Transformada de Foley-Sammon Nula para reconhecimento facial

A **Transformada Nula de Foley-Sammon (NFST)** é uma técnica avançada de redução dimensional e análise discriminativa utilizada, principalmente, em problemas de reconhecimento facial e classificação multiclasse em espaços de alta dimensionalidade.

Objetivo Principal

O objetivo da NFST é encontrar um subespaço discriminativo em que as amostras pertencentes à mesma classe (por exemplo, imagens da mesma pessoa) fiquem próximas entre si, enquanto as amostras de classes diferentes estejam bem separadas. Essa abordagem busca superar limitações das técnicas tradicionais como a Análise Discriminante Linear (LDA), especialmente em situações onde o número de amostras é menor que a dimensionalidade dos dados — o chamado problema da alta dimensionalidade com poucos exemplos.

Fundamentação Teórica

Na NFST, consideram-se duas matrizes principais de dispersão:

- **Matriz de dispersão total Sx**: quantifica a variabilidade total dos dados.
- **Matriz de dispersão dentro da classe Sw**: quantifica a variabilidade das amostras dentro da mesma classe.

Diferentemente da LDA tradicional, que resolve um problema de autovalores para maximizar a razão entre a dispersão entre classes e dentro da classe, a NFST foca em encontrar o **espaço nulo de SwS_wSw** intersectado com o espaço de linha de Sx. Ou seja, procura o subespaço onde a dispersão dentro das classes é anulada (espaço nulo de Sw) e que ainda contenha toda a informação relevante (espaço de linha de Sx).

Matematicamente, isso é expresso como:

$$W = \text{interseção}\big(\mathcal{R}(S_x), \mathcal{N}(S_w)\big)$$

onde R indica o espaço de linha e N o espaço nulo (kernel).

Este subespaço W possui a propriedade desejada de maximizar a discriminabilidade entre as classes, ao mesmo tempo em que elimina variações internas irrelevantes.

Método Computacional

Para encontrar o subespaço W, a NFST utiliza a Decomposição em Valores Singulares (SVD) das matrizes Xt e Xw, que representam os dados centralizados globalmente e por classe, respectivamente.

1. A SVD de Xt fornece a base ortonormal do espaço de linha de Sx.
2. A interseção com o espaço nulo de Sw é calculada através da resolução do espaço nulo da projeção de XwT na base obtida.
3. O subespaço Wé obtido como uma combinação linear das bases ortonormais, resultando em um espaço de dimensionalidade reduzida, tipicamente c−1, onde c é o número de classes.

Vantagens da NFST

- **Resolução do problema da singularidade**: Em problemas de alta dimensionalidade com poucas amostras, as matrizes de dispersão são geralmente singulares, o que dificulta técnicas como a LDA. A NFST contorna essa dificuldade utilizando o espaço nulo de Sw.
- **Melhor capacidade discriminativa**: Ao focar na interseção entre o espaço nulo e o espaço de linha, a NFST maximiza a separabilidade das classes.
- **Redução eficaz da dimensionalidade**: Produz um subespaço compacto que mantém as características essenciais para a classificação.

Aplicação no Reconhecimento Facial

No projeto, a NFST foi aplicada para reduzir a dimensionalidade das imagens faciais e melhorar a separação entre as classes (pessoas diferentes). Após calcular o subespaço discriminativo, as imagens foram projetadas nesse espaço para serem classificadas usando o método do vizinho mais próximo, obtendo assim uma maior precisão no reconhecimento.

No reconhecimento facial, as imagens são representadas por vetores de alta dimensão, correspondentes aos pixels das imagens. Esse alto número de dimensões dificulta o processamento e a classificação, além de incluir variações que não ajudam a distinguir pessoas, como iluminação, pose e expressão.

A NFST busca reduzir essa dimensionalidade, preservando as características que discriminam diferentes classes (pessoas), ao mesmo tempo em que minimiza as variações internas de cada classe.

Como funciona a NFST?

A técnica baseia-se em duas matrizes principais:

- **Matriz de dispersão total (Sx)**: representa a variabilidade total dos dados.
- **Matriz de dispersão dentro da classe (Sw)**: representa a variabilidade dos dados dentro de cada classe (variações internas).

A NFST busca o subespaço que é a interseção do espaço de linha da matriz Sx e o espaço nulo (kernel) da matriz Sw. Em termos práticos, isso significa encontrar um espaço onde as diferenças entre classes são mantidas, enquanto as diferenças internas são anuladas.

Para isso, a técnica utiliza a decomposição em valores singulares (SVD) para extrair bases ortonormais desses espaços, possibilitando o cálculo do subespaço discriminativo.

Aplicação no projeto

No projeto, o processo foi o seguinte:

1. **Carregamento e seleção dos dados**: utilizamos imagens do conjunto LFW, reduzindo para duas classes para simplificar o problema.
2. **Pré-processamento**: as imagens foram transformadas em vetores e centralizadas, calculando a média global e a média por classe para construir as matrizes Xt e Xw.
3. **Cálculo do subespaço discriminativo**: aplicamos a SVD para obter a base do espaço de linha da matriz Xt e, em seguida, identificamos o subespaço que está no espaço nulo de Sw.

4. **Projeção dos dados**: as imagens foram projetadas nesse subespaço, reduzindo a dimensionalidade e destacando as características discriminativas.
5. **Classificação**: utilizamos o método de vizinho mais próximo (k-NN) para classificar as imagens projetadas, avaliando a acurácia do modelo.

Benefícios da NFST

- **Redução da dimensionalidade**: diminui a complexidade computacional e o ruído nos dados.
- **Melhora a separação entre classes**: facilita a distinção entre diferentes indivíduos.
- **Minimiza variações internas**: reduz os efeitos de variações como iluminação e expressão facial.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split

# --- Tarefa 1: Carregar dados ---
# Carregar o dataset LFW com pelo menos 70 imagens por pessoa
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

print(f"Número total de imagens: {lfw_people.images.shape[0]}")
print(f"Dimensão das imagens: {lfw_people.images.shape[1:]}")
print(f"Número de pessoas (classes): {len(lfw_people.target_names)}")

# --- Tarefa 2: Analisar dados ---
num_amostras = lfw_people.data.shape[0]
num_recursos = lfw_people.data.shape[1]
num_classes = len(lfw_people.target_names)

print(f"Número total de amostras: {num_amostras}")
print(f"Número de recursos (pixels): {num_recursos}")
```

```
print(f"Número de classes-alvo (pessoas): {num_classes}")

# --- Tarefa 3: Visualizar dados carregados ---
num_imagens = 3
indices_aleatorios = np.random.choice(lfw_people.images.shape[0], num_ima

fig, axes = plt.subplots(1, num_imagens, figsize=(12, 4))
for i, idx in enumerate(indices_aleatorios):
    ax = axes[i]
    ax.imshow(lfw_people.images[idx], cmap='gray')
    ax.set_xlabel(lfw_people.target_names[lfw_people.target[idx]])
    ax.axis('off')
plt.tight_layout()
plt.show()

# --- Tarefa 4: Carregar dados seletivos (duas classes) ---
# Selecionar as duas primeiras pessoas para simplificar
pessoas_selecionadas = lfw_people.target_names[:2]

lfw_two_classes = fetch_lfw_people(min_faces_per_person=70, resize=0.4,
                                   classes=pessoas_selecionadas)

X = lfw_two_classes.data  # dados (n_samples x n_features)
y = lfw_two_classes.target  # rótulos inteiros

print(f"Número de amostras selecionadas: {X.shape[0]}")
print(f"Classes selecionadas: {pessoas_selecionadas}")
print(f"Rótulos únicos: {np.unique(y)}")

# --- Tarefa 5: Divisão treino/teste ---
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"Tamanho treino: {X_train.shape[0]} amostras")
print(f"Tamanho teste: {X_test.shape[0]} amostras")

# --- Tarefa 6: Calculando Matrizes de Dispersão ---
```

```
def getXwXt(X, y):
    """
    Recebe:
        X: matriz (n_samples x n_features)
        y: vetor rótulos inteiros

    Retorna:
        Xw: matriz (d x n) dos dados centralizados por classe, concatenados
        Xt: matriz (d x n) dos dados centralizados globalmente
    """
    X = X.T  # d x n
    media_global = np.mean(X, axis=1, keepdims=True)
    Xt = X - media_global

    classes = np.unique(y)
    Xw_list = []

    for c in classes:
        Xc = X[:, y == c]
        media_classe = np.mean(Xc, axis=1, keepdims=True)
        Xw_c = Xc - media_classe
        Xw_list.append(Xw_c)

    Xw = np.hstack(Xw_list)
    return Xw, Xt

Xw, Xt = getXwXt(X_train, y_train)
print(f"Shape Xt: {Xt.shape}")
print(f"Shape Xw: {Xw.shape}")

# --- Tarefa 7: Computando o subespaço discriminativo ---
def compute_subspace(Xw, Xt, c):
    r_Xt = np.linalg.matrix_rank(Xt)
    print(f"Rank de Xt: {r_Xt}")

    U, S, Vt = np.linalg.svd(Xt, full_matrices=False)
    U_r = U[:, :r_Xt]
    print(f"Shape U_r: {U_r.shape}")
```

```
    A = Xw.T @ U_r
    Sb, Sigma_b, Bt = np.linalg.svd(A, full_matrices=False)

    Bc_minus_1 = Bt[-(c-1):, :].T
    print(f"Shape Bc_minus_1: {Bc_minus_1.shape}")

    W = U_r @ Bc_minus_1
    print(f"Shape W: {W.shape}")

    return W

c = len(np.unique(y_train))
W = compute_subspace(Xw, Xt, c)

# --- Tarefa 8: Projeção do subespaço ---
X_train_p = (W.T @ X_train.T).T
X_test_p = (W.T @ X_test.T).T

print(f"Shape X_train_p: {X_train_p.shape}")
print(f"Shape X_test_p: {X_test_p.shape}")

# --- Tarefa 9: Avaliação no conjunto de teste ---
def knn_predict(X_train_p, y_train, X_test_p):
    y_pred = []
    for x_test in X_test_p:
        distancias = np.linalg.norm(X_train_p - x_test, axis=1)
        indice_vizinho = np.argmin(distancias)
        y_pred.append(y_train[indice_vizinho])
    return np.array(y_pred)

y_pred = knn_predict(X_train_p, y_train, X_test_p)
acertos = np.sum(y_pred == y_test)
precisao = acertos / len(y_test)
print(f"Acurácia no teste: {precisao:.4f} ({acertos} de {len(y_test)})")

# --- Tarefa 10: Visualizar previsões ---
indices_aleatorios = np.random.choice(X_test.shape[0], 6, replace=False)
```

```
fig, axes = plt.subplots(2, 3, figsize=(15, 8))
for i, idx in enumerate(indices_aleatorios):
    ax = axes.flat[i]
    imagem = X_test[idx].reshape(lfw_two_classes.images.shape[1], lfw_two_cl
    ax.imshow(imagem, cmap='gray')

    real = lfw_two_classes.target_names[y_test[idx]]
    previsto = lfw_two_classes.target_names[y_pred[idx]]
    cor = 'green' if y_pred[idx] == y_test[idx] else 'red'

    ax.set_xlabel(f"Real: {real}")
    ax.set_ylabel(f"Previsto: {previsto}", color=cor)
    ax.axis('off')

plt.tight_layout()
plt.show()
```

- **Carrega imagens de faces do dataset LFW** com pessoas que tenham pelo menos 70 imagens no conjunto, para garantir dados suficientes por pessoa.
- **Analisa os dados carregados**, mostrando quantas imagens, dimensões e classes (pessoas) existem.
- **Visualiza algumas imagens aleatórias** do dataset para ter uma ideia do conteúdo.
- **Seleciona apenas duas classes (duas pessoas)** para simplificar o problema.
- **Divide os dados em treino e teste** para avaliar a performance do modelo.
- **Calcula as matrizes de dispersão** dos dados — isto é, dados centralizados por classe e globalmente.
- **Computa um subespaço discriminativo** (uma transformação linear que ajuda a separar as classes) usando decomposição SVD, parecido com o método LDA (Linear Discriminant Analysis).
- **Projeta os dados no subespaço calculado** para redução dimensional e melhor separação.

- **Faz predição usando K-Nearest Neighbors (KNN)** no espaço projetado para classificar as imagens do conjunto de teste.
- **Avalia e mostra a acurácia final** do classificador.
- **Visualiza algumas imagens do teste mostrando a previsão e o rótulo verdadeiro**, colorindo o texto de verde quando a previsão está correta e vermelho quando está errada.