matriz de confusão.

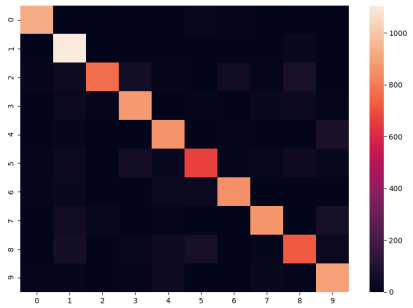
B. Resultados

Inicialmente, já no primeiro experimento com a LDA se concluiu que o atributo de soma dos valores em escala de cinza não seria suficiente, pois sua precisão foi de apenas 31%. Então o grupo tomou a decisão de aplicar o mesmo critérios, mas agora para vários setores da imagem e assim poder predizer melhor os resultados. Dividindo a imagem em setores, pode ser estimar a classe baseado nos padrões em cada setor e não apenas na imagem inteira como era antes. Com essa estratégia e o critério da média dos valores do centro, obtivemos o valor de precisão de 74% para a divisão em 16 setores(4x4) e precisão de 85,3% para a divisão em 49 setores(7x7).

Para realizar a escolha dos Ks vizinhos na classificação pelo algoritmo Knn, foram realizados testes com valores de K de 1 a 30 para verificação e procura da melhor acurácia. Ao realizar esses testes, foi percebido que a acurácia do algoritmo Knn foi em média de 84% com uma diferença maior quando o valor de K era 2, onde a acurácia caiu para 79%. Para escolher o valor de K, também deve ser lembrado que se o valor de K for muito pequeno, a classificação fica sensível a ruídos, e se o valor



(a) Para 16 categorias, acurácia 74%



(b) Para 49 categorias, acurácia 85,3%

Figura 1: Matriz de confusão do LDA por número de categorias

de K for muito grande, a vizinhança pode incluir elementos de outras classes. Sendo assim, os valores escolhidos para os vizinhos foram 2 e 15.

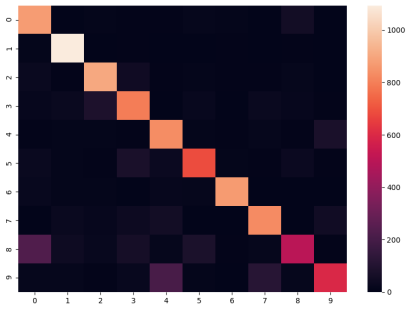
Essa acurácia é correspondente as imagens do dataset divididas em 16 setores (4x4). Para 49 setores (7x7) a precisão do algoritmo chega a 93,3% para k igual a 2 e 93,7% para k igual a 15.

Por último, os algoritmos foram testados quando o dataset foi dividido em 196 setores (14x14) e as precisões encontradas foram de 87,6%, 94,6% e % para o LDA, Knn onde k igual a 2 e Knn onde k igual a 15.

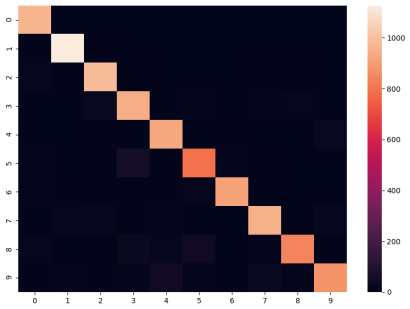
As figuras 1, 2 e 3 são as matrizes de confusão do experimento, para diferentes categorias a fim de efeitos de comparação. Percebe-se claramente que ao aumentar o tamanho de subdivisões da imagem, aumenta-se a precisão do algoritmo, mas que essa melhora diminui significativamente conforme o valores de divisões aumenta.

IV. CONCLUSÃO

O trabalho permitiu que fosse verificado dois algoritmos para classificação de um objeto. Para este trabalho foram utilizados os algoritmos LDA e KNN e comparado os resultados obtidos por ambos. Foi percebido que o algoritmo KNN resultou em uma melhor acurácia na classificação do objeto em estudo do que o algoritmo LDA. Este resultado se dá pelo motivo do objeto ser mais simples que o utilizado na realidade



(a) Para 16 categorias, acurácia 83,9%



(b) Para 49 categorias, acurácia 93,3%

Figura 2: Matriz de confusão do Knn (k=2) por número de categorias

do Machine Learning, sendo assim, como explicado na parte teórica, o algoritmo KNN possui uma alta porcentagem de acurácia sobre esses objetos. Porém, pelos conceitos apresentados neste relatório, ao utilizarmos objetos de complexidade maior que o estudado neste trabalho, iremos obter uma maior acurácia utilizando o algoritmo LDA ao invés do KNN.

O objetivo deste trabalho foi de realizar uma classificação e estudar os princípios de classificação utilizados em sistemas inteligentes. Pode-se afirmar que o objetivo foi alcançado com sucesso e sem dificuldades. Foi necessário utilizar de anotações de aula e algumas pesquisa na internet para se ter o conhecimento teórico necessário para ser colocado em prática por meio do código fonte.

Referências

[1] Anotações de aula. Prof Díbio

[2] Algoritmo de Aprendizagem de Máquina - Conceitos Básicos, Cap. 3. https://www.wattpad.com/story/36202147-algoritmos-de-aprendizagem-de-máquinas.

[3] Raschka, Sebastian - Linear Discriminant Analysis.

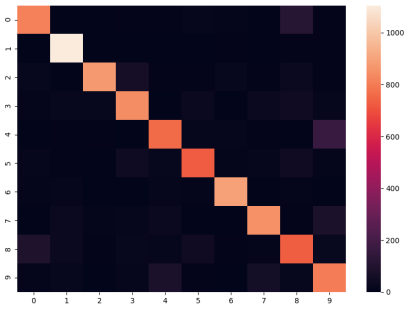
[4] Classification—Linear Discriminant Analysis. https://towardsdatascience.com/classification-part-2-linear-discriminant-analysis-ea60c45b9ee5.

[5] Linear discriminant analysis. https://en.wikipedia.org/wiki/Linear_discriminant_analysis.

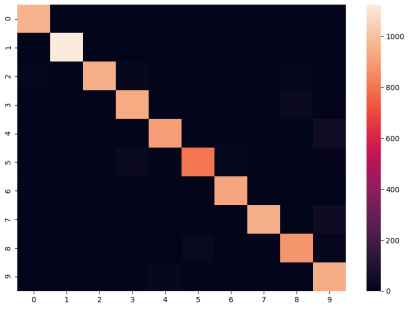
[6] Balakrishnama, Suresh and Ganapathiraju, Aravind. Linear discriminant analysis-a brief tutorial.

[7] Cover, Thomas M and Hart, Peter E and others. Nearest neighbor pattern classification.

[8] Li, Ming and Yuan, Baozong. 2D-LDA: A statistical linear discriminant analysis for image matrix.



(a) Para 16 categorias, acurácia 87,2%



(b) Para 49 categorias, acurácia 93,7%

Figura 3: Matriz de confusão do Knn (k=15) por número de categorias