Parte IV: Aprendizado de Máquina.

Prof. Fabiano Araujo Soares, Dr. / FGA 0221 - Inteligência Artificial

Universidade de Brasília

2025

O que é Aprendizado Não Supervisionado?

Definição:

- Treinar modelos com dados **sem rótulos**
- Descobrir **estruturas internas**
- Sem "gabarito" fornecido
- Aprender **representações úteis**

Entrada vs. Saída:

- **Entrada:** Apenas dados X
- **Saída:** Clusters, padrões, redução de dimensionalidade
- **Objetivo:** Organizar/compreender dados

Diferença Fundamental

Supervisionado: $(X, y) \rightarrow$ Predição
Não Supervisionado: $X \rightarrow$ Estrutura

Importância dos Métodos Não Supervisionados

Desafios práticos:

- Rotular dados é **caro** e **lento**
- Dados não rotulados **abundantes**
- Estrutura desconhecida **a priori**
- Exploração de dados complexos

Benefícios:

- Descoberta automática de padrões
- Preparação para aprendizado supervisionado
- Detecção de anomalias
- Compressão e interpretabilidade

Exemplo Real

Segmentação de clientes em milhões de registros de compra sem categorias pré-definidas

Quando Usar Aprendizado Não Supervisionado?

Cenário	**Aplicação**
Exploração inicial	Entender estrutura e características dos dados
Segmentação	Agrupar usuários, produtos, imagens
Redução de dimensionalidade	Comprimir características, visualizar dados
Detecção de anomalias	Identificar comportamentos anormais
Pré-processamento	Preparar dados para supervisionado
Extração de features	Aprender representações úteis

Decisão: Use não supervisionado quando dados são abundantes, mas rótulos são escassos ou desconhecidos.

Princípios Fundamentais

1. **Similaridade:** Objetos similares devem estar próximos
2. **Compactação:** Minimizar variação intra-cluster
3. **Separação:** Maximizar distância inter-cluster
4. **Esparsidade:** Destacar estrutura principal (redução de dimensões)
5. **Regularização:** Evitar overfitting em dados não etiquetados

Métrica Central: Função de Objetivo Sem Rótulos

Maximizar compacidade (nível de compactação) e separação simultaneamente:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \boldsymbol{\mu}_i\|^2$$

onde $\boldsymbol{\mu}_i$ é o centroide do cluster i

Categorias Principais de Algoritmos

1. **Clustering (Agrupamento)**
   - K-Means
   - DBSCAN
   - Hierarchical Clustering
   - GMM (Gaussian Mixture Models)
2. **Redução de Dimensionalidade**
   - PCA (Principal Component Analysis)
   - t-SNE
   - UMAP
   - Autoencoders
3. **Detecção de Anomalias**
   - Isolation Forest
   - Local Outlier Factor
   - Autoencoders
4. **Aprendizado de Representação**
   - Autoencoders
   - Variational Autoencoders (VAE)
   - Contrastive Learning

K-Means: Algoritmo Clássico de Clustering

Princípio: Dividir dados em k clusters minimizando distância intra-cluster

Algoritmo:

1. Inicializar k centróides aleatoriamente
2. Atribuir cada ponto ao centróide mais próximo
3. Recalcular centróides como média dos pontos no cluster
4. Repetir até convergência

Função Objetivo:

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \boldsymbol{\mu}_i\|^2$$

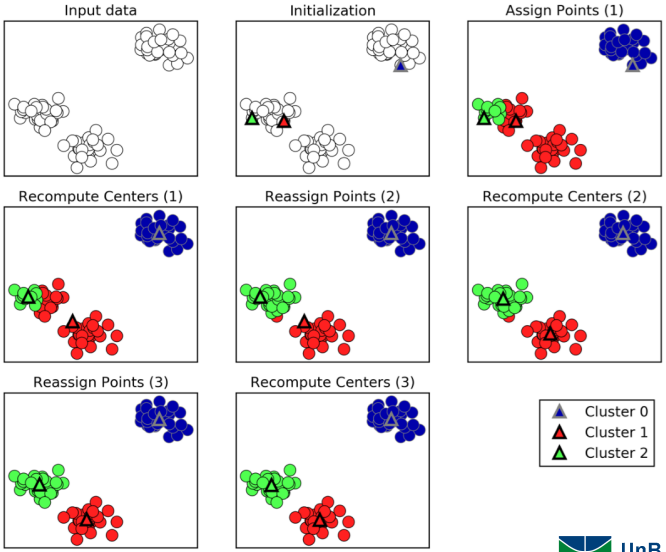
Vantagens:

- Rápido, escalável
- Fácil de implementar
- Interpretável

Limitações:

- Requer k pré-definido
- Sensível a inicialização
- Assume clusters esféricos

K-Means: Exemplo



DBSCAN: Clustering Baseado em Densidade

Por que DBSCAN?

- Clusters reais nem sempre têm formas esféricas ou tamanhos semelhantes (problema do K-Means).
- Desejamos identificar regiões de alta densidade separadas por regiões de baixa densidade.
- Além disso, identificar pontos que não pertencem a nenhum grupo: **outliers**.

Ideia-chave:

- Uma região suficientemente densa de pontos é um cluster.
- Ruídos e pontos isolados não são agrupados.

Funcionamento do Algoritmo DBSCAN

Etapas principais:

1. Para cada ponto, encontre todos os vizinhos dentro do raio ϵ.
2. Se houver pelo menos MinPts vizinhos, criar/expandir um cluster iniciando desse ponto (**core point**).
3. Todos os vizinhos densamente conectados são adicionados ao cluster (processo recursivo).
4. Pontos com menos de MinPts, mas conectados a um core point, são **border points**.
5. Pontos que não são alcançáveis por clusters são classificados como **outliers**.

Obs: O processo repete até todos os pontos serem classificados.

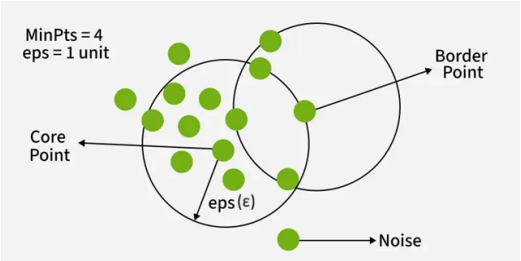
Intuição Visual e Parâmetros do DBSCAN

Visualização dos Pontos:

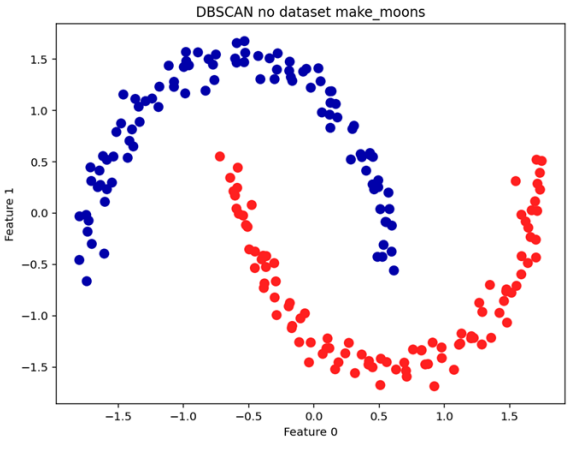
- regiões de alta densidade = clusters;
- pontos isolados = outliers.

Escolha dos parâmetros:

- ϵ define o "tamanho" do cluster local: muito pequeno separa demais, muito grande une clusters diferentes.
- MinPts define a "densidade" necessária: valores típicos 4–6.



Intuição Visual e Parâmetros do DBSCAN



Resumo e Aplicações do DBSCAN

Principais pontos recapitulados:

- Algoritmo eficiente para identificar clusters de formas arbitrárias.
- Não precisa do número de clusters a priori.
- Robusto a ruídos e outliers.

Aplicações:

- Detecção de anomalias em dados financeiros.
- Agrupamento de pontos geográficos.
- Identificação de padrões em imagens.

PCA (Principal Component Analysis)

O problema da Alta Dimensionalidade:

- Dados com 100+ atributos são **caros computacionalmente**
- Difícil visualizar e interpretar
- Ruído se amplifica em muitas dimensões
- Overfitting em algoritmos supervisionados

A ideia do PCA:

- Nem todas as dimensões são igualmente importantes
- Algumas direções concentram **mais variação** dos dados
- Essas direções são as **componentes principais**
- Podemos descartar dimensões com pouca variação (ruído)

Exemplo: Dados 2D espalhados principalmente ao longo de uma linha diagonal

- Podemos projetar tudo em 1D (a linha) sem perder informação essencial

Algoritmo PCA - Passo a Passo

Etapa 1: Centralizar os dados

- Subtrair a média: $X_{centered} = X - \boldsymbol{\mu}$
- Garante que o algoritmo não é enviesado por deslocamento

Etapa 2: Calcular matriz de covariância

- $C = \frac{1}{n} X_{centered}^T X_{centered}$
- Mede como as dimensões variam **juntas**
- Matriz simétrica de tamanho $d \times d$ (onde d = número de atributos)

Etapa 3: Encontrar autovalores e autovetores

- Resolver: $Cv = \lambda v$
- **Autovalores** λ: magnitude da variação em cada direção
- **Autovetores** v: direções (componentes principais)
- Ordenar por λ decrescente

Etapa 4: Projetar nos k primeiros autovetores

- $X_{pca} = X_{centered} V_k$ (selecionar k componentes)
- Reduz de d dimensões para k dimensões

Escolhendo o Número de Componentes

Variância Explicada por Componente:

- Proporção: $\text{Var}_i = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$
- Componentes com λ grandes capturam mais informação
- Descartar componentes com λ pequenos = descartar ruído

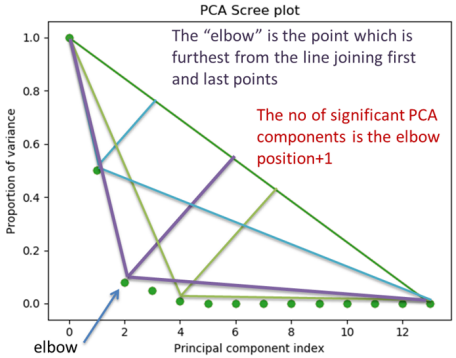
Regra prática para escolher k:

- **Visualização:** Escolher $k = 2$ ou $k = 3$ (gráficos 2D/3D)
- **Preservar 95% de variância:** Encontrar k mínimo tal que $\sum_{i=1}^{k} \text{Var}_i \geq 0.95$
- **Cotovelo (Elbow):** Plotar variância acumulada vs. k e procurar onde a curva "dobra"

Escolhendo o Número de Componentes

Visualização do Cotovelo:

- Eixo X: número de componentes k
- Eixo Y: variância acumulada (%)
- Ponto de "cotovelo"= k ideal



Limitações do PCA e Alternativas

Limitações do PCA:

- **Assume linearidade:** Dados não-lineares não são bem reduzidos
- Exemplo: Dados em espiral 3D $\Rightarrow$ PCA não "desenrola"bem
- **Interpretabilidade:** Componentes principais não têm significado intuitivo

Alternativas não-lineares:

t-SNE

- Preserva proximidades locais
- Ótimo para visualização (2D/3D)
- Lento em dados grandes

UMAP

- Mais rápido que t-SNE
- Preserva estrutura local e global
- Melhor para dados multidimensionais

Recomendação

Use PCA para compressão rápida; use t-SNE/UMAP se dados têm estrutura não-linear complexa

Resumo: PCA em Aplicações Reais

Quando usar PCA:

- **Visualização:** Reduzir dados 50+ dimensões para 2D/3D, etc.
- **Pré-processamento:** Remover ruído antes de classificação/clustering
- **Detecção de anomalias:** Reconstruir dados e detectar alta perda
- **Compressão:** Armazenar menos dados sem perder informação essencial

Exemplos de Aplicação:

- **Análise de imagens:** Reduzir de milhões de pixels para 100 componentes
- **Dados genômicos:** Reduzir 20.000+ genes para componentes interpretáveis
- **Séries temporais:** Comprimir histórico financeiro antes de LSTM

Dica importante:

- **Sempre padronizar** dados antes de PCA (média 0, desvio 1)
- Não padronizar $\Rightarrow$ atributos com escalas grandes dominam

Autoencoders

O problema:

- PCA funciona bem, mas é LINEAR
- Muitos dados reais têm estruturas não-lineares complexas
- Precisamos de um método que aprenda representações não-lineares automaticamente

A ideia dos Autoencoders:

- Rede neural que **aprende a comprimir dados por si mesma**
- Sem rótulos: aprende apenas vendo os dados
- Captura padrões complexos (não-lineares)
- Usa redes neurais profundas para melhor representação

Analogia:

- Um **fotógrafo** tira a foto (Encoder: dados $\rightarrow$ representação)
- Depois **reconstrói** a cena (Decoder: representação $\rightarrow$ dados)

Arquitetura dos Autoencoders

Componentes principais:

1. **ENCODER (Compressão):**
   - Entrada: X (Dimensionalidade alta, ex: 784 pixels)
   - Camadas ocultas reduzem tamanho progressivamente
   - Saída: z (latent code, ex: 32 dimensões)
   - **Aprendizado:** Quais características são importantes?

2. **DECODER (Reconstrução):**
   - Entrada: z (representação comprimida)
   - Camadas ocultas aumentam de tamanho progressivamente
   - Saída: $\hat{X}$ (reconstrução de X)
   - **Objetivo:** Reconstruir original fielmente

O BOTTLENECK (gargalo):

- Camada z é muito menor que X
- Força a rede a aprender apenas o **essencial**

Como o Autoencoder Aprende?

Processo de Treinamento:

1. **Forward Pass:**
   - Entrada X passa pelo Encoder $\rightarrow$ obtém z
   - z passa pelo Decoder $\rightarrow$ obtém $\hat{X}$
2. **Calcular Perda (Loss):**
   - Comparar X (original) com $\hat{X}$ (reconstruído)
   - $L = \|X - \hat{X}\|^2$ (erro quadrático)
   - Se L é pequeno $\Rightarrow$ boa compressão!
3. **Backpropagation:**
   - Atualizar pesos para **minimizar** a perda
   - Encoder aprende a extrair características importantes
   - Decoder aprende a reconstruir bem

Resultado após treinamento:

- Encoder: ferramenta para **extrair features** (redução de dimensionalidade)
- Decoder: geralmente descartado

Tipos de Autoencoders

1. **Autoencoder Simples (Vanilla):**
   - Encoder e Decoder simétricos
   - Objetivo: compressão e redução de ruído
   - Exemplo: $784 \rightarrow 256 \rightarrow 32 \rightarrow 256 \rightarrow 784$
2. **Denoising Autoencoder (DAE):**
   - Entrada: dados com **ruído adicionado**
   - Saída: dados **limpos**
   - Aprende a remover ruído automaticamente
3. **Variational Autoencoder (VAE):**
   - Adiciona distribuição probabilística no latent space
   - Permite **gerar novos dados** (não apenas reconstruir)
   - Mais complexo, mais poderoso
4. **Convolutional Autoencoder (CAE):**
   - Usa camadas convolucionais (para imagens)
   - Melhor captura de padrões espaciais
   - Eficiente em altas dimensões

Aplicações Práticas dos Autoencoders

1. **Detecção de Anomalias:**
   - Treinar com dados **normais** apenas
   - Se $\|X - \hat{X}\|$ é alto $\Rightarrow$ anomalia!
   - Exemplo: Fraude bancária, detecção de intrusão
2. **Compressão de Imagens:**
   - Reduzir tamanho de arquivo sem perder muita qualidade
   - Melhor que JPEG em alguns cenários
   - Exemplo: Armazenamento na nuvem
3. **Feature Learning (Aprendizado de Representação):**
   - Extrair características automaticamente (sem engenharia manual)
   - Usar o Encoder como pré-processador para algoritmos supervisionados
   - Exemplo: Preparar dados para classificação
4. **Remoção de Ruído:**
   - Treinar com pares: dados ruidosos $\rightarrow$ dados limpos
   - Aprende a filtrar ruído automaticamente
   - Exemplo: Recuperação de imagens antigas

Autoencoder vs PCA: Quando usar cada um?

PCA:

- Rápido, eficiente
- Interpretável
- Linear
- Poucos dados
- Estruturas não-lineares

Autoencoder:

- Não-linear, flexível
- Dados complexos
- Anomalias
- Mais lento
- Precisa de muitos dados

Regra de decisão:

- Poucos dados (<10k amostras), estrutura linear $\Rightarrow$ **PCA**
- Muitos dados (>100k), estrutura complexa $\Rightarrow$ **Autoencoder**
- Amostras muito grandes (imagens, áudio) $\Rightarrow$ **Convolutional AE**
- Precisa gerar novos dados $\Rightarrow$ **VAE**

Resumo: Autoencoders

Principais Conceitos:

- Rede neural que **aprende a comprimir dados automaticamente**
- **Encoder**: extrai features ($X \rightarrow z$)
- **Decoder**: reconstrói dados ($z \rightarrow \hat{X}$)
- **Bottleneck**: camada comprimida (gargalo)
- Treina minimizando $L = \|X - \hat{X}\|^2$ (Erro quadrático médio)

Vantagens:

- Captura estruturas não-lineares (vs PCA)
- Feature learning automático (sem engenharia manual)
- Detecção de anomalias
- Aplicações em imagens, áudio, séries temporais

Exemplos Práticos: Segmentação de Clientes

Cenário: Loja online com milhões de transações

Dados: Frequência de compra, valor gasto, categorias preferidas, localização

Algoritmo: K-Means com $k = 4$

Resultado:

- **Cluster 1:** Clientes ativos, alto valor (VIP)
- **Cluster 2:** Clientes moderados, valor médio
- **Cluster 3:** Clientes ocasionais, baixo valor
- **Cluster 4:** Clientes inativos (churn risk)

Ação de Negócio:

- Cluster 1 $\rightarrow$ Programas VIP exclusivos
- Cluster 4 $\rightarrow$ Campanhas de reativação

Impacto: Personalização automática, aumento de retenção

Exemplos Práticos: Detecção de Anomalias

Cenário: Monitoramento de transações bancárias

Dados: Valor, horário, localização, comerciante, histórico

Algoritmo: Autoencoder

Funcionamento:

- Treinar modelo em transações "normais"
- Reconstrução alta = normal
- Reconstrução baixa ou distância alta = anomalia

Exemplos de Anomalias Detectadas:

- Transação em país diferente 2 horas depois
- Valor $>$ 10x da média histórica
- Múltiplas tentativas falhas

Impacto: Fraude bloqueada em tempo real, sem falsos positivos excessivos

Exemplos Práticos: Compressão de Imagens

Cenário: Armazenar e transmitir imagens de alta resolução

Dados: Imagens $256 \times 256 \times 3$ (196.608 dimensões)

Algoritmo: Autoencoder ou PCA

Processo:

1. Encoder reduz para z de 32 dimensões
2. Decoder reconstrói imagem próxima à original
3. Taxa de compressão: $\approx 6000 : 1$

Resultado:

- Original: 50 MB
- Comprimido: $\approx$ 8 KB
- Qualidade visual: 95% da original

Impacto: Redução de armazenamento, transmissão mais rápida

Exemplos Práticos: Análise de Textos

Cenário: Organizar milhões de artigos sem categorias pré-definidas

Dados: Documentos em formato de vetores de palavras (TF-IDF ou embeddings)

Algoritmo: DBSCAN

Resultado:

- Cluster 1: Notícias sobre esportes
- Cluster 2: Artigos sobre tecnologia
- Cluster 3: Publicações de economia
- Outliers: Documentos mistos ou únicos

Pós-Processamento:

- Usar clusters como treino para classificador supervisionado
- Recomendação de artigos similares
- Detecção de conteúdo fraudulento ou fake news

Impacto: Organização automática, descoberta de tópicos, recomendação

Resumo: Aprendizado Não Supervisionado

Principais Pontos:

1. Descobre estruturas em dados **sem rótulos**
2. Essencial quando rotulação é cara ou impossível
3. Três categorias: Clustering, Redução de Dimensionalidade, Detecção de Anomalias
4. Algoritmos principais: K-Means, DBSCAN, PCA, Autoencoders
5. Aplicações reais: Segmentação, Detecção, Compressão, Exploração

Próxima Aula:

- Aprendizado por reforço!

Referências

- Russell, S., Norvig, P., "Artificial Intelligence: A Modern Approach", 4th ed., Person, 2022.
- Muller, A. C., Guido, S., "Introduction to Machine Learning with Python A Guide for Data Scientists", O'Reilly, 2017.

Obrigado!

E-mail: fabianosoares@unb.br

LinkedIn: https://www.linkedin.com/in/fabiano-soares-06b6a821a/

Site do curso: https://www.fabianosoares.eng.br/fga0221-inteligência-artificial