UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
CAMPUS DE SÃO JOÃO DA BOA VISTA

KAUANE PRISCO

Compensação de efeitos não lineares em interconectores ópticos utilizando DBSCAN

São João da Boa Vista
2024

Kauane Prisco

Compensação de efeitos não lineares em interconectores ópticos utilizando DBSCAN

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Eletrônica e de Telecomunicações do Campus de São João da Boa Vista, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Eletrônica e de Telecomunicações .

Orientador: Profº Dr. Ivan Aritz Aldaya Garde

São João da Boa Vista
2024

P959c

Prisco, Kauane
Compensação de efeitos não lineares em interconectores ópticos utilizando DBSCAN / Kauane Prisco. -- São João da Boa Vista, 2024
55 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Telecomunicações) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, São João da Boa Vista
Orientador: Ivan Aritz Aldaya Garde

1. Aprendizado do computador. 2. Comunicações ópticas. 3. Sistemas não lineares. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA - CÂMPUS DE SÃO JOÃO DA BOA VISTA
GRADUAÇÃO EM ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

TRABALHO DE CONCLUSÃO DE CURSO

COMPENSAÇÃO DE EFEITOS NÃO LINEARES EM INTERCONECTORES ÓPTICOS UTILIZANDO DBSCAN

Aluno: Kauane Prisco
Orientador: Prof. Dr. Ivan Aritz Aldaya Garde

Banca Examinadora:

- Ivan Aritz Aldaya Garde (Orientador)
- Guilherme Simon da Rosa (Examinador)
- Rafael Abrantes Penchel (Examinador)

Os formulários de avaliação e a ata da defesa, na qual consta a aprovação do trabalho, devidamente assinados pela banca encontram-se no prontuário eletrônico do aluno.

São João da Boa Vista, 30 de outubro de 2024

AGRADECIMENTOS

Em primeiro lugar, quero expressar minha profunda gratidão ao meu pai, que sempre me apoiou incondicionalmente, assumindo tanto o papel de pai quanto de mãe em minha vida. Sou imensamente grata por tudo o que ele representa para mim.

Agradeço de coração à família Adduci, que entrou em minha vida no ano em que perdi uma das pessoas mais importantes para mim, minha avó Maria Lucia Santos. Eles me acolheram com carinho e sempre me incentivaram, tanto nos estudos quanto no meu desenvolvimento pessoal.

A Matheus, meu sincero agradecimento por estar ao meu lado nos momentos mais desafiadores e nos mais felizes. É uma grande alegria fortalecer nossa união a cada dia.

Sou grata também aos meus padrinhos, que cuidaram de mim desde o meu nascimento, e aos meus amigos da UNESP, que me apoiaram nesta etapa final do curso.

Por fim, agradeço ao meu orientador, Ivan Aritz Aldaya Garde, pelas oportunidades e orientações valiosas ao longo da minha graduação, que foram essenciais para meu crescimento acadêmico e pessoal.

RESUMO

Este trabalho investiga a eficácia do algoritmo de agrupamento espacial baseado em densidade, conhecido como DBSCAN (*Density-based spatial clustering of applications with noise*), em combinação com o método K-ésimo Vizinho mais Próximo (*K — Nearest Neighbors*, KNN), para mitigar distorções não lineares em um sistema digital coerente com enlace de 175 km e modulação 16-QAM. A otimização dos hiperparâmetros aplicados resultou em melhorias significativas no desempenho do sistema. Os resultados mostram que o DBSCAN, quando utilizado isoladamente, apresentou altas taxas de erro de bit (*Bit Error Rate*, BER), enquanto a combinação DBSCAN+KNN demonstrou uma capacidade eficaz de classificar pontos ruidosos. Essa abordagem destaca a importância da classificação na melhoria da qualidade do sinal, contribuindo para a redução dos ruídos. Embora o método K-means tenha se mostrado uma alternativa competitiva, a combinação DBSCAN+KNN se destacou especialmente em condições de maior potência. Este estudo ressalta a relevância de estratégias híbridas de agrupamento e classificação para melhorar o desempenho de sistemas de comunicação óptica, evidenciando seu potencial para aplicações práticas em ambientes ruidosos.

PALAVRAS-CHAVE: k-means; DBSCAN; KNN; machine learning; comunicações ópticas; não-linearidades.

ABSTRACT

This work investigates the effectiveness of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm combined with K-Nearest Neighbors (KNN) for mitigating nonlinear distortions in a coherent digital system with a 175 km link and 16-QAM modulation. Optimizing the applied hyperparameters led to significant improvements in system performance.

The results indicate that DBSCAN alone exhibited high Bit Error Rate (BER) values, whereas the DBSCAN+KNN combination demonstrated an effective ability to classify noisy points. This approach underscores the importance of classification in enhancing signal quality, contributing to a reduction in noise. Although K-means emerged as a competitive alternative, the DBSCAN+KNN method particularly excelled under higher power conditions.

This study highlights the relevance of hybrid clustering and classification strategies in improving the performance of optical communication systems, emphasizing their potential for practical applications in noisy environments.

KEYWORDS: k-means; DBSCAN; KNN; machine learning; optical communications; nonlinearities.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DBSCAN	Agrupamento espacial de aplicações com ruído baseado em densidade - *Density-based spatial clustering of applications with noise*
KNN	K-ésimo Vizinho mais Próximo - *K — Nearest Neighbors*
BER	Taxa de erros de bit - *Bit Error Rate*
LO	Laser oscilador local - *Local Oscillator*
EDFA	Amplificadores ópticos de fibra dopada com érbio - *Erbium Doped Fiber Amplifier*
WDM	Multiplexação por divisão em comprimento de onda - *Wavelength Division Multiplexing*
QAM	Modulação de amplitude em quadratura - *Quadrature Amplitude Modulation*
GVD	Dispersão de velocidade de grupo - *Group-Velocity Dispersion*
PMD	Dispersão de modo de polarização - *Polarization Mode Dispersion*
DSP	Processamento de sinal digital - *Digital Signal Processing*
QPSK	Modulação de chaveamento de mudança de fase em quadratura - *Quadrature Phase Shift Keying*
DCI	Interconexões entre datacenters - *Datacenter Interconnect*
16QAM	Modulação de amplitude em quadratura de polarização dupla com dezesseis estados - *Quadrature Amplitude Modulation*
DWDM	Multiplexação por divisão de comprimento de onda densa - *Dense Wavelength Division Multiplexing*
SPM	Auto-modulação de fase - *Self-Phase Modulation*
XPM	Modulação cruzada de fase - *Cross-Phase Modulation*
FWM	Mistura de quatro ondas - *Four Wave Mixing*
SMF	Fibras de modo único - *Single-Mode Fiber*
DBP	Retro-propagação digital - *Digital Back-Propagation*
IVSTF	Função de transferência da série de Volterra inversa - *Inverse Volterra Series Transfer Function*
ML	Aprendizagem de máquina - *Machine Learning*

PRBS	Sequências de bits pseudoaleatórios - *Pseudorandom Bit Sequences*
DAC	Conversor digital/analógico - *Digital-to-Analog Converter*
PBC	Combinador de feixe de polarização - *Polarization Beam Combiner*
SSMF	Padrão Fibra Monomodo - *Standard Single-Mode Fiber*
PON	Rede óptica passiva - *Passive Optical Network*
PBS	Divisor de feixe de polarização - *Polarization Beam Splitter*
PDs	Fotodetectores - *Photodetectors*
CD	Dispersão cromática - *Chromatic Dispersion*
MMA	Algoritmo de múltiplos módulos - *Multi-Modulus Algorithm*
MV	Máxima Verossimilhança - *Maximum Likelihood*

LISTA DE SÍMBOLOS

n	Índice de refração linear
n_j	Índice de refração para a polarização j
n'_j	Índice de refração modificado pelo efeito Kerr
$\bar{n}_2$	Coeficiente de índice de refração não linear
P	Potência óptica
A_{eff}	Área modal efetiva da fibra
β	Constante de propagação original
β'	Constante de propagação alterada pela não linearidade
k_0	Número de onda no vácuo
γ	Parâmetro não linear da fibra
Φ_{NL}	Fase não linear
L	Comprimento da fibra óptica
$P(z)$	Potência em função da posição z ao longo da fibra
P_{in}	Potência de entrada na fibra óptica
L_{eff}	Comprimento efetivo da fibra
α	Coeficiente de perda da fibra óptica
λ	Comprimento de onda da luz
Φ_j^{NL}	Deslocamento de fase não linear para o j-ésimo canal
P_j	Potência do j-ésimo canal óptico
P_m	Potência de outros canais ópticos ($m \neq j$)
$\chi^{(3)}$	Coeficiente de não linearidade de terceira ordem

SUMÁRIO

1 INTRODUÇÃO

1.1 SISTEMA DE COMUNICAÇÕES ÓPTICAS COERENTES DIGITAIS

No início da década de 1990, diversas pesquisas concentraram-se na utilização de formatos de modulação eficientes combinados com detecção óptica coerente (Betti et al., 1995). Esse método consiste em combinar o sinal óptico recebido com o de um laser oscilador local (*Local Oscillator*, LO), convertendo todas as informações contidas no sinal para o domínio elétrico antes da digitalização (Neto, 2014). Contudo, a detecção coerente perdeu relevância com o advento dos amplificadores ópticos de fibra dopada com érbio (*Erbium Doped Fiber Amplifier*, EDFA) e das técnicas de multiplexação por divisão em comprimento de onda (*Wavelength Division Multiplexing*, WDM). Essas inovações permitiram explorar plenamente a ampla banda de operação das fibras ópticas, possibilitando transmissões de longas distâncias e alta capacidade sem a necessidade de regeneradores optoeletrônicos (Tonguz and Wagner, 1991). Assim, os sistemas passaram a adotar esquemas mais simples e com excelente relação custo-benefício, abrindo novas possibilidades para a infraestrutura de comunicação óptica e permitindo um crescimento exponencial na capacidade de transmissão de dados a longas distâncias (Diniz, 2013).

O ano de 2005 marcou um ponto significativo na evolução das comunicações ópticas coerentes, quando a demonstração da estimativa digital da fase da portadora em receptores coerentes revitalizou o interesse nessa área (S. Tsukamoto, 2005). Essa renovação se deve à capacidade dos receptores coerentes digitais de utilizar uma variedade de formatos de modulação altamente eficientes em termos espectrais, como o M-chaveamento de mudança de fase e a modulação de amplitude em quadratura (*Quadrature Amplitude Modulation*, QAM). Essa versatilidade é viável graças a uma estimativa estável da fase da portadora no domínio digital. Além disso, a preservação das informações de fase após a detecção possibilita a correção de distorções lineares de transmissão, como a dispersão de velocidade de grupo (*Group-Velocity Dispersion*, GVD) e a dispersão de modo de polarização (*Polarization Mode Dispersion*, PMD), por meio do processamento de sinal digital (*Digital Signal Processing*, DSP) (Kikuchi, 2016). Essas características oferecem um potencial significativo para revolucionar os sistemas de comunicação óptica existentes. Recentemente, sistemas de transmissão com velocidades de 100 Gb/s foram desenvolvidos, utilizando modulação de chaveamento de mudança de fase em quadratura (*Quadrature Phase Shift Keying*, QPSK), multiplexação por divisão de polarização e detecção homódina de diversidade de fase, assistidos por DSP de alta velocidade a uma taxa de símbolo de 25 GBd (Kikuchi, 2011). Esses sistemas têm sido implementados em redes comerciais, possibilitando a colocação de canais de multiplexação por divisão de comprimento de onda em grades espaçadas de 50 GHz, resultando em uma capacidade total de transmissão de até 8,8 Tb/s em uma única fibra (Kikuchi, 2011). Atualmente, esforços globais estão em andamento para desenvolver receptores coerentes capazes de lidar com taxas de bits superiores a 400 Gb/s por canal WDM (Kikuchi, 2016).

Os sistemas coerentes digitais, amplamente empregados em redes ópticas de longo alcance, como conexões intercontinentais e redes metropolitanas, revolucionaram o campo das telecomunicações. Com vantagens notáveis, como alta sensibilidade do receptor e elevada eficiência espectral, além da capacidade de compensar impedimentos por meio de técnicas de DSP, esses sistemas permitiram um

aumento expressivo no produto comprimento-largura de banda (Winzer et al., 2018).

Recentemente, houve um avanço na adoção de sistemas coerentes digitais em ambientes menores, como as interconexões entre datacenters (*datacenter interconnect*, DCI). Esse movimento é impulsionado pela crescente demanda por maior eficiência e capacidade de transmissão em conexões de curta e média distância, devido ao aumento exponencial de dados gerados e processados em datacenters. Um exemplo notável é o padrão de comunicações 400ZR (OIF400ZR).

O 400ZR é um acordo de implementação de rede interoperável publicado pelo OIF (*Optical Interworking Forum*) em março de 2020. Seu foco principal é em redes curtas para interconexão de datacenters utilizando fibra óptica (OIF, 2020). O padrão 400ZR transmitirá cargas úteis Ethernet (*Gigabit Ethernet*, GE) de até 400 Gb em linhas DCI variando de 80 km a 120 km, utilizando multiplexação por divisão de comprimento de onda densa (*Dense Wavelength Division Multiplexing*, DWDM) e modulação de alta ordem. O objetivo é garantir uma implantação de baixo custo e de longo prazo, baseando-se na operação de portadora única em 400G, empregando modulação de amplitude em quadratura de polarização dupla com dezesseis estados (*Quadrature Amplitude Modulation*, 16QAM) a uma taxa de dez gigabits por segundo (OIF400ZR).

1.2 EFEITO DAS NÃO LINEARIDADES EM SISTEMAS DE COMUNICAÇÕES ÓPTICA DIGITAIS

Em sistemas de comunicação óptica, fenômenos não lineares, como o espalhamento de Brillouin e Raman, além do efeito Kerr, podem impactar significativamente o desempenho, dependendo da potência óptica transmitida pela fibra. Esses efeitos podem limitar a eficiência e a qualidade da transmissão (Agrawal, 2014). O efeito Kerr é frequentemente considerado a principal fonte de distorção não linear, gerando fenômenos como auto-modulação de fase (*Self-Phase Modulation*, SPM), modulação cruzada de fase (*Cross-Phase Modulation*, XPM) e mistura de quatro ondas (*Four-Wave Mixing*, FWM). Tais fenômenos são de grande relevância nos sistemas coerentes, pois convertem variações de intensidade em variações de fase (Agrawal, 2021).

A ocorrência do efeito Kerr deve-se à variação do índice de refração local na presença de um campo óptico intenso, especialmente quando múltiplos sinais de diferentes comprimentos de onda se propagam simultaneamente em uma fibra, particularmente em fibras de modo único (*Single-Mode Fiber*, SMF), onde a alta intensidade é concentrada devido à pequena área modal. O efeito Kerr é caracterizado pela suscetibilidade elétrica de terceira ordem, que causa diversas distorções no sinal óptico (Agrawal, 2014; Oliveira Junior, 2020).

1.2.1 Auto-Modulação de Fase (SPM)

A SPM ocorre quando a intensidade da luz altera o índice de refração de um material óptico, um efeito causado pela presença do efeito Kerr. Esse efeito aumenta o índice de refração n proporcionalmente à intensidade luminosa, o que pode impactar significativamente o desempenho de sistemas ópticos de telecomunicações (Agrawal, 2014).

O índice de refração n', que considera o efeito Kerr, é dado por:

$$n' = n + \bar{n}_2 \frac{P}{A_{eff}}, \tag{1}$$

onde $\bar{n}_2$ é o coeficiente de índice não linear, P é a potência óptica, e A_{eff} é a área modal efetiva. Assim, a intensidade da luz provoca um aumento em n', resultando na auto-modulação de fase, fenômeno em que a fase da onda óptica é alterada pela própria intensidade.

Na fibra óptica, essa modificação altera a constante de propagação β, que se torna dependente da intensidade da luz. A nova constante de propagação, β', é dada por:

$$\beta' = \beta + \gamma P, \tag{2}$$

onde γ é o parâmetro não linear, definido como:

$$\gamma = \frac{2\pi \bar{n}_2}{\lambda A_{eff}}. \tag{3}$$

A fase não linear acumulada ao longo da fibra, Φ_{NL}, depende dessa variação em β e é expressa por:

$$\Phi_{NL} = \int_0^L \gamma P(z)\, dz. \tag{4}$$

Se a potência $P(z)$ é constante ao longo da fibra e igual à potência de entrada P_{in}, a fase não linear pode ser simplificada para:

$$\Phi_{NL} = \gamma P_{in} L_{eff}, \tag{5}$$

onde L_{eff} é o comprimento efetivo da fibra, dado por:

$$L_{eff} = \frac{1 - e^{-\alpha L}}{\alpha}, \tag{6}$$

e α é o coeficiente de atenuação da fibra.

1.2.2 Modulação de Fase Cruzada (XPM)

Outra característica não linear, conhecida como modulação de fase cruzada, é causada pela dependência do índice de refração em relação à intensidade. Esse fenômeno ocorre quando dois ou mais canais ópticos são transmitidos simultaneamente em uma fibra óptica utilizando a técnica de multiplexação por divisão em comprimento de onda. Nesses sistemas, a modificação da fase não linear de um canal específico não depende apenas de sua própria potência, mas também da potência dos outros canais (Agrawal, 2014). O deslocamento de fase para o j-ésimo canal é dado por:

$$\Phi_j^{NL} = \gamma L_{eff} \left(P_j + 2 \sum_{m \neq j} P_m \right). \tag{7}$$

A somatória se estende sobre a quantidade de canais presentes (Agrawal, 2021).

1.2.3 Mistura de Quatro Ondas (FWM)

A mistura de quatro ondas é um fenômeno que ocorre em fibras ópticas quando quatro sinais luminosos de diferentes frequências interagem e se combinam, resultando em um novo sinal com uma frequência distinta das originais. Esse fenômeno permite uma maior capacidade de transmissão de dados, possibilitando a transmissão simultânea de mais informações no mesmo comprimento de onda (Carman et al., 1966).

A FWM é um efeito da não linearidade de terceira ordem, descrita pelo coeficiente $\chi^{(3)}$. Isso ocorre quando pelo menos duas frequências distintas da componente óptica se propagam juntas em um meio não linear.

Considerando duas componentes de frequência de entrada co-propagadas, v_1 e v_2 (com $\nu_2 > \nu_1$), a modulação do índice de refração acontece na frequência de diferença, gerando duas novas componentes de frequência:

$$\nu_3 = \nu_1 - (\nu_2 - \nu_1) = 2 \cdot \nu_1 - \nu_2, \tag{8}$$

$$\nu_4 = \nu_2 - (\nu_2 - \nu_1) = 2 \cdot \nu_2 - \nu_1. \tag{9}$$

Adicionalmente, podemos calcular as frequências ν_5 e ν_6:

$$\nu_5 = 2 \cdot \nu_1 + \nu_2, \tag{10}$$

$$\nu_6 = \nu_1 + 2 \cdot \nu_2. \tag{11}$$

Entretanto, essas frequências adicionais são menos comuns devido à dificuldade em obter uma combinação coerente em fase, especialmente em fibras. Além disso, há a possibilidade de amplificação paramétrica das ondas pré-existentes nas frequências ν_3 e ν_4 (Carman et al., 1966).

1.3 MITIGAÇÃO DAS NÃO LINEARIDADES EM SISTEMAS COERENTES DIGITAIS

Diversas técnicas são utilizadas em sistemas de comunicações ópticas para compensar as não linearidades, como:

- **Compensação *mid-span*:** Refere-se a um ponto intermediário em um link de fibra óptica onde dispositivos são aplicados para melhorar o desempenho do sistema. Este ponto pode ser crucial para mitigar problemas como a perda de sinal e distorções causadas por efeitos não lineares na fibra óptica (Chou et al., 2000).

- **Método *twinwaves*:** Consiste em utilizar duas ondas gêmeas conjugadas em fase, alinhadas de maneira anti-correlacionada. As distorções que afetam uma onda tendem a cancelar as distorções da outra. Ao combinar essas duas ondas ao final da transmissão, é possível reduzir os problemas causados por essas distorções (Liu et al., 2013).

- **Técnicas baseadas em eletrônica:** Essas técnicas oferecem facilidade de implementação para compensar as não linearidades (de Mesquita Argel, 2018).

As abordagens iniciais para compensação digital eram fundamentadas na inversão do modelo. Um exemplo é a retro-propagação digital (*digital back-propagation*, DBP), uma técnica avançada usada em comunicações ópticas para melhorar a qualidade do sinal transmitido através de fibras ópticas, especialmente em sistemas de alta capacidade e longas distâncias (da Silva et al., 2015). Durante a transmissão, o sinal sofre várias distorções devido a efeitos como a dispersão cromática e a não linearidade (Agrawal, 2014). A ideia do DBP é compensar essas distorções aplicando um processo inverso às ocorridas durante a transmissão. Na simulação de propagação inversa, o sinal recebido é digitalmente processado. O primeiro passo é criar uma simulação digital da fibra óptica que tenta replicar os efeitos de distorção que o sinal sofreu. Com base nessa simulação, o DBP aplica algoritmos de compensação ao sinal recebido, incluindo ajustes para a dispersão cromática e correção de efeitos não lineares. Este processo é iterativo, onde o sinal é ajustado várias vezes para melhorar gradualmente a qualidade, aproximando-se do sinal original transmitido (Sillekens et al., 2020).

Outra técnica é a função de transferência da série de Volterra inversa (*Inverse Volterra Series Transfer Function*, IVSTF) (Liu et al., 2011). Para lidar com efeitos não lineares, aplicamos a transferência da série de Volterra inversa, uma forma matemática de modelar sistemas não lineares, que visa "reverter"ou compensar os efeitos não lineares ocorridos durante a transmissão. Embora essas técnicas apresentem desempenho satisfatório, elas têm desvantagens, como a necessidade de grandes quantidades de operações, o que pode ser um obstáculo para aplicações em tempo real (COSTA, Camila et al., 2022). Por esse motivo, aplicações utilizando Machine Learning (*Machine Learning*, ML) estão sendo exploradas para reduzir as complexidades computacionais, empregando algoritmos classificados como supervisionados, não supervisionados e de reforço (Cajahuanca et al., 2022). Algumas dessas técnicas já foram implementadas em sistemas de comunicações ópticas para melhorar o desempenho.

- **K-means:** O método K-means foi empregado para reduzir os impactos de não linearidades em sistemas ópticos digitais coerentes DP-16QAM. Este algoritmo de aprendizado não supervisionado divide um conjunto de dados em grupos, chamados *clusters*, identificando k centroides que minimizam a soma dos quadrados das distâncias entre os pontos de dados e seus centroides correspondentes (COSTA, Camila et al., 2022).

- **Rede Neural Artificial:** A aplicação de redes neurais artificiais mostrou resultados promissores na mitigação de não linearidades em sistemas de comunicações ópticas coerentes digitais com multiplexação de polarização. As redes neurais, inspiradas no funcionamento do cérebro humano, são compostas por camadas de unidades chamadas neurônios. Durante o treinamento, a rede ajusta seus parâmetros com base nos dados de entrada, aprimorando suas previsões ou classificações (Cossa, 2023).

- **DBSCAN + KNN:** No projeto de tecnologia aprimorada de recuperação de sinal, foi combinada a técnica de agrupamento espacial de aplicações com ruído baseado em densidade (*Density-based spatial clustering of applications with noise*, DBSCAN) e o algoritmo K-vizinhos mais próximos

(*K-Nearest Neighbors*, KNN). O DBSCAN identifica *clusters* como regiões de alta densidade de amostras, classificando pontos com baixa densidade como *outliers*. Após essa identificação, o KNN classifica os pontos restantes com base nos k pontos de treinamento mais próximos, permitindo a redução do ruído e a melhoria da qualidade do sinal (Huang et al., 2022).

Essas são algumas das técnicas já aplicadas, mas muitos outros métodos de *Machine Learning* estão disponíveis além dos mencionados.

1.4 OBJETIVO

Aplicações de *Machine Learning* estão sendo exploradas para a mitigação de efeitos não lineares, visando reduzir a complexidade computacional e possibilitar sua aplicação em tempo real. Técnicas de clusterização, como o DBSCAN, combinadas com métodos de classificação, como o KNN, já foram estudadas e apresentam resultados publicados. Entretanto, o diferencial deste trabalho está na otimização de hiperparâmetros específicos — ϵ e *min_samples* do DBSCAN e o parâmetro k do KNN — uma abordagem que ainda não foi explorada. Neste estudo, compararemos as técnicas K-means e DBSCAN+KNN para avaliar qual delas apresenta a menor taxa de erro de bit (*Bit Error Rate*, BER). Além disso, analisaremos o desempenho do sistema para uma única polarização em um nível de potência inicial e, em seguida, em diversos níveis de potência. Focaremos inicialmente na polarização simples, gerando 16 *clusters*; ao considerar polarização dupla, o número de *clusters* aumentaria para 256, o que elevaria significativamente a complexidade do trabalho. A análise com polarização dupla será considerada como proposta para estudos futuros.

1.5 ESTRUTURA DO TRABALHO

Este trabalho será estruturado da seguinte forma:

- **Capítulo 2** - Descrição do algoritmo KNN.
- **Capítulo 3** - Apresentação do algoritmo DBSCAN.
- **Capítulo 4** - Simulação do sistema.
- **Capítulo 5** - Apresentação dos resultados.
- **Capítulo 6** - Conclusão do trabalho, resumindo os principais achados e as implicações dos resultados.

2 DBSCAN

O algoritmo DBSCAN é uma técnica de clusterização que pertence ao grupo de métodos de aprendizado não supervisionado, pois não requer rótulos ou categorias pré-definidas para as amostras de dados. Em vez disso, ele classifica as amostras com base na densidade das regiões, identificando pontos centrais de alta densidade e expandindo os *clusters* a partir deles. Esse método é capaz de detectar múltiplos *clusters* com diferentes características nos dados, permitindo a identificação de *clusters* de variados formatos e tamanhos em grandes conjuntos de dados, mesmo na presença de ruídos e *outliers* (Ester et al., 1996).

Os dois hiperparâmetros principais que governam a operação do DBSCAN são:

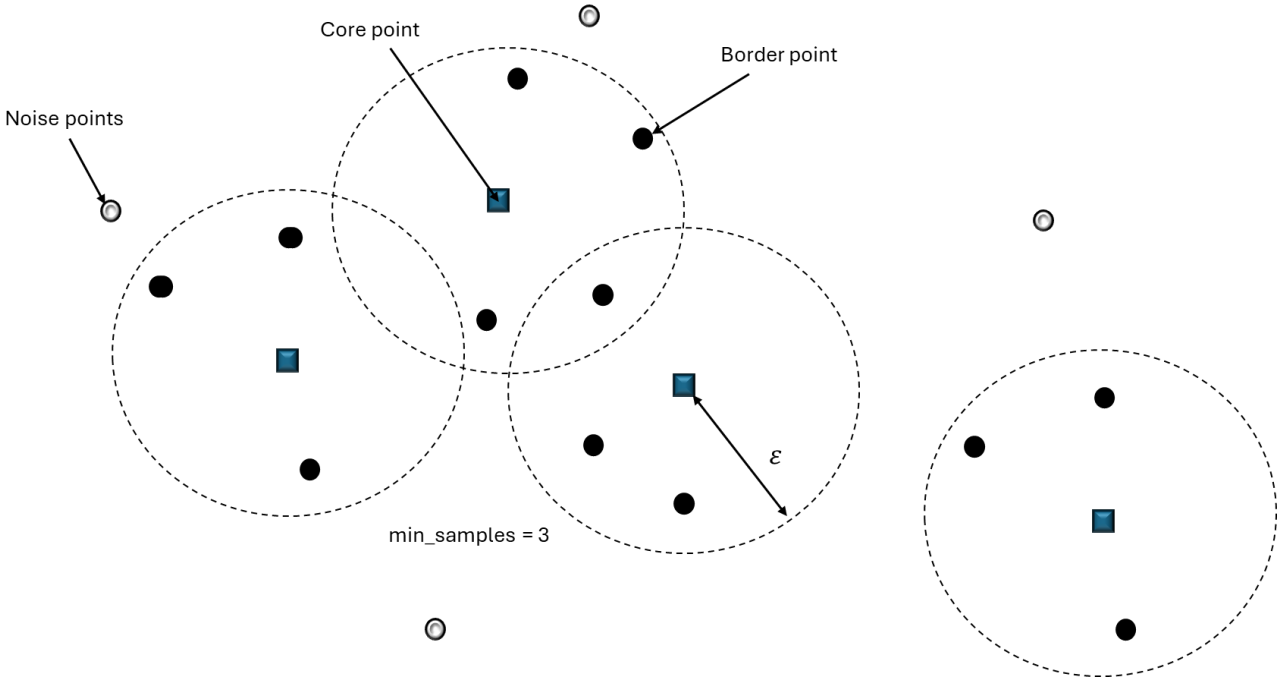
- `min_samples`: número mínimo de amostras que uma região deve ter para ser considerada densa.
- `eps` (ϵ): distância máxima entre duas amostras para que uma seja considerada vizinha da outra.

Na Figura 1, identificam-se os pontos centrais, também conhecidos como *core points* ou núcleos. A partir desses pontos, determina-se um valor de ϵ para delimitar a área abrangida pelo círculo pontilhado na figura. Dentro desse círculo, define-se um número mínimo de amostras, que neste caso é *min_samples* $= 3$. Esse procedimento é repetido para todos os pontos centrais, e quaisquer pontos que não estejam contidos dentro desse círculo são classificados como "pontos de ruído"(*noise points*).

As etapas do procedimento são as seguintes:

- O algoritmo inicia selecionando aleatoriamente um ponto do conjunto de dados e continua esse processo até que todos os pontos tenham sido analisados.
- Se existirem pelo menos `min_samples` pontos dentro de uma distância ϵ do ponto selecionado, todos esses pontos são considerados parte do mesmo *cluster*.
- A expansão dos *clusters* ocorre por meio da repetição recursiva do cálculo da vizinhança para cada ponto adjacente.

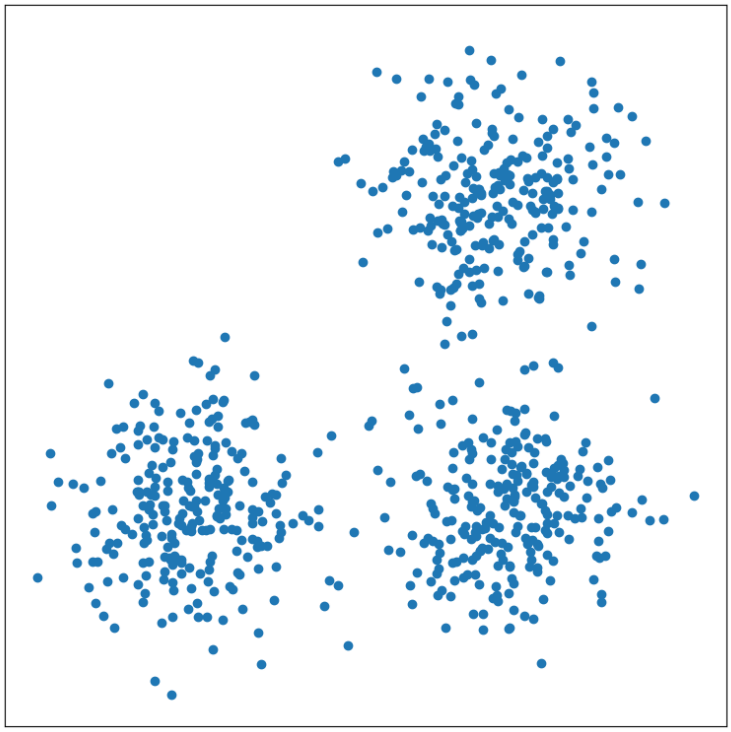
Figura 1 – Método utilizado no DBSCAN.



Elaborado pela autora.

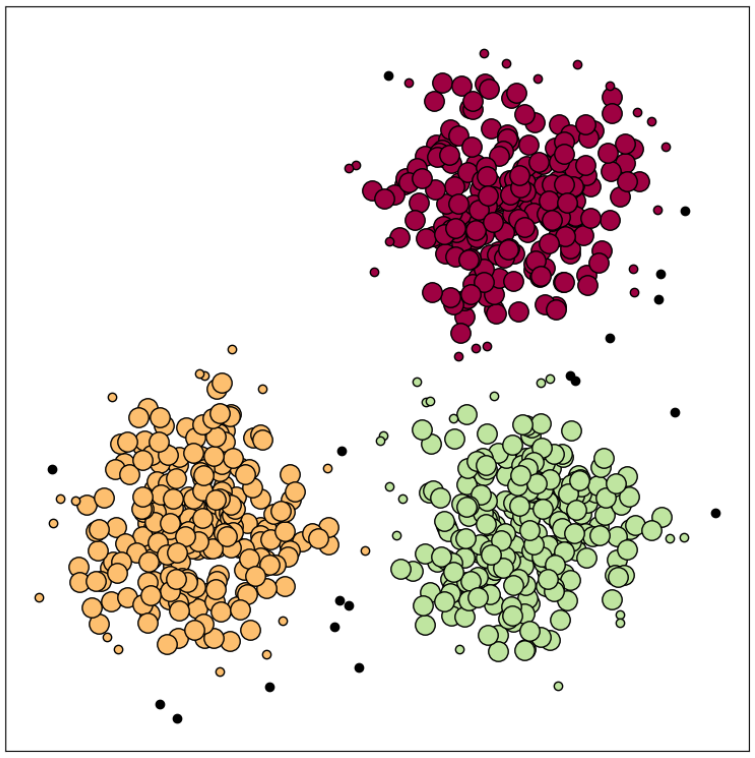
A seguir, temos duas figuras: a Figura 2, que mostra os dados iniciais sem agrupamento, e a Figura 3, onde aplicamos o algoritmo DBSCAN para obter as classificações. Na segunda imagem, é possível observar a formação de três *clusters* distintos, além da presença de *outliers* (pontos ruidosos), representados pelos pontos pretos.

Figura 2 – DBSCAN sem classificação.



Elaborado pela autora.

Figura 3 – DBSCAN com a classificação.



Elaborado pela autora.

3 KNN

O KNN é um algoritmo supervisionado, pois requer um conjunto de dados rotulados para aprender a associar características específicas a determinadas classes e, assim, fazer classificações de novos dados. O processo supervisionado envolve, então, utilizar amostras de treinamento com rótulos para que o modelo possa "aprender"a categorizar pontos com base nas características observadas.

Para realizar a classificação, o KNN calcula a similaridade entre o ponto que deseja classificar e os pontos rotulados em seu entorno. A métrica mais comumente usada para essa comparação é a distância euclidiana, definida como:

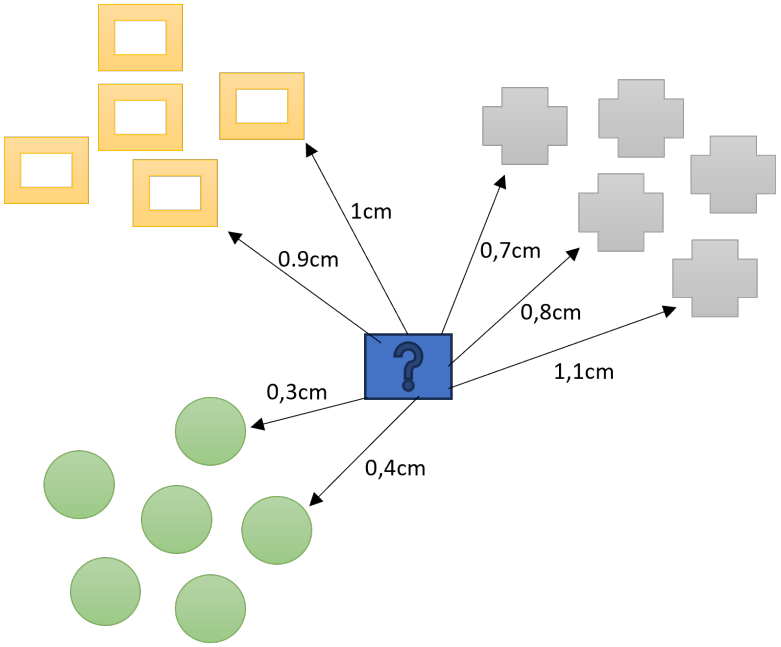
$$d(i,j) = \sqrt{\sum_{m=1}^{n}(x_i^m - x_j^m)^2},$$

onde $d(i,j)$ representa a distância entre dois pontos i e j em um espaço n-dimensional, e x_i^m e x_j^m são as coordenadas dos pontos em cada dimensão m. Essa métrica é escolhida, pois mede a proximidade direta no espaço, favorecendo a classificação de pontos com características semelhantes em regiões próximas. Em situações com dispersão nos dados, o KNN tende a agrupar pontos de classes semelhantes na mesma região, o que facilita a identificação de padrões entre os dados.

Ao definir o número de vizinhos (k), o KNN identifica a classe predominante entre os k-ésimos vizinhos mais próximos e atribui esta classe ao ponto em análise (Fix and Hodges, 1989).

A seguir, apresentamos um exemplo para ilustrar melhor este método. Confira a figura abaixo:

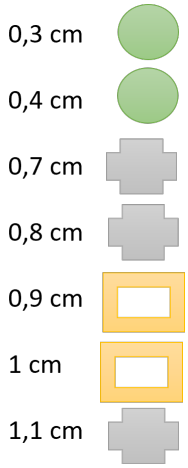
Figura 4 – Exemplo de classificação utilizando KNN.



Elaborado pela autora.

Na Figura 4, temos um grupo de círculos, retângulos e cruzes; no entanto, existe um ponto de interrogação cuja associação a um grupo é desconhecida. Para isso, utilizamos a distância euclidiana. Vamos calcular os objetos mais próximos a esse ponto para descobrir a qual *cluster* ele pertence. Após realizar todas as medições, ordenamos as distâncias em ordem crescente.

Figura 5 – Tabela de distância de cada figura.



Elaborado pela autora.

Após ordenarmos as figuras da menor para a maior distância, podemos selecionar o valor de k que determinará a classe do vizinho mais próximo. Se escolhermos $k = 2$, a classe será definida pelas duas figuras mais próximas, resultando na seleção do "círculo", que possui a menor distância. Ao considerarmos $k = 3$, observamos a presença de dois "círculos"e uma "cruz". Nesse caso, aplicamos o critério de desempate, e a classe majoritária, que é o "círculo", é escolhida novamente.

Por outro lado, ao utilizarmos $k = 4$, encontramos dois "círculos"e duas "cruzes", o que gera um empate. Nessa situação, não conseguimos determinar a qual grupo o ponto deve pertencer. Essas observações nos levam à conclusão de que é fundamental optar por um valor de k ímpar para evitar empates, assegurando, assim, uma decisão clara na classificação dos pontos.

No KNN, além da distância euclidiana, podemos utilizar outras três medidas de distância:

- **Distância de Hamming**: Esta métrica calcula a distância entre vetores binários, contando o número de posições em que os bits dos vetores são diferentes (Pena et al., 2018).
- **Distância de Manhattan**: Esta métrica calcula a distância entre vetores reais somando as diferenças absolutas entre suas coordenadas. Também é conhecida como Distância City Block (Gao and Li, 2020).
- **Distância de Minkowski**: Esta métrica é uma generalização das distâncias Euclidiana e de Manhattan, incorporando diferentes expoentes para ajustar a sensibilidade em relação às dimensões dos vetores (Marques, 2021).

4 SIMULAÇÃO DO SISTEMA

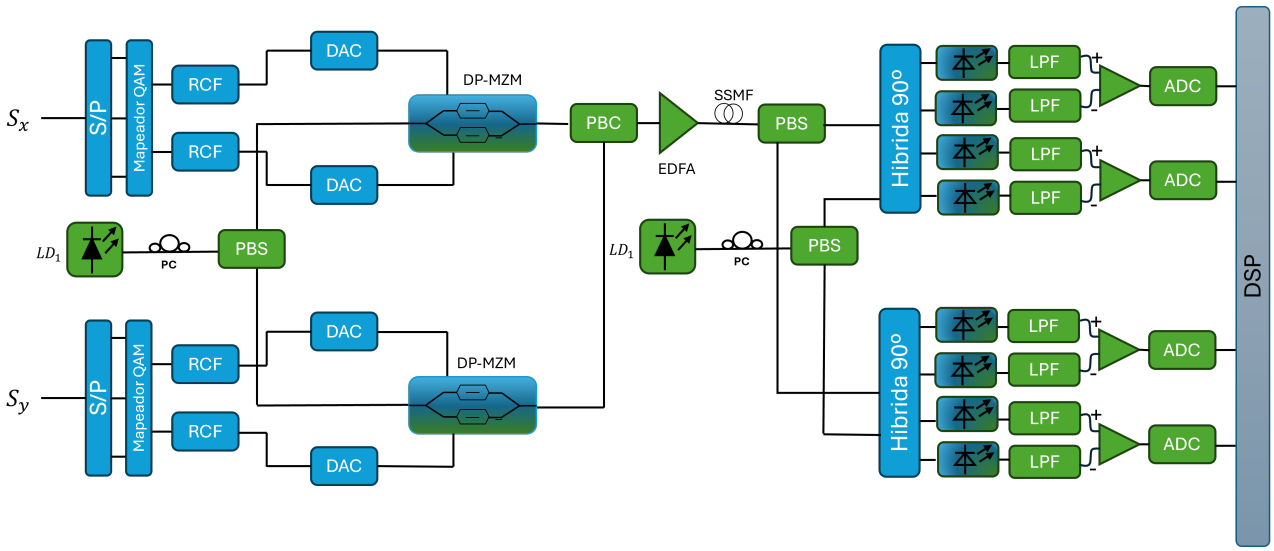
4.1 CONFIGURAÇÃO DA SIMULAÇÃO DO SISTEMA

As simulações foram conduzidas pela estudante de mestrado Camila Costa, e seus detalhes podem ser encontrados em sua dissertação (COSTA, Camila et al., 2022).

Para realizar as simulações, utilizou-se o software *Transmission Maker*, que permite a criação de sinais por meio de diversos métodos de modulação. Neste trabalho, optou-se pela modulação 16-QAM, abordando também as etapas de transmissão e detecção do sinal. No processamento digital do sinal (*Digital Signal Processing*, DSP), o sistema foi projetado para compensar as dispersões cromáticas. Por fim, aplicaram-se técnicas de clusterização, utilizando os algoritmos DBSCAN combinados com KNN, com o objetivo de mitigar as não linearidades da fibra óptica.

A Figura 6 ilustra o diagrama de blocos da simulação para verificar o desemprenho do sistema em termos de potência óptica lançada (*Launched Optical Power*, LOP) e comprimento do enlace de fibra.

Figura 6 – Diagrama de blocos da simulação usada para análise de sistemas coerentes digitais com DP-16QAM. Componentes: S/P (conversor serial/paralelo), RCF (filtro de cosseno levantado), DAC (conversor digital/analógico), DP-MZM (modulador Mach-Zehnder paralelo), LD (diodo laser), PC (controlador de polarização), PBS (divisor de feixe por polarização), PBC (combinador de feixe por polarização), EDFA (amplificador de fibra dopada com érbio), SSMF (fibra monomodo), LPF (filtro passa-baixa), ADC (conversor analógico/digital) e DSP (processamento de sinal digital).



Fonte: Adaptado de (COSTA, Camila et al., 2022).

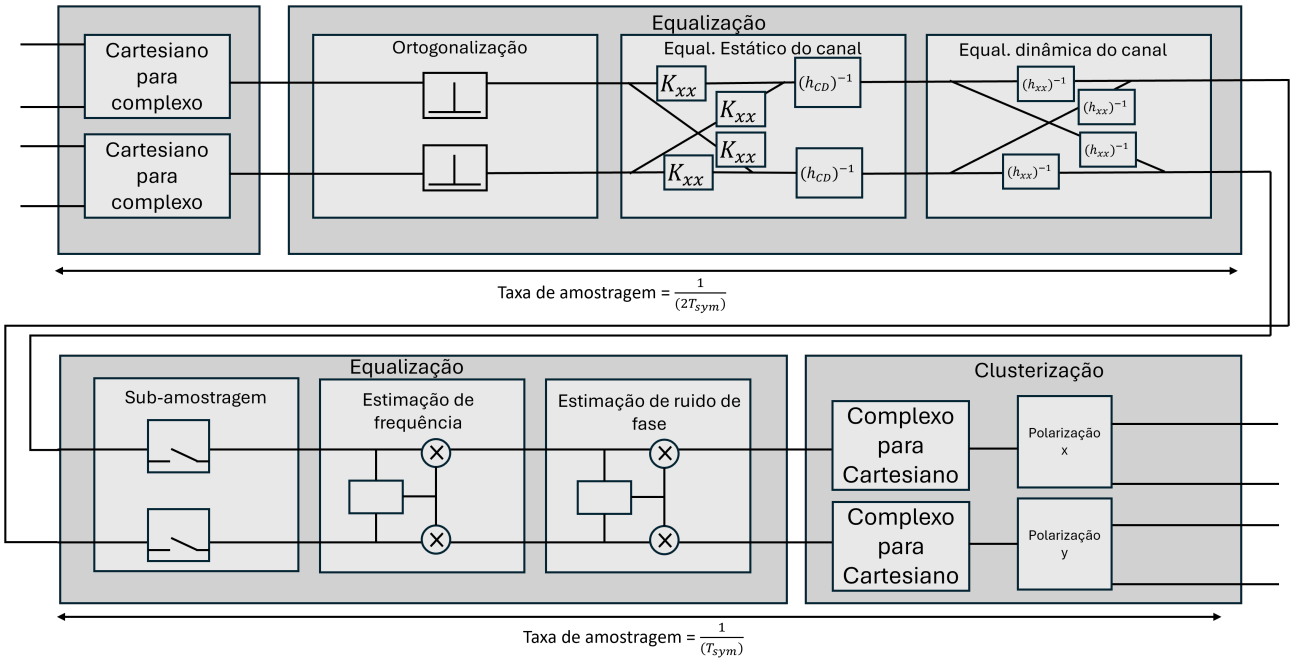
No transmissor, foram geradas duas sequências de bits pseudo-aleatórias (*pseudorandom bit sequences*, PRBS) de maneira independente, as quais representam as informações a serem transmitidas através das polarizações ortogonais da portadora óptica. Inicialmente, essas sequências foram organizadas em grupos de 4 bits, utilizando um conversor de serial para paralelo (S/P). Cada um desses grupos foi então convertido em um símbolo da constelação 16QAM. As componentes em fase e quadratura

do símbolo passaram por um processo de sobre-amostragem. Posteriormente, o sinal irá passar pelo filtro de cosseno levantado (*raised cosine pulse*, RCF), onde será filtrado pelo filtro de Nyquist com formato de cosseno levantado, apresentando um fator de *roll-off* de 20%. Os sinais foram novamente sobre-amostrados e passarão pelo conversor digital/analógico (*digital-to-analog converter*, DAC).

O diodo de laser (*laser diode*, LD_1) emite luz com uma largura de linha de 200 kHz e está configurado para operar a 1550 nm. A luz emitida passa por um controlador de polarização (*polarization controller*, PC), que ajusta sua polarização, e em seguida é direcionada para um divisor de feixe por polarização (*polarization beam splitter*, PBS). Este divisor separa os feixes em polarizações vertical e horizontal, permitindo que sejam modulados de forma independente. As informações provenientes do DAC são então moduladas pelo Mach-Zehnder duplo-paralelo (*dual-parallel Mach-Zehnder modulator*, DP-MZM) utilizando a luz emitida pelo diodo de laser. Após a modulação, os sinais são combinados através do combinador de feixe por polarização (*polarization beam combiner*, PBC). Uma vez combinados, os sinais são amplificados por pelo amplificador EDFA, que foi configurado com uma figura de ruído de 3,8 dB. Além disso, o amplificador controla a potência óptica lançada na entrada do enlace, variando de 0 a 15 dBm. Esse mesmo sinal é transmitido através de um *span* de fibra monomodo padrão (*standard single-mode fiber*, SSMF), onde os comprimentos do enlace variam de 125 a 175 km.

No receptor, o PBS divide o sinal óptico em duas partes ortogonais, utilizando um divisor de feixe por polarização. Cada uma dessas partes é então combinada com o sinal do oscilador local em uma rede híbrida de 90º. Os sinais resultantes são detectados por fotodetectores balanceados e, em seguida, passam pelo filtro passa-baixa (*low pass filter*, LPF) para filtrar altas frequências. Após isso, o sinal passa pelo amplificador diferencial, onde os quatro sinais correspondentes à fase e quadratura das duas polarizações são sub-amostrados para obter duas amostras por símbolo. Posteriormente, o sinal é transformado de analógico para digital (*analog-to-digital converter*, ADC) para que possa ser processado no DSP.

Figura 7 – Diagrama de blocos do processamento de sinal digital utilizado na simulação.



Fonte: Adaptado de (COSTA, Camila et al., 2022).

Na Figura 7, é possível observar a configuração do sistema de processamento digital de sinais DSP, que é composto por diversas etapas de processamento. O sistema é organizado em subsistemas, cada um responsável por compensar a dispersão cromática.

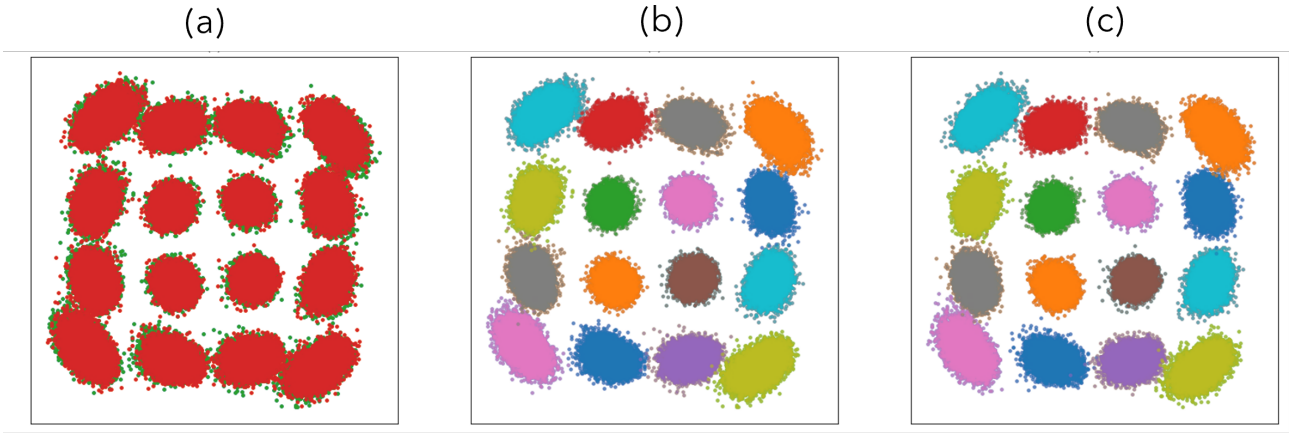
O primeiro módulo do DSP converte as componentes em fase e quadratura em amostras complexas. Em seguida, essas amostras são paralelizadas para compensar a dispersão cromática no domínio da frequência. Após a correção da dispersão cromática, é realizado o alinhamento de polarização utilizando 32 símbolos de treinamento. Em seguida, a taxa de amostragem é reduzida para uma única amostra por símbolo. Essa etapa requer um processo de sincronização do tempo, implementado através de um algoritmo de Módulos Múltiplos (*Multi-Modulus Algorithm*, MMA), que opera em uma versão sobre-amostrada do sinal.

Além disso, foi realizada uma varredura dupla, variando a potência óptica lançada por polarização de 0 a 15 dBm, enquanto o comprimento do enlace da fibra óptica foi ajustado de 125 a 175 km, utilizando modulação 16-QAM. Também foi simulado um sistema WDM de 5 canais para avaliar o desempenho do algoritmo proposto na presença de efeitos não-lineares inter-canal (COSTA, Camila et al., 2022).

Após a simulação, os símbolos obtidos foram coletados para análises, incluindo máxima verossimilhança, k-means, DBSCAN combinado com KNN e inicialização em 2D. Essas análises têm como objetivo mitigar as não linearidades e comparar quais dos métodos apresentam a menor BER, considerando tanto um único nível de potência quanto diversos níveis de potência.

Na Figura 8, é possível visualizar a constelação para ambas as polarizações, permitindo uma análise da qualidade de recepção dos sinais modulados em cada uma delas. A representação separada das polarizações auxilia na identificação dos efeitos não lineares e distorções que podem impactar os símbolos de forma distinta. Ao observarmos a distribuição de todos os pontos, nota-se que as polarizações *X* e *Y* apresentam características semelhantes. Essa análise foi implementada em Python, com os dados específicos para o enlace de 175 km e potência de 10 dBm.

Figura 8 – Representação da constelação para um único nível de potência. (a) Conjunto de todos os pontos para ambas as polarizações; (b) Constelação rotulada da polarização X; (c) Constelação rotulada da polarização Y.



Elaborado pela autora.

A seguir, na Tabela 1, veja os valores dos parâmetros utilizados na simulação:

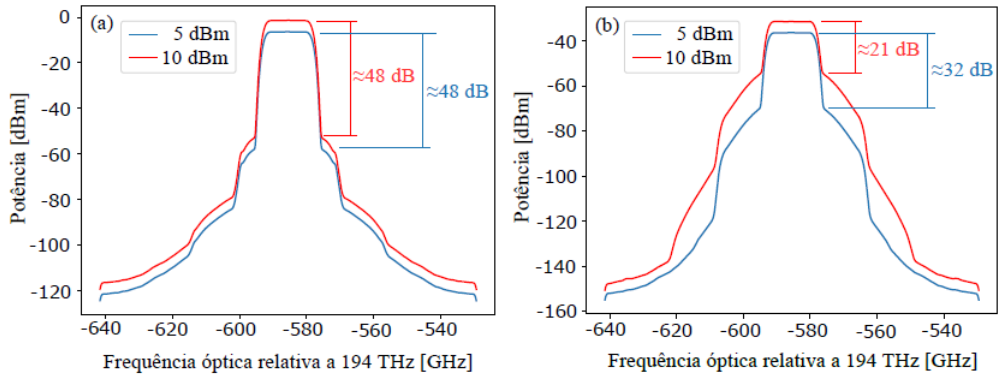
Tabela 1 – Lista dos parâmetros empregados na simulação.

	Parâmetro	**Valor**	**Unidade**
Transmissor	Número de canais	1-5	-
	Espaçamento entre canais	100	GHz
	Largura de banda	50	GHz
	Potência do laser	0	dBm
	Largura de linha do laser	200	kHz
	Comprimento de onda de operação	1550	nm
	Perda de inserção do modulador	6	dB
	Razão de extinção do modulador	35	dB
	Figura de ruído do amplificador	3,8	dB
	Potência de saída do amplificador	0-15	dBm
Fibra	Comprimento da fibra	125-175	km
	Parâmetro de dispersão	16	ps/(nm km)
	Inclinação de dispersão	0,08	ps/(nm² km)
	Coeficiente de atenuação	0,2	dB/Km
	índice não linear	0,26	μm²/GW
	Área efetiva modal	80	μm²
	Coeficiente PMD	0,01	ps/$\sqrt{km}$
	Comprimento de coerência PMD	50	m
Receptor	Potência do laser do oscilador local	14	dBm
	Largura de linha do oscilador local	200	kHz
	Frequência de operação nominal	193,1	THz
	Desvio de frequência	50	MHz
	Responsividade do fotodiodo	0,7	A/W
	Densidade de corrente térmica do fotodiodo	21	pA/$\sqrt{Hz}$
	Ruído de disparo do fotodiodo	Sim	-
DSP	Número de etapas no equalizador dinâmico	25	-
	Número de iterações no equalizador dinâmico	20	-
	Número de símbolos na detecção de fase	5	-
Sinal	Formato de modulação	DP-16QAM	-
	Taxa de transmissão de símbolos	14	GBaud
	Taxa transmissão de *bits*	112	Gb/s
	Fator de roll-off	0,2	-
	Número de símbolos simulados	262.144	-
	Taxa de amostragem da simulação	896	GSa
	Amostras por símbolo no transmissor	8	-

Fonte: (COSTA, Camila et al., 2022).

4.2 OBSERVAÇÕES DOS EFEITOS NÃO LINEARES NA SIMULAÇÃO

Figura 9 – Analise do espectros de (a) entrada e (b) saída para as potências de 5dBm e 10dBm lançadas na fibra.



Fonte: (COSTA, Camila et al., 2022).

Na entrada do enlace de fibra, o espectro do sinal é avaliado para duas potências de transmissão: 5 dBm e 10 dBm. Embora os espectros de entrada apresentem formas semelhantes, refletindo componentes de frequência equivalentes, existe um deslocamento vertical de 5 dB entre eles, que indica a diferença de potência de transmissão. A razão sinal-interferência (*signal-interference*, SIR) na entrada é de aproximadamente 48 dB, independentemente da potência utilizada. Essa interferência inicial pode ser atribuída a fatores como a distorção não linear que ocorre no transmissor.

Na saída do enlace de fibra, notamos a influência das não linearidades intrínsecas da própria fibra, que resulta na diminuição da SIR. Para a potência de 5 dBm, a SIR reduz-se de 48 dB para cerca de 32 dB, indicando que as não linearidades da fibra introduzem distorções e interferências adicionais. Para 10 dBm, a redução da SIR é ainda mais pronunciada, caindo para 21 dB. Nesse cenário, os efeitos não lineares se tornam mais intensos, levando a um aumento das componentes fora da banda original, o que resulta em uma distorção e interferência mais significativas.

Esse comportamento evidencia que um aumento na potência de transmissão intensifica os efeitos não lineares da fibra óptica, gerando componentes de frequência adicionais que interferem no sinal original. Com potências mais elevadas, a geração de componentes fora de banda se torna mais acentuada, tornando a interferência mais evidente e perceptível no espectro de saída (COSTA, Camila et al., 2022).

5 RESULTADOS

Após a coleta dos dados da simulação, concentramos nossa análise em uma única polarização, utilizando os dados correspondentes a uma potência de 10 dBm em um enlace de 175 km. Na primeira etapa, calculamos a BER para as polarizações X e Y, aplicando o método de Máxima Verossimilhança. Em seguida, calculamos a média da BER das polarizações X e Y utilizando o mesmo método. Os resultados obtidos foram os seguintes:

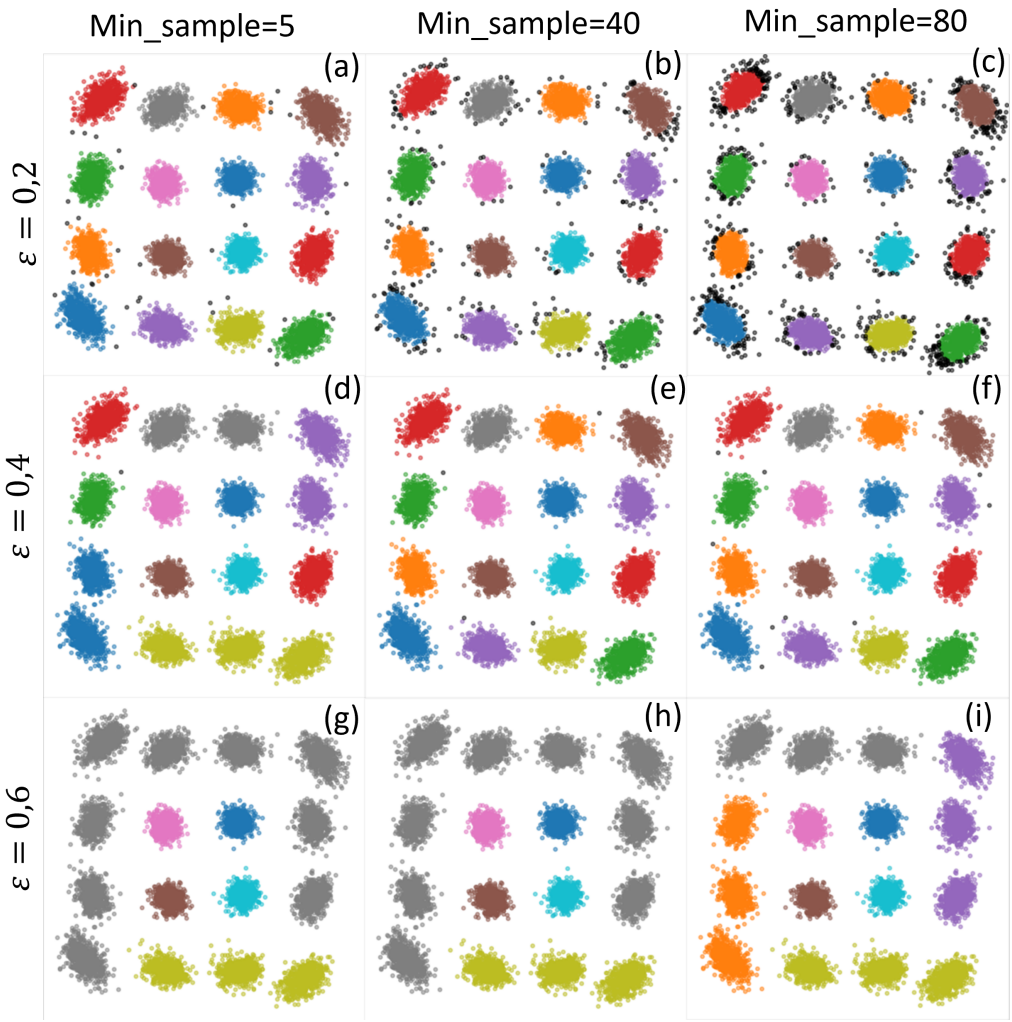
Tabela 2 – Resultado da BER utilizando a detecção por Máxima Verossimilhança.

Descrição	**Valor**
BER da polarização X utilizando MV	6.15×10^{-4}
BER da polarização Y utilizando MV	4.21×10^{-4}
Média da BER utilizando MV	5.18×10^{-4}

Elaborado pela autora.

Agora, iremos aplicar o algoritmo de clusterização DBSCAN, que aprenderá a realizar a clusterização dos dados. Como estamos analisando o sinal 16-QAM, é fundamental gerar 16 *clusters*. Para isso, devemos definir valores para os hiperparâmetros ϵ e *min_samples*, pois, através desses parâmetros, poderemos identificar quais valores são mais adequados para a clusterização. A Figura 10 a seguir apresenta os resultados dos *clusters* gerados, juntamente com os respectivos valores selecionados para os hiperparâmetros.

Figura 10 – Classificação utilizando o algoritmo DBSCAN, variando os parâmetros *min_samples* (número mínimo de amostras) e ϵ (distância máxima entre duas amostras).



Elaborado pela autora.

Tabela 3 – Resultados de Clustering DBSCAN com diferentes parâmetros e comparação de BER.

Parâmetros DBSCAN	**Nº *Clusters***	**Nº Pontos Ruidosos**	**BER (DBSCAN)**
ϵ = 0.2, min_samples = 5	16	39	9.75×10^{-4}
ϵ = 0.2, min_samples = 40	16	268	6.70×10^{-3}
ϵ = 0.2, min_samples = 80	16	798	1.995×10^{-2}
ϵ = 0.4, min_samples = 5	11	1	7.6375×10^{-2}
ϵ = 0.4, min_samples = 40	16	4	1.5×10^{-4}
ϵ = 0.4, min_samples = 80	16	10	3.25×10^{-4}
ϵ = 0.6, min_samples = 5	6	0	1.55175×10^{-1}
ϵ = 0.6, min_samples = 40	6	0	1.55175×10^{-1}
ϵ = 0.6, min_samples = 80	8	0	1.243×10^{-1}

Elaborado pela autora.

As imagens de (a) a (i) na Figura 10 mostram os resultados da aplicação do algoritmo DBSCAN com diferentes valores dos hiperparâmetros. Em cada figura, observamos os grupos de pontos coloridos,

representando os *clusters* identificados pelo DBSCAN, enquanto os pontos em preto representam ruídos ou *outliers*.

Para os casos em que $\epsilon = 0,2$ e *min_samples* = 5, 40 e 80, é possível observar a geração de 16 *clusters*, porém com muitos pontos não associados. Isso indica que, embora o modelo identifique os *clusters* esperados, ele classifica muitos pontos como ruído. Por outro lado, para $\epsilon = 0,6$ e *min_samples* = 5, 40 e 80, observamos nenhuma presença de pontos ruidosos, mas com a formação de um número menor de *clusters*. Esse comportamento sugere que o modelo tende a agrupar pontos de forma mais abrangente, resultando em menos *clusters* do que o desejado.

Ao analisarmos a imagem (c), com $\epsilon = 0,2$ e *min_samples* = 80, identificamos um caso de *overfitting*. Nessa configuração, o modelo gera 16 *clusters*, mas também gera um número elevado de pontos como ruído (798 pontos ruidosos, conforme mostrado na Tabela 3). Esse comportamento indica que o *DBSCAN* está capturando excessivamente as pequenas variações locais, tratando flutuações mínimas como *clusters* distintos. Consequentemente, a BER aumenta para $1,995 \times 10^{-2}$, indicando uma menor precisão na detecção dos *clusters* reais.

Em contraste, a imagem (g), com $\epsilon = 0,6$ e *min_samples* = 5, exemplifica um caso de *underfitting*. Aqui, o DBSCAN agrupa muitos pontos em *clusters* grandes, ignorando variações relevantes na distribuição dos dados. Como indicado na Tabela 3, o modelo identifica apenas 6 *clusters*, sem pontos ruidosos. Esse resultado sugere que o DBSCAN não conseguiu distinguir adequadamente os 16 *clusters* esperados, indicando que o modelo é incapaz de capturar a complexidade dos dados.

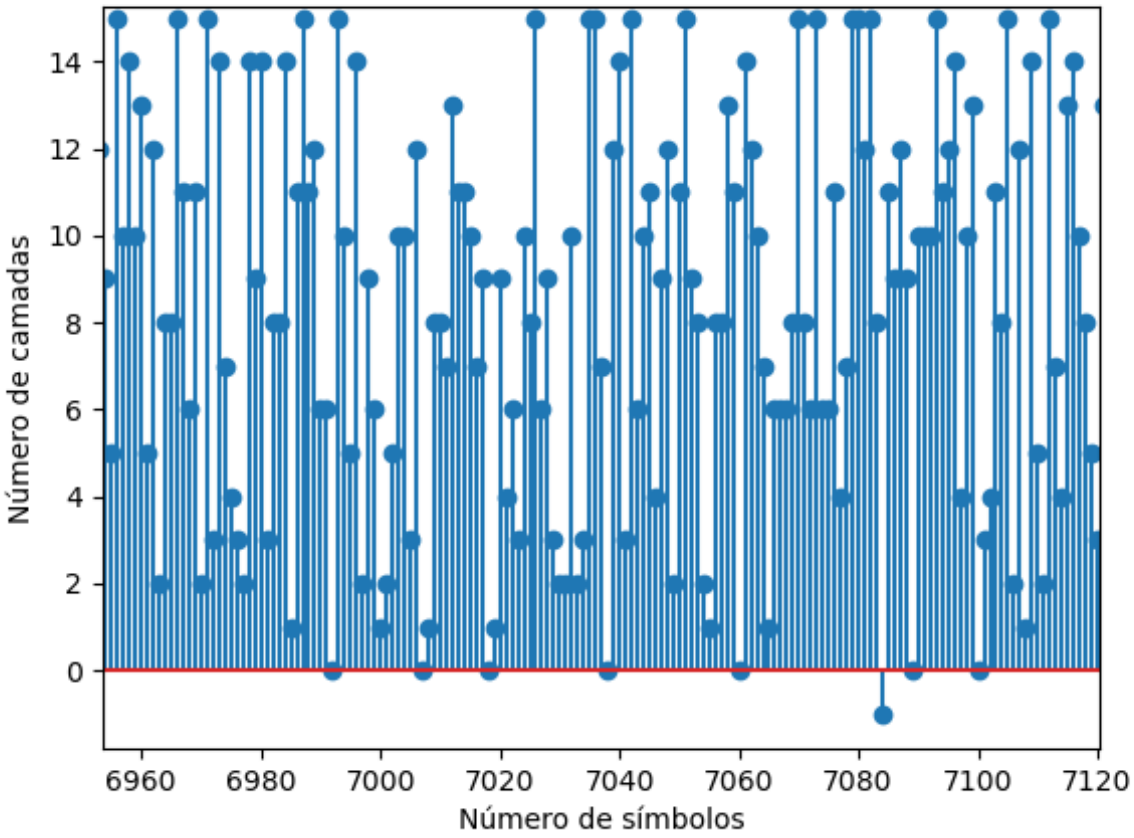
Por fim, para $\epsilon = 0,4$ e *min_samples* = 40, observamos a formação dos 16 *clusters* desejados, com apenas 4 pontos ruidosos. Essa configuração apresentou a menor BER, de $1,5 \times 10^{-4}$, o que indica um excelente equilíbrio entre a identificação dos *clusters* e a minimização dos ruídos. Portanto, essa combinação de hiperparâmetros foi selecionada como a mais adequada, e seguirá para o método de classificação KNN, que será utilizado para associar os pontos não classificados aos *clusters* existentes.

Tabela 4 – BER para os hiperparâmetros $\epsilon = 0,4$ e *min_samples* = 40 para a polarização X utilizando DBSCAN.

Método	**BER**
DBSCAN	1.5×10^{-4}
Pontos ruidosos	4
Número de *clusters*	16

Elaborado pela autora.

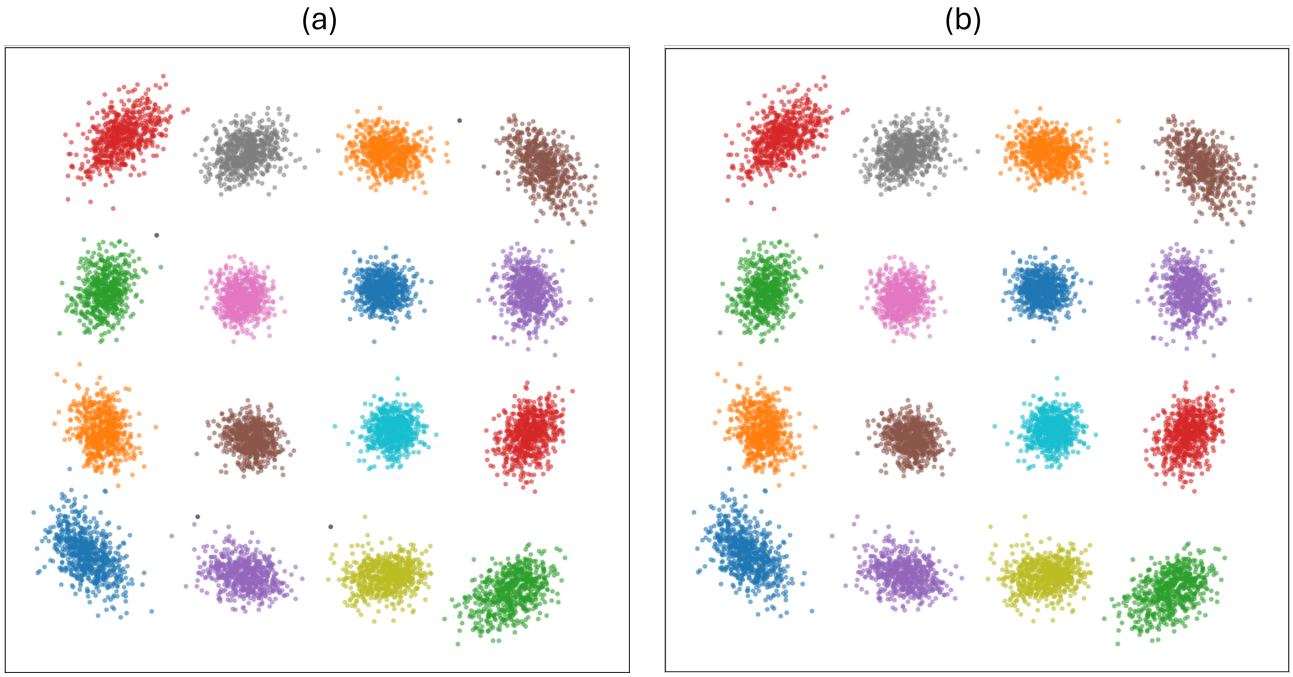
Figura 11 – Gráfico de haste da simulação DBSCAN com os parâmetros $\epsilon = 0.4$ e $\texttt{min\_samples} = 40$.



Elaborado pela autora.

Na Figura 11, é possível observar 16 níveis distintos no gráfico de hastes, correspondentes aos diferentes *clusters* identificados. Os pontos situados abaixo da linha vermelha representam elementos que não foram associados a nenhum *clusters*. Para essa simulação, foi utilizado um total de 10.000 símbolos para um único nível de potência. A imagem foi ampliada para facilitar a visualização dos níveis, e o intervalo exibido corresponde ao segmento de 6.960 a 7.120 símbolos, dentro do intervalo completo de 0 a 10.000 símbolos selecionados para análise.

Figura 12 – Aplicação do *DBSCAN* com os hiperparâmetros $\epsilon = 0.4$ e `min_samples` $= 40$ (a), seguido pela classificação dos pontos utilizando *KNN* com $k = 3$ (b).



Elaborado pela autora.

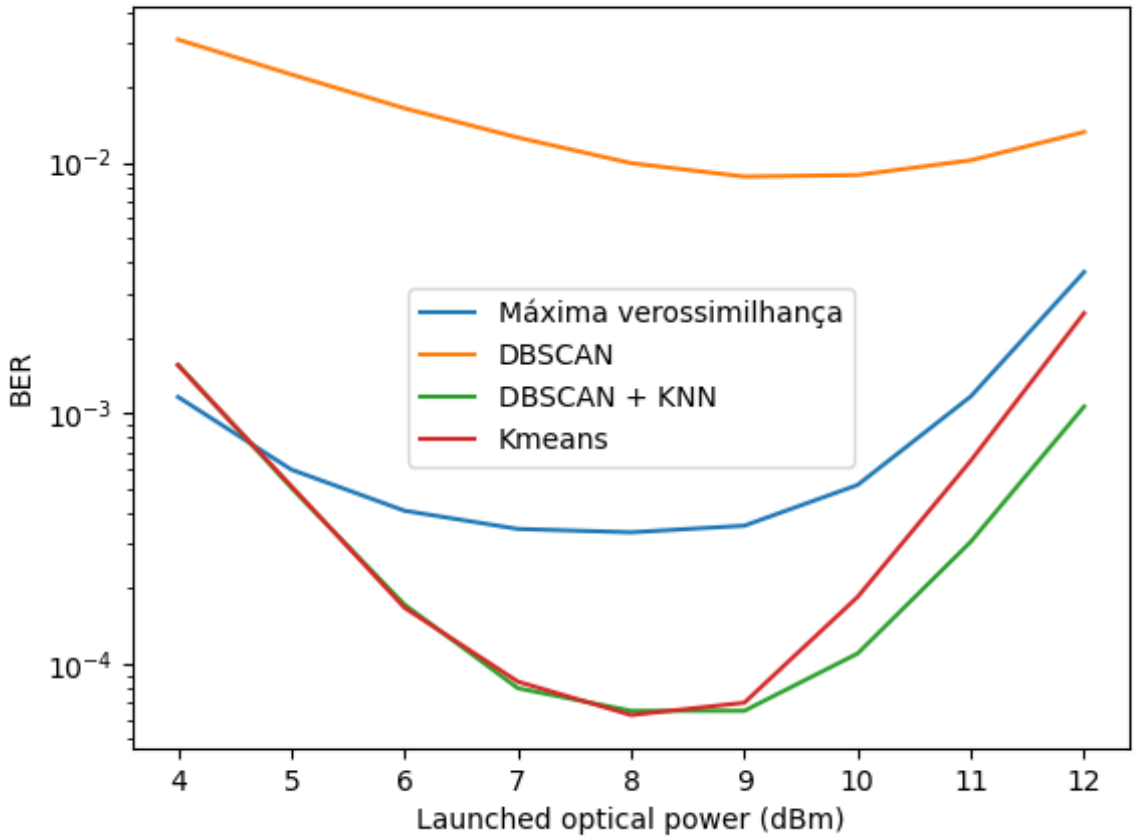
Tabela 5 – BER para a polarização X utilizando DBSCAN isolado e DBSCAN combinado com KNN.

Método	**BER**
DBSCAN	1.5×10^{-4}
DBSCAN + KNN	5×10^{-5}

Elaborado pela autora.

Após sintonizar os hiperparâmetros do DBSCAN, ϵ e `min_samples`, aplicamos o método de classificação KNN para associar os pontos não atribuídos aos clusters. Na Figura 12 (a), apresentamos o resultado da aplicação do DBSCAN, onde podemos observar os quatro pontos ruidosos. Já na Figura (b), mostramos o resultado da simulação da classificação utilizando KNN. Essa abordagem permitiu classificar os pontos considerados ruidosos, reduzindo assim o erro e resultando em uma BER de 5×10^{-5}. Os resultados evidenciam a importância da combinação entre o método de agrupamento e a classificação para um único nível de potência.

Figura 13 – Comparação dos resultados da BER para os métodos MV, DBSCAN, DBSCAN + KNN e KMeans em diferentes níveis de potência.



Elaborado pela autora.

Para a utilização do método DBSCAN e KNN, foram definidos os seguintes hiperparâmetros:

- Para o DBSCAN, utilizou-se $\epsilon = 0.4$ e `min_samples = 60`.
- Para o KNN, foi escolhido $k = 3$.

O uso combinado dos métodos DBSCAN e KNN demonstrou ser a abordagem mais eficaz para reduzir a BER nas simulações realizadas, superando tanto o método de Máxima Verossimilhança quanto o DBSCAN isolado para diversos níveis de potência, conforme a Figura 13. Esse resultado sugere que técnicas híbridas de agrupamento e classificação apresentam um grande potencial para otimizar o desempenho em sistemas de comunicação óptica, especialmente em cenários com alta presença de ruído.

Ao compararmos o método K-means, observamos que, para baixas potências, a diferença no desempenho entre as abordagens é mínima. Entretanto, em condições de potência mais elevada, a aplicação do DBSCAN + KNN se torna vantajosa, evidenciando sua superioridade na identificação e classificação de pontos ruidosos e, consequentemente, na redução da BER. Para altas potências, o método DBSCAN combinado com KNN apresentou a melhor BER em relação ao K-means, pois, em

altas potências, temos uma maior presença de não linearidades. Nesses casos, o método de agrupamento isolado pode não ser suficiente, resultando em um número maior de pontos não associados aos *clusters*. Por esse motivo, a combinação com um algoritmo de classificação é fundamental para altas potências.

6 CONCLUSÕES

Neste trabalho, nosso objetivo foi analisar o algoritmo de clusterização DBSCAN e a classificação KNN para mitigar distorções não lineares em um sistema coerente digital. O enlace avaliado teve uma extensão de 175 km, utilizando modulação 16-QAM. Um diferencial deste estudo foi a otimização dos hiperparâmetros aplicados, que possibilitou obter resultados significativos com o método DBSCAN + KNN.

A partir da análise dos resultados, podemos extrair as seguintes conclusões:

- Os valores da taxa de erro de bit do DBSCAN isolado mostraram-se consistentemente mais altos em comparação aos outros métodos. Isso se deve ao fato de que, sem a classificação adicional dos pontos ruidosos, muitos erros permanecem não corrigidos, resultando em uma BER elevada.
- Em todas as simulações, o método DBSCAN + KNN superou o DBSCAN isolado, evidenciando que a classificação adicional dos pontos ruidosos pelo KNN é uma estratégia eficaz para melhorar o desempenho geral.
- O K-means apresentou valores de BER ligeiramente superiores ao DBSCAN + KNN, mas ainda inferiores ao DBSCAN isolado. Isso demonstra que o K-means é uma técnica competitiva para classificação, embora o DBSCAN + KNN se destaque especialmente em potências mais altas.
- A significativa redução na taxa de erro de bit ao aplicar o KNN após o DBSCAN ressalta a importância de tratar adequadamente os pontos ruidosos em problemas de comunicação óptica. Isso evidencia que métodos de agrupamento com etapas de refinamento podem ser extremamente eficazes na minimização de erros de bit.
- Para altas potências, o método K-means apresentou um desempenho inferior em comparação ao DBSCAN combinado com KNN. Isso se deve ao fato de que, em altas potências, há uma maior presença de não linearidades. Nesses casos, apenas um algoritmo de clusterização pode não ser suficiente para capturar a complexidade dos dados. Por essa razão, os resultados evidenciam a importância tanto do método de agrupamento quanto do de classificação para lidar com altas potências, onde os pontos não associados aos clusters podem ser classificados posteriormente, reduzindo assim a BER.

Dessa forma, este trabalho apresentou resultados relevantes para a otimização dos hiperparâmetros, evidenciando que, em potências mais altas, a combinação DBSCAN + KNN pode reduzir a quantidade de pontos ruidosos, resultando em uma taxa de erro de bit mais baixa e robusta. Contudo, as diferenças em relação ao K-means foram mínimas, o que indica a necessidade de investigar a complexidade computacional entre esses métodos. O DBSCAN + KNN pode ser computacionalmente mais intensivo, especialmente em grandes volumes de dados ou em ambientes de alta dimensionalidade. Assim, se o tempo de processamento for um fator limitante, o K-means pode ser uma escolha mais vantajosa.

Para trabalhos futuros, planejamos aplicar o mesmo método em um cenário de dupla polarização. Essa abordagem será mais complexa, exigindo o agrupamento em quatro dimensões, o que resultará em um total de 256 aglomerados. Por essa razão, a investigação desse tema foi adiada para etapas posteriores. Contudo, os resultados obtidos para uma única polarização foram satisfatórios e promissores.

REFERÊNCIAS

Govind P. Agrawal. *Sistemas de comunicações por fibra óptica*. Elsevier Editora Ltda., Rio de Janeiro, RJ, Brazil, 2014.

Govind P. Agrawal. *Sistemas de Comunicação por Fibra Óptica*. Elsevier, Rio de Janeiro, RJ, Brazil, 2021.

S. Betti, G. de Marchis, and E. Iannone. Coherent optical communication systems. Master's thesis, Nova Iorque, NY, EUA: John Wiley & Sons, 1995.

Julio Elvis Valero Cajahuanca, Ángel Fernando Navarro Raymundo, Alfredo César Larios Franco, and Janett Deisy Julca Flores. Deserción universitaria: Evaluación de diferentes algoritmos de machine learning para su predicción. *Revista de ciencias sociales*, 28(3):362–375, 2022.

R. L. Carman, R. Y. Chiao, and P. L. Kelley. Observation of degenerate stimulated four-photon interaction and four-wave parametric amplification. *Phys. Rev. Lett.*, 17:1281–1283, Dec 1966. doi: 10.1103/PhysRevLett.17.1281. URL `https://link.aps.org/doi/10.1103/PhysRevLett.17.1281`.

MH Chou, I. Brener, G. Lenz, R. Scotti, EE Chaban, J. Shmulovich, D. Philen, S. Kosinski, KR Parameswaran, and MM Fejer. Inversor espectral de midspan sintonizável e de banda larga eficiente usando não linearidades em cascata em guias de onda de linbo3. 12:82–84, 2000. doi: 10.1109/68.817501.

Grazielle Caroline Sebastião Cossa. Mitigação do efeito das não linearidades em sistemas de comunicações ópticas coerentes digitais com multiplexação de polarização utilizando redes neurais artificiais. *Universidade Estadual Paulista (Unesp)*, 2023.

COSTA, Camila, Lucio Borges, Rafael A Penchel, Marcelo LF Abbade, Elias Giacoumidis, Jinlong Wei, Jose A de Oliveira, Mirian Santos, Shi Li, André Richter, et al. Self-phase modulation and inter-polarization cross-phase modulation mitigation in single-channel dp-16qam coherent pon employing 4d clustering. *SSRN Electronic Journal*, 01 2022. doi: 10.2139/ssrn.4063425.

Edson P da Silva, Knud J Larsen, and Darko Zibar. Impairment mitigation in superchannels with digital backpropagation and mlsd. *Optics Express*, 23(23):29493–29501, 2015.

Gonçalo Rafael de Mesquita Argel. Experimental validation of nonlinear effects compensation techniques in ofdm radio over fiber systems. Master's thesis, Universidade de Coimbra (Portugal), 2018.

Júlio César Medeiros Diniz. *Estimador de desvio de frequência para receptores ópticos coerentes digitais*. PhD thesis, Dissertaçao (Mestrado)—Universidade Estadual de Campinas, 2013.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/1403797.

Xing Gao and Guilin Li. A knn model based on manhattan distance to identify the snare proteins. *Ieee Access*, 8:112922–112931, 2020.

Xingyuan Huang, Yongjun Wang, Li Chao, Hui Xu, Qi Zhang, Leijing Yang, and Xiangjun Xin. Improved dbscan algorithm based signal recovery technology in coherent optical communication systems. *Optics Communications*, 521, 06 2022. doi: 10.1016/j.optcom.2022.128590.

K. Kikuchi. Digital coherent optical communication systems: Fundamentals and future prospects. *IEICE Electron*, 8:1642–1662, 2011.

Kazuro Kikuchi. Fundamentals of coherent optical fiber communications. *Journal of Lightwave Technology*, 34:157–179, 2016. doi: 10.1109/JLT.2015.2463719.

Ling Liu, Liangchuan Li, Yuanda Huang, Kai Cui, Qianjin Xiong, F.N. Hauske, Changsong Xie, and Yi Cai. Intra-channel nonlinearity compensation by inverse volterra series transfer function. *Journal of Lightwave Technology*, 30:310–316, 01 2011. doi: 10.1109/JLT.2011.2182038.

Xiang Liu, A. Chraplyvy, Peter Winzer, Robert Tkach, and Sethumadhavan Chandrasekhar. Phase-conjugated twin waves for communication beyond the kerr nonlinearity limit. *Nature Photonics*, 7: 560–568, 05 2013. doi: 10.1038/nphoton.2013.109.

Artur Marques. O meta-dataset amtriangle para experiências com supervised machine learning. *Revista da UI_IPSantarém*, 9(4):58–68, 2021.

Osmar Almeida Luz Neto. Caracterização sistêmica de um transmissor integro dp-qpsk. Master's thesis, São Carlos, SP, Brazil, 2014.

OIF. Implementation agreement 400zr, oif-400zr-01.0. Technical report, Optical Internetworking Forum (OIF), 2020.

OIF400ZR. Implementation Agreement 400ZR. [s.l: s.n.], 2024. Disponível em: https://www.oiforum.com/wp-content/uploads/OIF-400ZR-01.0_reduced2.pdf. Acesso em: 13 out. 2024.

Geraldo Antonio de Oliveira Junior. Compensação de não linearidades em redes pon coerentes de longa distância usando clusterização baseada em histograma. *Universidade Estadual Paulista (Unesp)*, 2020.

Augusto Duarte Pena et al. Pesos de hamming generalizados e códigos parametrizados. *Universidade Federal de Uberlândia*, 2018.

Katoh; K. Kikuch S. Tsukamoto, D.-S. Ly-Gagnon; K. Coherent demodulation of 40-gbit/s polarization-multiplexed qpsk signals with 16-ghz spacing after 200-km transmission. *Optical Fiber Communication Conf.*, pages 6–11, 2005.

Eric Sillekens, Wenting Yi, Daniel Semrau, Alessandro Ottino, Boris Karanov, Domanic Lavery, Lidia Galdino, Polina Bayvel, Robert I. Killey, Sujie Zhou, Kevin Law, and Jack Chen. Retropropagação digital aprendida no domínio do tempo. In *Workshop IEEE 2020 sobre Sistemas de Processamento de Sinais (SiPS)*, pages 1–4, 2020. doi: 10.1109/SiPS50750.2020.9195253.

Ozan K. Tonguz and Richard E. Wagner. Equivalence between preamplified direct detection and heterodyne receivers. *IEEE Photonics Technology Letters*, 3:835–837, 1991. URL `https://api.semanticscholar.org/CorpusID:30595331`.

Peter Winzer, David Neilson, and Andrew Chraplyvy. Fiber-optic transmission and networking: the previous 20 and the next 20 years [invited]. *Optics Express*, 26:24190, 08 2018. doi: 10.1364/OE.26.024190.

APÊNDICE A – CÓDIGOS GERADOS EM PYTHON

A.1 COMPENSAÇÃO DE EFEITOS NÃO LINEARES EM INTERCONECTORES ÓPTICOS UTILIZANDO DBSCAN

```
#%reset
%matplotlib qt

import pandas as pd                # For data processing
import numpy as np                 # For numerical calculations
import matplotlib.pyplot as plt    # For ploting and representation
from sklearn.cluster import DBSCAN

# Load ideal (reference) data
folder_name = "Data_175km"
df = pd.read_csv(folder_name+"/DP_IdealConstellationDiagram_0dbm.csv",
    skiprows=5)
data_ideal = np.round(2*df.to_numpy())
m,n = np.shape(data_ideal)

# Load real (distorted) data
data_real = np.zeros([16,m,n]) # Generate a matrix of zeros
# Populate the matrix with the distorted data
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_0dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[0,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_1dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[1,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_2dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[2,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_3dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[3,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_4dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
```

```
data_real[4,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_5dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[5,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_6dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[6,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_7dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[7,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_8dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[8,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_9dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[9,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_10dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[10,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_11dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[11,:,:] = data_real_aux
df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_12dbm.csv",
    skiprows=5)
data_real_aux = df.to_numpy()
data_real[12,:,:] = data_real_aux
#df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_13dbm.csv",
    skiprows=5)
#data_real_aux = df.to_numpy()
#data_real[13,:,:] = data_real_aux
#df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_14dbm.csv",
    skiprows=5)
#data_real_aux = df.to_numpy()
#data_real[14,:,:] = data_real_aux
#df = pd.read_csv(folder_name+"/DP_RealConstellationDiagram_15dbm.csv",
    skiprows=5)
```

```
#data_real_aux = df.to_numpy()
#data_real[15,:,:] = data_real_aux

print('The number of symbols is:',m,2**18)

%matplotlib qt
signal_ID_array = np.arange(4,13)
signal_ID = 10

#---------------- 0. Detection of the ideal symbols --------------------#

# Labels for the X polarization
aux = np.round((data_ideal[:,0]+3)/2)*4+np.round((data_ideal[:,1]+3)/2)
sym_X = aux.astype(int)
# Labels for the Y polarization
aux = np.round((data_ideal[:,2]+3)/2)*4+np.round((data_ideal[:,3]+3)/2)
sym_Y = aux.astype(int)

#---------------- 1. Generation of the labels and input matrices
    --------------------#

# Output for the X polarization
Y_X = data_ideal[:,:2]
Y_Y = data_ideal[:,2:]
Y = data_ideal[:,:]

# Create the input matrix:
#    data_X_I, data_X_Q, data_Y_I, data_Y_Q
X_X = data_real[signal_ID,:,:2]
X_Y = data_real[signal_ID,:,2:]
X = data_real[signal_ID,:,:]

#------------------ 2. Representation of the constellations
    ----------------------#
plot_flag = 1

# 2.1. Representation of all the points
if plot_flag == 1:
    plt.figure(1,figsize=(16, 6), dpi=80)
    plt.subplot(1,3,1)
    plt.plot(X_X[:,0],X_X[:,1],'.')
    plt.plot(X_Y[:,0],X_Y[:,1],'.')
    plt.axis('square')
```

```
105     #plt.xlabel('In-phase')
106     #plt.ylabel('Quadrature')
107     plt.xticks([-3,0,3])
108     plt.yticks([-3,0,3])
109     plt.title('Representacao de todos os pontos')
110 
111 # 2.2. Representation of the labeled points:
112 #print(sym_X)
113 if plot_flag == 1:
114     # X polarization
115     plt.subplot(1,3,2)
116     for ind in range(0,16):
117         ind_class = np.where(sym_X==ind)
118         #print(ind_class)
119         data_aux_I =  np.transpose(X_X[ind_class,0])
120         data_aux_Q =  np.transpose(X_X[ind_class,1])
121         plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
122         plt.title('Polarizacao x')
123     plt.axis('square')
124     #plt.xlabel('In-phase')
125     #plt.ylabel('Quadrature')
126     plt.xticks([-3,0,3])
127     plt.yticks([-3,0,3])
128 
129     # Y polarization
130     plt.subplot(1,3,3)
131     for ind in range(0,16):
132         ind_class = np.where(sym_Y==ind)
133         data_aux_I =  np.transpose(X_Y[ind_class,0])
134         data_aux_Q =  np.transpose(X_Y[ind_class,1])
135         plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
136         plt.title('Polarizacao Y')
137     plt.axis('square')
138     #plt.xlabel('In-phase')
139     #plt.ylabel('Quadrature')
140     plt.xticks([-3,0,3])
141     plt.yticks([-3,0,3])
142 
143 #--------- 3. Detection of each constellation using maximum likelihood
        ----------#
144 
145 print('ML detection')
146 
```

```
147 # 3.1. Classification of X polarization
148 X_X_dig = 4*np.clip(np.round((X_X[:,0]-1)/2)+2,0,3)+np.clip(np.round((X_X
        [:,1]-1)/2)+2,0,3)
149 sym_X_ML = X_X_dig.astype(int)
150 BER_X_ML = (sum(sym_X_ML != sym_X)/len(sym_X))/4
151 print('\t The BER of pol X using ML is:',BER_X_ML)
152
153 # 3.2. Classification of Y polarization
154 X_Y_dig = 4*np.clip(np.round((X_Y[:,0]-1)/2)+2,0,3)+np.clip(np.round((X_Y
        [:,1]-1)/2)+2,0,3)
155 sym_Y_ML = X_Y_dig.astype(int)
156 BER_Y_ML = (sum(sym_Y_ML != sym_Y)/len(sym_Y))/4
157 print('\t The BER of pol Y using ML is:',BER_Y_ML)
158
159 # 3.3. Averaging the two polarizations
160 BER_ML = (BER_X_ML+BER_Y_ML)/2
161 #BER_ML_array[counter] = BER_ML
162 print('\t The average BER using ML is:',BER_ML)
163
164
165 #---------------------- 4. Clusterization using DBSCAN
        -------------------------#
166 #Nsymbols_DBSCAN = 100000
167 Nsymbols_DBSCAN = 10000
168 X_X_reduced = X_X[:Nsymbols_DBSCAN,:]
169 sym_X_reduced = sym_X[:Nsymbols_DBSCAN]
170 #print(np.shape(X_X_reduced))
171
172 #DBSCAN_clustering = DBSCAN(eps=0.2, min_samples=5).fit(X_X_reduced)
173 #DBSCAN_clustering = DBSCAN(eps=0.2, min_samples=40).fit(X_X_reduced)
174 #DBSCAN_clustering = DBSCAN(eps=0.2, min_samples=80).fit(X_X_reduced)
175 #DBSCAN_clustering = DBSCAN(eps=0.4, min_samples=5).fit(X_X_reduced)
176 DBSCAN_clustering = DBSCAN(eps=0.4, min_samples=40).fit(X_X_reduced)
177 #DBSCAN_clustering = DBSCAN(eps=0.4, min_samples=80).fit(X_X_reduced)
178 #DBSCAN_clustering = DBSCAN(eps=0.6, min_samples=5).fit(X_X_reduced)
179 #DBSCAN_clustering = DBSCAN(eps=0.6, min_samples=40).fit(X_X_reduced)
180 #DBSCAN_clustering = DBSCAN(eps=0.6, min_samples=80).fit(X_X_reduced)
181
182 DBSCAN_labels = DBSCAN_clustering.labels_
183 #print(np.shape(DBSCAN_labels))
184 #print(DBSCAN_labels)
185 print('  Number of generated clusters:',max(DBSCAN_labels)+1)
186
```

```
## Grafico de haste do DBSCAN
plt.figure()
plt.stem(DBSCAN_labels)
plt.xlabel('Numero de simbolos')
plt.ylabel('Numero de camadas')
#plt.title('Grafico de haste do DBSCAN')



# Classificando os clusters
DBSCAN_labels_old = DBSCAN_labels
DBSCAN_labels = np.zeros(np.shape(DBSCAN_labels),dtype='int')
print(type(DBSCAN_labels[0]))
for ind in range(0,max(DBSCAN_labels_old+1)):
    ind_class = np.where(DBSCAN_labels_old==ind)
    class_value = int(np.median(sym_X_reduced[ind_class[0]]))
    DBSCAN_labels[ind_class] = class_value
ind_class = np.where(DBSCAN_labels_old==-1)
DBSCAN_labels[ind_class] = -1

# Representacao de cluster
plt.figure()
#plt.title('Representacao de cluster usando DBSCAN')
for ind in range(0,max(DBSCAN_labels+1)):
    ind_class = np.where(DBSCAN_labels==ind)
    data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
    data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
    plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
plt.axis('square')
# Representacao de pontos com ruido
ind_class = np.where(DBSCAN_labels==-1)
data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
plt.plot(data_aux_I,data_aux_Q,'.k',alpha=0.5)
plt.xticks([])  # Remove os rotulos do eixo x
plt.yticks([])  # Remove os rotulos do eixo y
#plt.figure()
#plt.plot(sym_X_reduced,DBSCAN_labels,'o')
BER_DBSCAN = np.mean(DBSCAN_labels != sym_X_reduced)/4
print('The BER of pol X using ML is:',BER_X_ML)
print('The BER of pol X using DBSCAN is:',BER_DBSCAN)


```

```
print('')
print('Summary of the DBSCAN clustering:')
print('  Number of generated clusters:',max(DBSCAN_labels)+1)
print('  Number of noisy points:',len(ind_class[0]))

#---------------------- 5. Applying KNN to refine the results
    -------------------------#
from sklearn.neighbors import KNeighborsClassifier

ind_noisy_points = np.where(DBSCAN_labels==-1)
ind_ref_points = np.where(DBSCAN_labels!=-1)
noisy_points = X_X_reduced[ind_noisy_points[0],:]
ref_points = X_X_reduced[ind_ref_points[0],:]
labels_ref_points = DBSCAN_labels[ind_ref_points[0]]

# Instance for KNN classifier
neigh = KNeighborsClassifier(n_neighbors=3)

# "Train" the KNN classifier
neigh.fit(ref_points, labels_ref_points)

# Find the labels for the noisy points
labels_noisy_points = neigh.predict(noisy_points)

# Build the new labels
DBSCAN_KNN_labels = np.zeros(np.shape(DBSCAN_labels),dtype='int')
DBSCAN_KNN_labels[ind_ref_points[0]] = labels_ref_points
DBSCAN_KNN_labels[ind_noisy_points[0]] = labels_noisy_points

BER_DBSCAN_KNN = np.mean(DBSCAN_KNN_labels != sym_X_reduced)/4
print('The BER of pol X using ML is:',BER_X_ML)
print('The BER of pol X using DBSCAN is:',BER_DBSCAN)
print('The BER of pol X using DBSCAN+KNN is:',BER_DBSCAN_KNN)

# Graphical respresentation

# DBSCAN representation
plt.figure()
for ind in range(0,max(DBSCAN_labels+1)):
    ind_class = np.where(DBSCAN_labels==ind)
    data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
    data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
    plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
```

```
plt.axis('square')
ind_class = np.where(DBSCAN_labels==-1)
data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
plt.plot(data_aux_I,data_aux_Q,'.k',alpha=0.5)

#plt.title('Representacao de cluster usando DBSCAN')


# DBSCAN representation
plt.figure()
for ind in range(0,max(DBSCAN_KNN_labels+1)):
    ind_class = np.where(DBSCAN_KNN_labels==ind)
    data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
    data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
    plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
plt.axis('square')
#plt.title('Representacao de cluster usando DBSCAN + KNN')
plt.xticks([])  # Remove os rotulos do eixo x
plt.yticks([])  # Remove os rotulos do eixo y

from sklearn.cluster import KMeans

%matplotlib qt
signal_ID_array = np.arange(4,13)
#signal_ID_array = np.arange(4,15)



BER_ML_array = np.zeros(len(signal_ID_array))
BER_DBSCAN_array = np.zeros(len(signal_ID_array))
BER_DBSCAN_KNN_array = np.zeros(len(signal_ID_array))
BER_kmeans_array = np.zeros(len(signal_ID_array))


for counter, signal_ID in enumerate(signal_ID_array):
    print('Processing',counter+1,'of',len(signal_ID_array))

    #---------------- 0. Detection of the ideal symbols
       --------------------#

    # Labels for the X polarization
```

```
313     aux = np.round((data_ideal[:,0]+3)/2)*4+np.round((data_ideal[:,1]+3)
            /2)
314     sym_X = aux.astype(int)
315     # Labels for the Y polarization
316     aux = np.round((data_ideal[:,2]+3)/2)*4+np.round((data_ideal[:,3]+3)
            /2)
317     sym_Y = aux.astype(int)
318
319     #---------------- 1. Generation of the labels and input matrices
            --------------------#
320
321     # Output for the X polarization
322     Y_X = data_ideal[:,:2]
323     Y_Y = data_ideal[:,2:]
324     Y = data_ideal[:,:]
325
326     # Create the input matrix:
327     #    data_X_I, data_X_Q, data_Y_I, data_Y_Q
328     X_X = data_real[signal_ID,:,:2]
329     X_Y = data_real[signal_ID,:,2:]
330     X = data_real[signal_ID,:,:]
331
332     #------------------ 2. Representation of the constellations
            ----------------------#
333     plot_flag = 0
334
335     # 2.1. Representation of all the points
336     if plot_flag == 1:
337         plt.figure(1,figsize=(16, 6), dpi=80)
338         plt.subplot(1,3,1)
339         plt.plot(X_X[:,0],X_X[:,1],'.')
340         plt.plot(X_Y[:,0],X_Y[:,1],'.')
341         plt.axis('square')
342         #plt.xlabel('In-phase')
343         #plt.ylabel('Quadrature')
344         plt.xticks([-3,0,3])
345         plt.yticks([-3,0,3])
346
347
348     # 2.2. Representation of the labeled points:
349     #print(sym_X)
350     if plot_flag == 1:
351         # X polarization
```

```
352         plt.subplot(1,3,2)
353         for ind in range(0,16):
354             ind_class = np.where(sym_X==ind)
355             #print(ind_class)
356             data_aux_I =  np.transpose(X_X[ind_class,0])
357             data_aux_Q =  np.transpose(X_X[ind_class,1])
358             plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
359         plt.axis('square')
360         #plt.xlabel('In-phase')
361         #plt.ylabel('Quadrature')
362         plt.xticks([-3,0,3])
363         plt.yticks([-3,0,3])
364
365
366         # Y polarization
367         plt.subplot(1,3,3)
368         for ind in range(0,16):
369             ind_class = np.where(sym_Y==ind)
370             data_aux_I =  np.transpose(X_Y[ind_class,0])
371             data_aux_Q =  np.transpose(X_Y[ind_class,1])
372             plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
373         plt.axis('square')
374         #plt.xlabel('In-phase')
375         #plt.ylabel('Quadrature')
376         plt.xticks([-3,0,3])
377         plt.yticks([-3,0,3])
378
379
380     #--------- 3. Deteccao de cada constelacao usando maxima
            verossimilhanca ----------#
381
382     # print('   ML detection')
383
384
385     # 3.2. Classification of Y polarization
386     X_Y_dig = 4*np.clip(np.round((X_Y[:,0]-1)/2)+2,0,3)+np.clip(np.round
            ((X_Y[:,1]-1)/2)+2,0,3)
387     sym_Y_ML = X_Y_dig.astype(int)
388     BER_Y_ML = (sum(sym_Y_ML != sym_Y)/len(sym_Y))/4
389     print('\t The BER of pol Y using ML is:',BER_Y_ML)
390
391     # 3.3. Averaging the two polarizations
392     BER_ML = (BER_X_ML+BER_Y_ML)/2
```

```
    #BER_ML_array[counter] = BER_ML
    print('\t The average BER using ML is:',BER_ML)

    BER_ML_array[counter] = BER_ML

    #---------------------- 4. Clustering using DBSCAN
       -------------------------#
    #Nsymbols_DBSCAN = 50000
    Nsymbols_DBSCAN = 100000
    X_X_reduced = X_X[:Nsymbols_DBSCAN,:]
    sym_X_reduced = sym_X[:Nsymbols_DBSCAN]
    #print(np.shape(X_X_reduced))

    #DBSCAN_clustering = DBSCAN(eps=0.2, min_samples=5).fit(X_X_reduced)
    #DBSCAN_clustering = DBSCAN(eps=0.2, min_samples=40).fit(X_X_reduced)
    #DBSCAN_clustering = DBSCAN(eps=0.1, min_samples=80).fit(X_X_reduced)
    DBSCAN_clustering = DBSCAN(eps=0.1, min_samples=60).fit(X_X_reduced)
    #DBSCAN_clustering = DBSCAN(eps=0.4, min_samples=40).fit(X_X_reduced)
    DBSCAN_labels = DBSCAN_clustering.labels_
    #print(np.shape(DBSCAN_labels))
    #print(DBSCAN_labels)

    ## Stem plot of the DBSCAN
    #plt.figure()
    #plt.stem(DBSCAN_labels)

    # Sorting out the clusters
    DBSCAN_labels_old = DBSCAN_labels
    DBSCAN_labels = np.zeros(np.shape(DBSCAN_labels),dtype='int')
    print(type(DBSCAN_labels[0]))
    for ind in range(0,max(DBSCAN_labels_old+1)):
        ind_class = np.where(DBSCAN_labels_old==ind)
        class_value = int(np.median(sym_X_reduced[ind_class[0]]))
        DBSCAN_labels[ind_class] = class_value
    ind_class = np.where(DBSCAN_labels_old==-1)
    DBSCAN_labels[ind_class] = -1

    # Cluster representation
    plt.figure()
    for ind in range(0,max(DBSCAN_labels+1)):
        ind_class = np.where(DBSCAN_labels==ind)
        data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
        data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
```

```
435         plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
436     plt.axis('square')
437 
438     # Noisy points representation
439     ind_class = np.where(DBSCAN_labels==-1)
440     data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
441     data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
442     plt.plot(data_aux_I,data_aux_Q,'.k',alpha=0.5)
443 
444 
445     #plt.figure()
446     #plt.plot(sym_X_reduced,DBSCAN_labels,'o')
447     BER_DBSCAN = np.mean(DBSCAN_labels != sym_X_reduced)/4
448 
449     #print('The BER of pol X using ML is:',BER_X_ML)
450     print('The BER of pol X using DBSCAN is:',BER_DBSCAN)
451 
452     print('')
453     print('Summary of the DBSCAN clustering:')
454     print('  Number of generated clusters:',max(DBSCAN_labels)+1)
455     print('  Number of noisy points:',len(ind_class[0]))
456 
457     BER_DBSCAN_array[counter] = BER_DBSCAN
458 
459     # ---------------------- 5. Applying KNN to refine the results
        -------------------------#
460 
461     ind_noisy_points = np.where(DBSCAN_labels==-1)
462     ind_ref_points = np.where(DBSCAN_labels!=-1)
463     noisy_points = X_X_reduced[ind_noisy_points[0],:]
464     ref_points = X_X_reduced[ind_ref_points[0],:]
465     labels_ref_points = DBSCAN_labels[ind_ref_points[0]]
466 
467     # Instance for KNN classifier
468     neigh = KNeighborsClassifier(n_neighbors=3)
469 
470     # "Train" the KNN classifier
471     neigh.fit(ref_points, labels_ref_points)
472 
473     # Find the labels for the noisy points
474     labels_noisy_points = neigh.predict(noisy_points)
475 
476     # Build the new labels
```

```
    DBSCAN_KNN_labels = np.zeros(np.shape(DBSCAN_labels),dtype='int')
    DBSCAN_KNN_labels[ind_ref_points[0]] = labels_ref_points
    DBSCAN_KNN_labels[ind_noisy_points[0]] = labels_noisy_points

    BER_DBSCAN_KNN = np.mean(DBSCAN_KNN_labels != sym_X_reduced)/4
    print('The BER of pol X using DBSCAN+KNN is:',BER_DBSCAN_KNN)

    BER_DBSCAN_KNN_array[counter] = BER_DBSCAN_KNN


     # ---------------------- ---------------------------------------- #
    # DBSCAN representation
    plt.figure()
    for ind in range(0,max(DBSCAN_KNN_labels+1)):
        ind_class = np.where(DBSCAN_KNN_labels==ind)
        data_aux_I =  np.transpose(X_X_reduced[ind_class,0])
        data_aux_Q =  np.transpose(X_X_reduced[ind_class,1])
        plt.plot(data_aux_I,data_aux_Q,'.',alpha=0.5)
    plt.axis('square')
    plt.title('Representacao de cluster usando DBSCAN + KNN')
    plt.xticks([])  # Remove os rotulos do eixo x
    plt.yticks([])  # Remove os rotulos do eixo y

    # ---------------------- 6. Applying K-means
       ---------------------------------------- #
    X_X_reduced = X_X[:Nsymbols_DBSCAN,:]
    sym_X_reduced = sym_X[:Nsymbols_DBSCAN]
    kmeans_clustering = KMeans(n_clusters=16, random_state=0).fit(
       X_X_reduced)
    kmeans_labels = kmeans_clustering.labels_

    # Sorting out the clusters
    kmeans_labels_old = kmeans_labels
    kmeans_labels = np.zeros(np.shape(kmeans_labels),dtype='int')
    print(type(DBSCAN_labels[0]))
    for ind in range(0,max(DBSCAN_labels_old+1)):
        ind_class = np.where(kmeans_labels_old==ind)
        class_value = int(np.median(sym_X_reduced[ind_class[0]]))
        kmeans_labels[ind_class] = class_value
    BER_kmeans = np.mean(kmeans_labels != sym_X_reduced)/4
    print('The BER of pol X using KMEANS is:',BER_kmeans)

    BER_kmeans_array[counter] = BER_kmeans
```

```
518
519     plt.figure()
520 P_array = np.arange(4,13)
521 print(P_array)
522 plt.semilogy(P_array,BER_ML_array, label = 'Maxima verossimilhanca')
523 plt.semilogy(P_array,BER_DBSCAN_array, label = 'DBSCAN')
524 plt.semilogy(P_array,BER_DBSCAN_KNN_array, label = 'DBSCAN + KNN')
525 plt.semilogy(P_array,BER_kmeans_array,label ='Kmeans')
526 plt.ylabel('BER')
527 plt.xlabel('POWER(dB)')
528 plt.legend()
```