UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

EA006: TRABALHO DE FIM DE CURSO

RELATÓRIO FINAL

Ajuste automático de hiperparâmetros do algoritmo t-SNE para visualização de dados de elevada dimensão

Guilherme Duarte Marmerola RA: 117097

Orientador: Prof. Dr. Fernando José Von Zuben

20 de Junho de 2017

Abstract. *t-Distributed Stochastic Neighbor Embedding (t-SNE) é uma técnica de redução da dimensionalidade não linear que tem atraído a atenção da academia nos últimos anos, particularmente em aplicações envolvendo dados não-estruturados e aprendizado profundo (deep learning) [Mnih et al. 2015, Maaten and Hinton 2008]. A técnica se propõe a resolver o problema de mapear dados representados por centenas, milhares ou até milhões de dimensões em um espaço de baixa dimensionalidade. Em geral, o mapeamento gerado é de duas ou três dimensões, de forma a permitir a visualização dos dados em gráficos de dispersão. Apesar do grande sucesso obtido pelo método na tarefa proposta, ainda existem dificuldades no ajuste dos seus parâmetros e na interpretação dos mapas de baixa dimensionalidade produzidos [Wattenberg et al. 2016]. Neste trabalho, propomos utilizar metodologias de otimização de hiperparâmetros conhecidas (Random Search, TPE e Processos Gaussianos) para configurar o t-SNE com o objetivo de gerar visualizações otimizadas. Mostramos resultados em seis conjuntos de dados de alta dimensionalidade sintéticos e reais, realizando análises quantitativas baseadas em três métricas de avaliação, além de uma análise qualitativa, mostrando que as visualizações geradas mostram mais claramente a estrutura intrínseca dos dados.*

1. Introdução e objetivos

Um dos avanços mais relevantes na área de redução da dimensionalidade e visualização de dados nos últimos anos veio com a proposta do t-SNE (t-Distributed Sthochastic Neighbor Embedding) [Maaten and Hinton 2008, Van Der Maaten 2014], um algoritmo que mapeia dados de alta dimensionalidade para um espaço de baixa dimensionalidade (geralmente 2D ou 3D) de forma a preservar a vizinhança dos objetos originais. A metodologia se mostrou superior na resolução de diversos problemas, e tem tido uma adoção significativa na comunidade de ciência de dados[1].

Sendo um método não paramétrico e não linear, o t-SNE é de difícil compreensão e ajuste de hiperparâmetros. Os hiperparâmetros disponíveis permitem o controle de diversos aspectos do modelo: (a) número efetivo de vizinhos próximos, (b) taxa de aprendizado, (c) número máximo de iterações, (d) fator multiplicativo de "exagero" na fase inicial de otimização e (e) nível de aproximação numérica nos cálculos (aproximação de Barnes-Hut). Além disso, antes de serem processados pelo t-SNE, os dados geralmente são submetidos a uma redução de dimensionalidade linear (por meio de PCA) para algumas centenas ou dezenas de dimensões, com o objetivo de suprimir eventuais ruídos e melhorar a eficiência dos cálculos de similaridade entre os objetos.

Mesmo diante do grande número de hiperparâmetros e da complexidade da sua interpretação, os autores reivindicam a robustez do algoritmo, argumentando que este é, de uma maneira geral, insensível a mudanças em sua configuração. Algumas boas práticas de utilização foram publicadas tanto pelos autores do algoritmo[2] quanto por engenheiros de software que o implementaram[3]. Porém, outros trabalhos desafiaram estas premissas,

[1] http://blog.kaggle.com/2012/11/02/t-distributed-stochastic-neighbor-embedding-wins-merck-viz-challenge/

[2] https://lvdmaaten.github.io/tsne/

[3] http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

mostrando casos em que os hiperparâmetros têm uma influência bastante pronunciada no resultado final do mapeamento [Wattenberg et al. 2016].

O ajuste de hiperparâmetros em problemas de aprendizado de máquina é estudado por uma disciplina de pesquisa que tem visto bastante atividade no passado recente: a de Configuração de Algoritmos. Nesta disciplina de pesquisa, são estudados problemas de otimização em que o espaço de busca é desafiador, formado por uma combinação de variáveis discretas, contínuas e condicionais (dependem de combinações de outras variáveis). Além disso, a configuração de algoritmos em geral define problemas de otimização em que a função-objetivo não é linear, e não há garantia de que ela seja determinística, convexa e diferenciável. Finalmente, a avaliação de tal função-objetivo geralmente tem um alto custo computacional, portanto é de grande importância encontrar uma solução adequada no menor número de iterações possível.

Alguns trabalhos recentes na área têm avançado o estado da arte. Em [Bergstra and Bengio 2012], é proposto o uso da busca aleatória ao invés de busca em grid e ajuste manual em situações em que relativamente poucos hiperparâmetros são importantes (mas estes não são conhecidos). Em trabalhos seguintes, são propostas duas metodologias para escolher o conjunto de hiperparâmetros mais promissor com base em configurações anteriores: TPE (Tree-structured Parzen Estimator) e Processos Gaussianos [Bergstra et al. 2011, Bergstra et al. 2013b]. A metodologia de Processos Gaussianos estima diretamente $P(y|x)$ onde y é o resultado da simulação de um algoritmo configurado com hiperparâmetros x. Por outro lado, TPE estima $P(x|y)$ e $P(y)$ separadamente. Tais metodologias, com exceção da busca aleatória, tentam aproximar a função-objetivo real através de uma simulação de menor custo computacional de avaliação, para então buscar nesta simulação pelo melhor conjunto de parâmetros possível.

Neste relatório, mostramos resultados quantitativos e qualitativos que suportam a utilização de metodologias de otimização de hiperparâmetros para configurar o t-SNE, objetivando construir mapeamentos de duas dimensões que melhor representem a estrutura de dados em alta dimensionalidade. Realizamos experimentos considerando três métricas de avaliação (*consistência de agrupamento*, *estrutura global* e *divergência KL final*) e seis conjuntos de dados. Além disso, evidenciamos que, apesar de os parâmetros recomendados em [Maaten and Hinton 2008] serem em geral robustos, é possível obter melhorias nas visualizações considerando espaços de busca de hiperparâmetros mais extensos do que o usual.

2. Métodos e algoritmos

Neste trabalho, temos como objetivo encontrar uma metodologia de ajuste automático dos parâmetros do t-SNE para a produção de visualizações de dados otimizadas. Para realizar tal tarefa, utilizaremos três algoritmos de busca de hiperparâmetros: Busca aleatória, TPE e Processos Gaussianos. Nesta seção, descrevemos estes algoritmos, além de realizar uma breve introdução ao t-SNE.

2.1. t-Distributed Stochastic Neighbor Embedding (t-SNE)

Nesta seção, é introduzido o principal algoritmo em análise neste trabalho, t-Distributed Stochastic Neighbor Embedding (t-SNE). Inicialmente, descrevemos a intuição que motivou a construção do algoritmo; depois, comentamos seu processo de otimização e o impacto dos hiperparâmetros disponíveis neste processo.

2.1.1. Intuição

Proposto inicialmente em [Hinton and Roweis 2002], o algoritmo Stochastic Neighbor Embedding (SNE) propõe uma abordagem probabilística ao problema de reduzir a dimensionalidade de objetos complexos. Intuitivamente, é de interpretação relativamente simples. O algoritmo inicialmente constrói uma quantidade p_{ij}, definida como a probabilidade de um objeto i escolher outro objeto j como seu vizinho no espaço de alta dimensionalidade.

$$p_{ij} = \frac{exp(-d_{ij}^2)}{\sum_{k \neq i} exp(-d_{ik}^2)}$$

As variáveis d_{ij}^2 representam as dissimilaridades entre os objetos. Quando não especificado de outra maneira, elas são calculadas como uma distância euclidiana ajustada (*affinity*).

$$d_{ij}^2 = \frac{||x_i - x_j||^2}{2\sigma_i^2}$$

O valor de σ_i é ajustado indiretamente através de um hiperparâmetro chamado de *perplexidade* (k). É imposta a condição de que a entropia da distribuição acima seja igual a $\log k$. Portanto, σ_i é encontrado de forma a satisfazer esta condição.

No espaço de baixa dimensionalidade, também é criada uma probabilidade de vizinhança q_{ij} que mede quão provável um determinado objeto de baixa dimensionalidade y_i escolha outro y_j como seu vizinho.

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

Em um mapa de baixa dimensionalidade que capture corretamente a vizinhança dos pontos no mapa de alta dimensionalidade, as probabilidades p_{ij} e q_{ij} são aproximadamente iguais. Portanto, o SNE posiciona os pontos no mapa de baixa dimensionalidade de forma a minimizar a divergência de Kullback-Leibler entre as distribuições $P_i = p_{ij}$ e $Q_i = q_{ij}$:

$$C = \sum_i KL(P_i||Q_i)$$

Depois, em [Maaten and Hinton 2008], foi criada uma variação do SNE que acabou se tornando um dos algoritmos mais usados para visualização de dados complexos na atualidade, o t-SNE. Esta variação realiza duas melhorias em comparação ao SNE: (a) utiliza uma versão simétrica da função-objetivo do SNE, simplificando os gradientes e (b) ao invés de uma gaussiana, utiliza uma distribuição de t-Student para modelar q_{ij}, resolvendo um problema do SNE chamado de *crowding problem*, onde diversos pontos se concentram no centro do mapa de baixa dimensionalidade.

2.1.2. Prática

Na prática, o t-SNE realiza o posicionamento dos objetos no espaço de baixa dimensionalidade em duas etapas:

1. **Early exaggeration:** na primeira fase de otimização do algoritmo, todos os valores p_{ij} são multiplicados por uma constante de "exagero" $\mathcal{E}$. Isso faz com que inicialmente os objetos formem agrupamentos concentrados e bastante separados entre si. Dessa forma, é permitido um movimento mais livre dos grupos no mapa (nenhum grupo se coloca entre outros dois mais similares), evitando-se mínimos locais.
2. **Gradient descent:** o fator de exagero é suprimido e a otimização segue como descrito anteriormente. De maneira similar a outros algoritmos de aprendizado de máquina, podemos controlar a taxa de aprendizado (η) e o número máximo de iterações ($\mathcal{N}$) do processo de otimização.

Na versão mais moderna do t-SNE, temos a opção de realizar uma aproximação numérica dos gradientes através de variações do algoritmo de Barnes-Hut, tornando sua complexidade $\mathcal{O}(NlogN)$ ao invés de $\mathcal{O}(N^2)$. Podemos controlar o compromisso entre exatidão numérica e eficiência computacional através do parâmetro de *trade-off* θ.

Finalmente, uma prática popular na aplicação do t-SNE é de inicialmente reduzir a dimensionalidade dos dados para algumas dezenas ou centenas de dimensões através de uma decomposição linear (como PCA), suprimindo ruído nos dados e simplificando o cálculo das similaridades entre os objetos. Nos nossos experimentos, propomos usar esta técnica, tratando o número de componentes n e a opção de *whitening* (para componentes de variância unitária e descorrelacionados) f_w como hiperparâmetros.

2.1.3. Síntese dos hiperparâmetros

Nas subseções anteriores, apresentamos a intuição sobre a qual o t-SNE foi construído, uma breve descrição do processo de otimização do algoritmo e os hiperparâmetros pertinentes à nossa análise. Na Tabela 1, sintetizamos cada um deles e seu papel no algoritmo.

Hiperparâmetro	Notação	Descrição
Perplexidade **[perplexity]**	k	Número efetivo de vizinhos próximos dos objetos
Fator de exagero **[early_exaggeration]**	$\mathcal{E}$	Controla quão comprimidos serão os clusters no início da otimização (evita mínimos locais)
Taxa de aprendizado **[learning_rate]**	η	Tamanho do passo de otimização
Número máximo de iterações **[n_iter]**	$\mathcal{N}$	Define um critério de parada para o processo de otimização
Fator de *trade-off* **[angle]**	θ	Controla o nível de aproximação numérica no t-SNE de Barnes-Hut
Número de dimensões (PCA) **[pca_dims]**	n	Define o número de dimensões da etapa preliminar de redução da dimensionalidade
Whitening **[whitening_flag]**	f_w	Flag que ativa ou não o processo de descorrelacionar e impor variância unitária aos componentes

Tabela 1. Resumo dos hiperparâmetros do t-SNE

Neste trabalho, utilizamos a implementação do t-SNE disponível no módulo `scikit-learn` [Pedregosa et al. 2011] para a linguagem `Python`.

2.2. Busca aleatória [rand]

A busca aleatória é uma técnica de meta-modelagem bastante simples e poderosa. Basicamente, a técnica consiste em definir um espaço de busca de hiperparâmetros através de distribuições de probabilidade, e a cada iteração configurar o algoritmo (ou simulação) de interesse a partir de amostras destas distribuições. Em [Bergstra and Bengio 2012], é apresentada a tese de que poucos hiperparâmetros são realmente importantes na configuração de um algoritmo, especialmente para casos em que há dezenas ou centenas deles. Contudo, diferentes parâmetros são importantes para diferentes conjuntos de dados. Com esse cenário em mente, foram comparados busca em grid, ajuste manual e busca aleatória em diversos experimentos, onde a última se mostrou mais eficiente, já que explora um espaço maior e carrega menor viés de construção que as outras técnicas.

Desta forma, utilizamos a busca aleatória como uma base de comparação natural com as outras metodologias. Foi utilizada a implementação disponível no módulo `hyperopt` [Bergstra et al. 2013a], para a linguagem `Python`.

2.3. Processos Gaussianos [GP]

Um processo gaussiano pode ser visto como uma generalização de uma distribuição normal multivariada para infinitas dimensões. Para os nossos propósitos, podemos interpretá-lo como uma metodologia de interpolação não linear dotada de cálculo de incertezas. Com o GP, podemos mapear o resultado de avaliações da função de interesse, interpolando estes resultados no espaço dos hiperparâmetros. À medida que nos distanciamos dos pontos

conhecidos, temos uma maior incerteza acerca da estimativa da função produzida. Na Figura 1, mostramos um exemplo de ajuste de processo gaussiano, evidenciando a incerteza estimada. Com base na interpolação produzida e nas estimativas de incerteza, podemos empregar uma estratégia para explorar o espaço e encontrar os melhores candidatos para a escolha de hiperparâmetros.

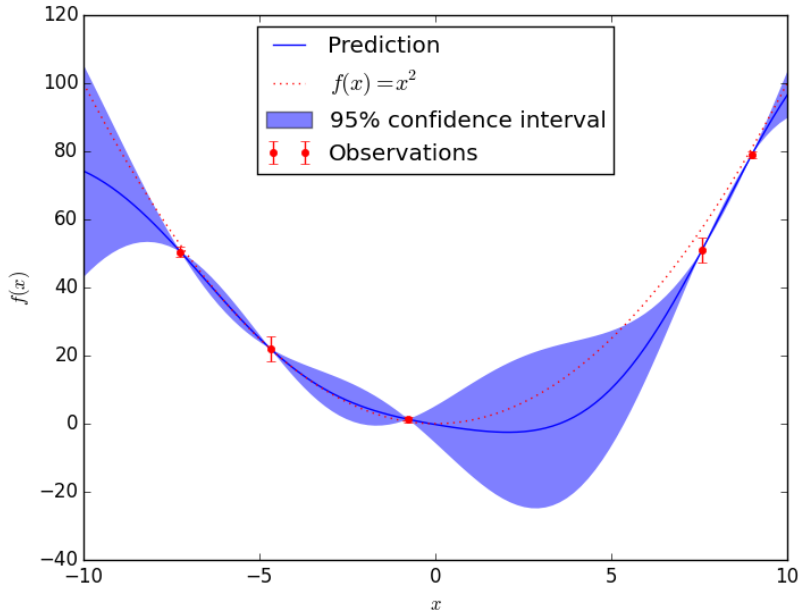

Figura 1. Exemplo de ajuste de processo gaussiano. É criada uma interpolação entre os pontos conhecidos, além de uma estimativa de banda de incerteza.

Existem diversos critérios para a escolha de novas amostras (funções de aquisição), tanto baseados no problema de *multi-armed bandits* quanto em noções estatísticas mais tradicionais: *expected improvement*, *probability of improvement*, *upper confidence bound*, *thompson sampling*, entre outros. Inicialmente, definimos nossa estratégia de otimização como *upper confidence bound* (UCB). Neste critério, queremos escolher a cada rodada a configuração que nos dê a melhor estimativa otimista de performance do nosso algoritmo. Para maior detalhe sobre o método, favor se referir ao trabalho em [Auer 2002].

Apesar de se mostrar como uma solução elegante para o problema da configuração de algoritmos, a metodologia baseada em processos gaussianos tem suas desvantagens. Primeiramente, ela não é bem adaptada para lidar com parâmetros categóricos (temos que gerar um processo gaussiano para cada uma das classes possíveis), além de ter seus próprios hiperparâmetros (como a escolha da função de aquisição, por exemplo). Além disso, tem uma complexidade cúbica em relação ao número de pontos e linear em relação ao número de variáveis, não sendo adequada para situações de alta dimensionalidade.

Neste trabalho, utilizamos o otimizador baseado em processos gaussianos disponível no módulo `BayesianOptimization`[4], construído sobre os alicerces disponíveis na biblioteca `scikit-learn` [Pedregosa et al. 2011]. Ambas as bibliotecas, assim como

[4]https://github.com/fmfn/BayesianOptimization

foi o caso da busca aleatória, são implementadas em `Python`. A função de aquisição escolhida é implementada na opção `ucb` da biblioteca. O algoritmo foi configurado para explorar mais do que tirar proveito, com `kappa` (*exploration/exploitation tradeoff*) igual a 10.0

2.4. Tree-structured Parzen Estimator [`tpe`]

O TPE integra a classe de algoritmos de estimativa de densidade de kernel (KDE). Como na busca aleatória, inicialmente devemos definir distribuições de probabilidade para os hiperparâmetros considerados (distribuição a priori). O algoritmo começa com uma pequena busca aleatória para coletar um conjunto inicial de rodadas de otimização. Depois, os dados das rodadas desta primeira etapa são divididos em dois grupos: (1) melhores tentativas (tipicamente entre 10-25% melhores) e (2) outras tentativas. O objetivo do algoritmo é encontrar um conjunto de parâmetros que tenha uma maior probabilidade de gerar um experimento pertencente ao grupo (1). Mostramos um exemplo na Figura 2, retirado de um artigo relacionado à otimização de redes neurais[5].

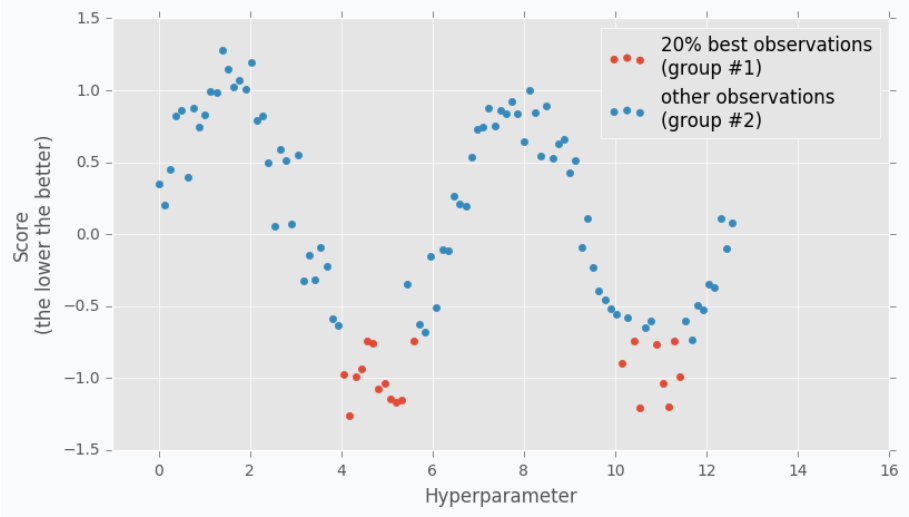

Figura 2. Exemplo de divisão de grupos pelo TPE, com um corte de 20%.

Seja $l(x)$ a probabilidade de um conjunto de parâmetros gerar um experimento com resultado no grupo (1) e $g(x)$ no grupo (2). A melhora esperada (*expected improvement*) na rodada atual pode ser calculada como:

$$EI(x) = \frac{l(x)}{g(x)}$$

No exemplo mostrado na Figura 3, é ilustrado o cálculo da melhora esperada. Porém, ao contrário de distribuições t-Student como mostrado na figura, são utilizadas estimativas de densidade baseadas em janelas de Parzen [Parzen 1962].

Cada hiperparâmetro possui uma estimativa de probabilidade associada à melhora diferente, organizada em uma hierarquia de árvore, acomodando bem variáveis categóricas e condicionais. A grande desvantagem do TPE é selecionar parâmetros independentemente,

[5]http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html

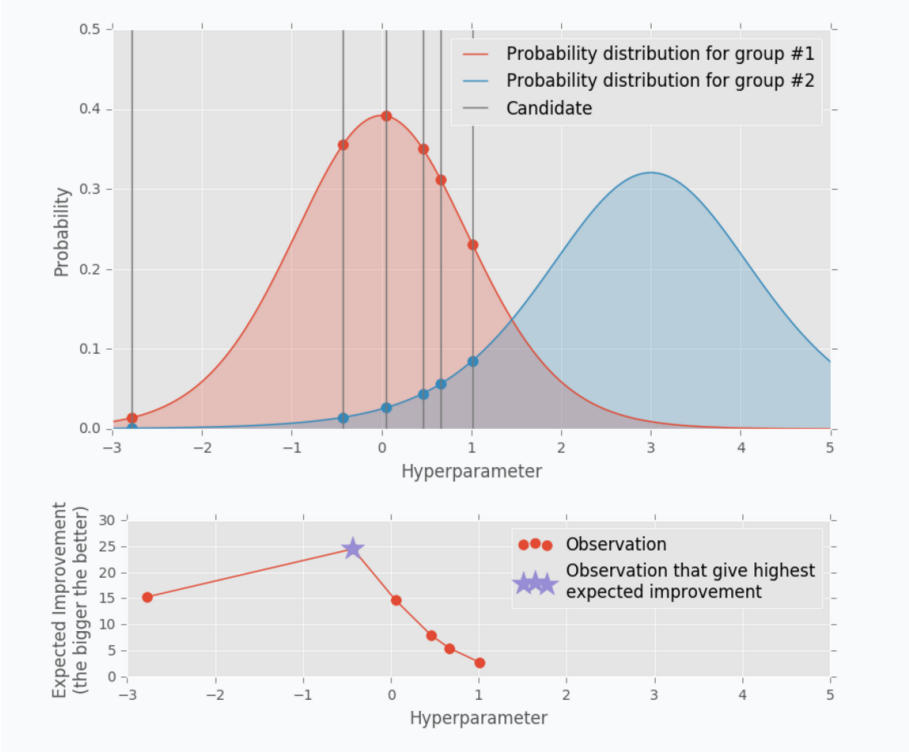

Figura 3. Exemplo de cálculo da melhora esperada pelo TPE.

deixando o trabalho de codificar relações mais fortes entre eles (como regularização e número de épocas de treino) para um especialista. Dado que tal conhecimento esteja disponível, é possível expressá-lo no espaço de busca do algoritmo (se existir regularização, usar uma distribuição que permita mais épocas de treino, por exemplo).

Nos nossos experimentos, utilizamos a implementação do TPE disponível no módulo `hyperopt` para `Python`. Configurações padrões foram utilizadas.

3. Abordagem experimental

Nesta seção, descrevemos os conjuntos de dados considerados para o trabalho e as métricas de avaliação utilizadas. O direcionamento do trabalho foi largamente baseado nos resultados de [Wattenberg et al. 2016]. Foram identificados alguns pontos principais onde podemos explorar as metodologias de configuração de algoritmos para otimizar as visualizações geradas pelo t-SNE:

- **A estrutura global nem sempre é mantida.** Uma das características do t-SNE é de precisar de um ajuste fino dos parâmetros para modelar a geometria global com maior fidelidade. Dessa forma, visamos usar as metodologias de configuração de algoritmos para otimizar alguma noção de geometria global dos grupos.
- **O t-SNE pode gerar visualizações com clusters bem definidos a partir de ruído.** Em alguns casos de teste mostrados em [Wattenberg et al. 2016], o t-SNE agrupou pontos aleatoriamente distribuídos, algo não desejável na prática. Portanto, um experimento interessante é de otimizar a representação para minimizar tal comportamento. Neste trabalho, implementamos este objetivo por meio de uma métrica de *consistência de agrupamento*: o resultado de um agrupamento

no espaço de alta dimensionalidade deve ser o mesmo que no espaço de baixa dimensionalidade.

- **O t-SNE pode ter sua convergência dificultada por mínimos locais.** O t-SNE tem uma tendência comum a encontrar mínimos locais. Os autores sugerem, inclusive, executar o algoritmo múltiplas vezes e adotar o melhor resultado em termos da divergência de Kullback-Leibler (KL). Estudamos qual é o efeito dos hiperparâmetros (com a exceção da perplexidade, que deve permanecer fixa) nos valores finais da divergência KL em múltiplas execuções, tentando otimizar essa métrica.

Objetivamos realizar otimizações de hiperparâmetros em todas as combinações possíveis de metodologias de otimização, conjuntos de dados e métricas de avaliação: (3 metodologias de otimização) $\times$ (3 métricas) $\times$ (6 conjuntos de dados) = 54 experimentos.

Nas próximas subseções, evidenciamos como pretendemos capturar matematicamente as propriedades discutidas por meio de métricas de avaliação, que servirão como objetivo da otimização de hiperparâmetros. Além disso, descrevemos os dados de teste sintéticos e reais propostos para executar nossos experimentos. Finalmente, mostramos uma visão geral do nosso *pipeline* de otimização.

3.1. Métricas de avaliação

Para traduzir em termos matemáticos as nossas intenções de otimização, propomos três métricas de avaliação da qualidade das visualizações geradas pelas diferentes configurações do t-SNE. As métricas objetivam capturar os seguintes aspectos dos mapas produzidos pelo algoritmo: (a) representação de estrutura global, (b) consistência de agrupamento e (c) otimização da divergência KL.

3.1.1. Representação de estrutura global (*rank-order correlation*)

Para capturar a qualidade da representação da estrutura global dos dados pelo t-SNE, temos que comparar as distâncias entre os grupos formados no mapa de baixa dimensionalidade e os já existentes. Porém, sabemos que não é possível representar distâncias com fidelidade em dimensionalidade reduzida. Dessa forma, propomos otimizar uma métrica de *ranking*: a **ordem** entre os grupos mais próximos no espaço de baixa dimensionalidade deve ser a mesma que no espaço original. Para isso, utilizamos a medida de *rank-order correlation*[6], que é basicamente o coeficiente de Spearman entre duas listas ordenadas. Para definir quais são os grupos, realizamos um agrupamento por meio do algoritmo HDBSCAN [Campello et al. 2013] no espaço de alta dimensionalidade (o mesmo agrupamento é mantido no espaço original e reduzido). Mostramos um exemplo na Figura 4.

No exemplo mostrado através da Tabela 2, podemos observar que a ordem de proximidade entre os grupos foi alterada pelo mapeamento em baixa dimensionalidade. Os grupos são definidos pela aplicação do HDBSCAN[7] no espaço de alta dimensionalidade.

[6]https://en.wikipedia.org/wiki/Spearman's_rank_correlation_coefficient

[7]Em todos os casos, o HDBSCAN foi aplicado com um tamanho de cluster mínimo de 10 e parâmetro `min_samples` igual a 1.

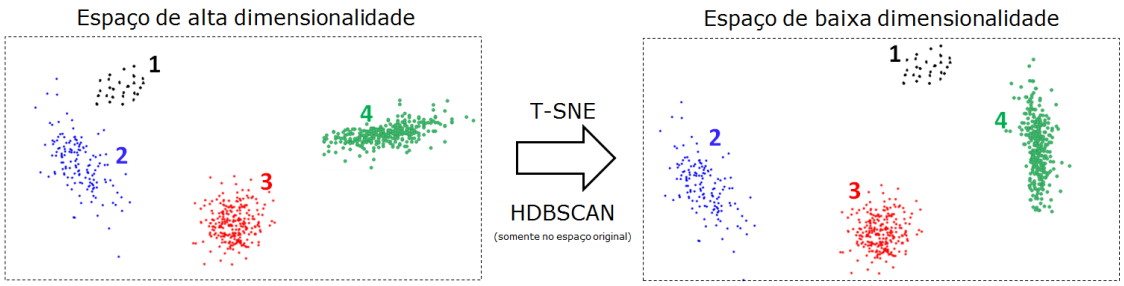

Figura 4. Exemplo de variação de estrutura global após aplicação do t-SNE.

Grupo 1			**Grupo 2**		
Ordem	Esp. Original	Esp. Reduzido	Ordem	Esp. Original	Esp. Reduzido
1	Grupo 2	Grupo 4	1	Grupo 1	Grupo 3
2	Grupo 3	Grupo 3	2	Grupo 3	Grupo 1
3	Grupo 4	Grupo 2	3	Grupo 4	Grupo 4
Grupo 3			**Grupo 4**		
Ordem	Esp. Original	Esp. Reduzido	Ordem	Esp. Original	Esp. Reduzido
1	Grupo 2	Grupo 2	1	Grupo 3	Grupo 1
2	Grupo 4	Grupo 4	2	Grupo 1	Grupo 3
3	Grupo 1	Grupo 1	3	Grupo 2	Grupo 2

Tabela 2. Comparação de ordem de proximidade entre clusters no espaço original e reduzido.

Para medir a assertividade entre os rankings de distâncias, computamos o coeficiente de Spearman entre a ordem de proximidade de cada grupo e todos os outros (*rank-order correlation*). Mostramos o cálculo na Tabela 3. Para uma métrica única de estrutura global M_G, fazemos a média de todos os coeficientes:

$$M_G = \frac{1}{N}\sum_{i=1}^{N} Spearman(Grupo_i)$$

No exemplo mostrado, $M_G = -0.125$. Os algoritmos de otimização deverão ser configurados para **maximizar** esta métrica: quanto maior o valor de M_G, maior a concordância entre a estrutura global dos dados no espaço original e no espaço reduzido.

Grupo	**Coeficiente de Spearman**
1	-1,00
2	0,50
3	1,00
4	-1,00

Tabela 3. Cálculo de coeficiente de Spearman para avaliar modelagem de estrutura global dos dados.

3.1.2. Consistência de agrupamento (*informação mútua ajustada*)

Um algoritmo de redução de dimensionalidade que consiga representar fielmente os dados em um espaço reduzido deve preservar a estrutura de agrupamento dos dados. Portanto, o resultado da aplicação de um algoritmo de agrupamento no espaço de alta dimensionalidade deve ser o mesmo que no espaço de baixa dimensionalidade.

Neste trabalho, capturamos essa definição aplicando o algoritmo HDBSCAN separadamente nos dados posicionados no espaço original e no espaço reduzido e comparamos o resultado deste agrupamento usando a métrica de Informação Mútua Ajustada (*Adjusted Mutual Information* - AMI). Realizamos este cálculo com a implementação disponível no módulo `scikit-learn`[8].

3.1.3. Valor final da *Divergência de Kullback-Leibler*

Na nossa última métrica de avaliação, objetivamos capturar o impacto de diferentes configurações no processo de otimização do t-SNE, com o propósito de evitar mínimos locais. Parâmetros como o fator de exagero ($\mathcal{E}$), taxa de aprendizado (η) e número máximo de iterações ($\mathcal{N}$) podem ser críticos nesse caso.

Vamos utilizar o valor final da divergência KL calculado pelo t-SNE como métrica de avaliação que deve ser minimizada pelos algoritmos de configuração. É importante notar que neste caso devemos fixar a perplexidade (k), já que a divergência KL é diretamente ligada a esta grandeza (entra diretamente no cálculo de p_{ij}). Nos nossos experimentos, fixamos $k = 30$.

3.2. Dados

Nos nossos experimentos, consideramos dados de teste reais e sintéticos, para validar nossa metodologia tanto em ambientes controlados como não-controlados. Nas próximas subseções, descrevemos os conjuntos de dados sintéticos e reais utilizados em nossos experimentos.

3.2.1. Dados sintéticos

Com dados sintéticos e controlados podemos não só comparar o resultado dos experimentos usando as classes reais dos objetos (*ground-truth*) como também conhecemos o processo que gerou os dados. Dessa forma, temos uma abordagem experimental com resultados esperados previamente determinados. Cada um dos casos de teste sintéticos visa dar ênfase a algumas das características-chave do estudo. Escolhemos os seguintes conjuntos de dados:

1. **Grupos bem separados em diferentes distâncias [`well_sep`].** Neste caso de teste, objetivamos testar principalmente o objetivo de preservação de geometria global dos dados. Foram criados 8 grupos centrados em posições aleatórias em um cubo unitário de 500 dimensões. O número de elementos em cada grupo variou entre 50 e 100, conforme uma distribuição uniforme discreta. Para criar os

[8] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html

elementos de cada grupo, os centros foram perturbados igualmente em todas as dimensões conforme distribuições gaussianas com $\sigma = 0.0025$ e com a geração de um total de 599 pontos.

2. **Grupos bem separados em diferentes distâncias com ruído [`well_sep_noise`].** Um conjunto similar ao primeiro, mas com ruído uniforme adicionado ao cubo unitário de 500 dimensões onde os dados estão contidos. Objetivamos medir o efeito de ruído na preservação da geometria global dos dados. Foram adicionados 198 pontos ruidosos ao conjunto anterior, totalizando 797 pontos.
3. **Ruído gaussiano [`gaussian_noise`].** Um conjunto com distribuições normais centradas na origem de desvio $\sigma = 2$ em 200 dimensões. Com esse conjunto, objetivamos testar principalmente o objetivo de consistência entre resultados de agrupamento (o t-SNE também deve produzir ruído com esses dados). O conjunto tem um total de 500 pontos.
4. **Distribuições iguais de densidades diferentes [`topology`].** Neste conjunto temos duas distribuições gaussianas de 1000 dimensões centradas na origem mas com densidades muito discrepantes com $\sigma_1 = 2$ e $\sigma_2 = 0.001$. Com esse conjunto, objetivamos testar como cada uma das métricas propostas interfere na topologia do mapeamento do t-SNE. O conjunto tem um total de 750 pontos, sendo 250 na distribuição mais dispersa e 500 na distribuição mais densa.

Na Figura 5 mostramos exemplos dos conjuntos de dados sintéticos propostos, em um corte bidimensional para fins de visualização. Em sentido horário, começando do canto superior esquerdo: `well_sep`, `well_sep_noise`, `topology`, `gaussian_noise`.

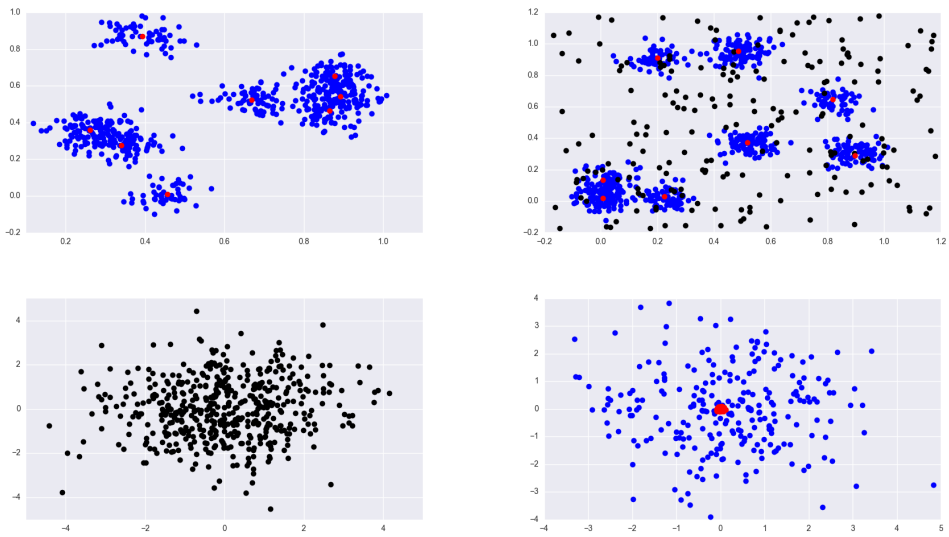

Figura 5. Cortes bidimensionais dos conjuntos de dados sintéticos propostos.

3.2.2. Dados reais

Além de dados sintéticos, propomos realizar nossos experimentos em dados reais, para avaliar as metodologias propostas em casos mais prováveis de serem encontrados na

prática. Escolhemos os seguintes conjuntos de dados para cada um dos casos de teste:

1. **COIL-20 [`coil_20`].** O conjunto de dados COIL-20 [Nene et al. 1996] é constituído de imagens em preto e branco de 20 objetos diferentes variando o ângulo da fotografia em passos de 5 graus, totalizando 1.440 exemplos. Todas as fotografias foram tiradas contra um fundo preto.
2. **Olivetti Faces [`olivetti`].** *The Database of Faces* [Samaria and Harter 1994], mais conhecido como o conjunto *Olivetti Faces*, é composto de 400 imagens de 40 indivíduos (10 imagens para cada um) onde para cada indivíduo são variadas algumas características como iluminação, expressões faciais, horário do dia e o uso de óculos.

Na Figura 6, mostramos alguns exemplos destes conjuntos. São conjuntos bastante conhecidos na área e mostraram resultados interessantes no primeiro trabalho do t-SNE. Portanto, são candidatos naturais para os experimentos propostos.

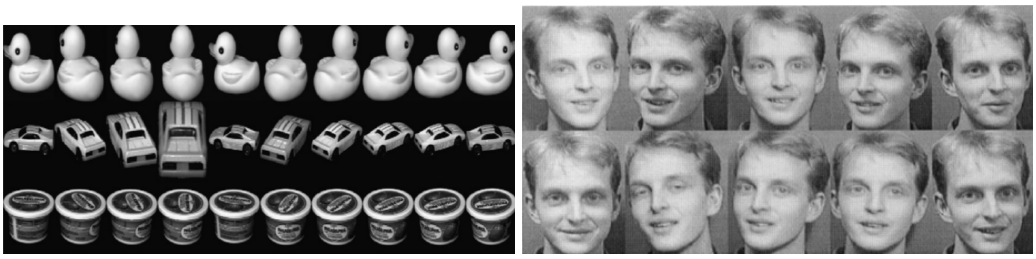

Figura 6. Exemplos de imagens retiradas dos conjuntos COIL-20 (objetos à esquerda) e Olivetti Faces (fotografias à direita).

3.3. *Pipeline* final

Nesta seção, apresentamos uma descrição geral da metodologia. Na Figura 7 está apresentado visualmente o *pipeline* de otimização proposto. Em linhas gerais, o processo é conduzido da seguinte forma:

1. Definem-se espaços de busca através de distribuições de probabilidade para os métodos de busca aleatória e TPE. Para Processos Gaussianos, basta definir o espaço de busca através de fronteiras (limites superiores e inferiores) para cada um dos hiperparâmetros.
2. É instanciada uma tarefa de otimização escolhendo um dos seis conjuntos de dados.
3. Para cada rodada de otimização, a sequência de PCA e t-SNE é executada sob a configuração de hiperparâmetros selecionada pela metodologia de otimização em análise. Utilizando os mapeamentos e os resultados de agrupamento, computamos as métricas de avaliação definidas. Para a métrica *rank-order correlation* (estrutura global), HDBSCAN é executado somente nos dados em alta dimensionalidade, enquanto que para *AMI* (consistência de agrupamento) HDBSCAN é executado tanto em alta dimensionalidade quanto em baixa dimensionalidade. Para a métrica *divergência KL*, HDBSCAN não é executado.
4. O algoritmo de otimização de hiperparâmetros recebe o resultado do experimento, calculando uma nova configuração para o *pipeline*.

Para todos os experimentos, o número de rodadas de otimização definido foi de 50. O algoritmo de Processos Gaussianos é inicializado com 10 experimentos aleatórios, restando 40 para otimização.

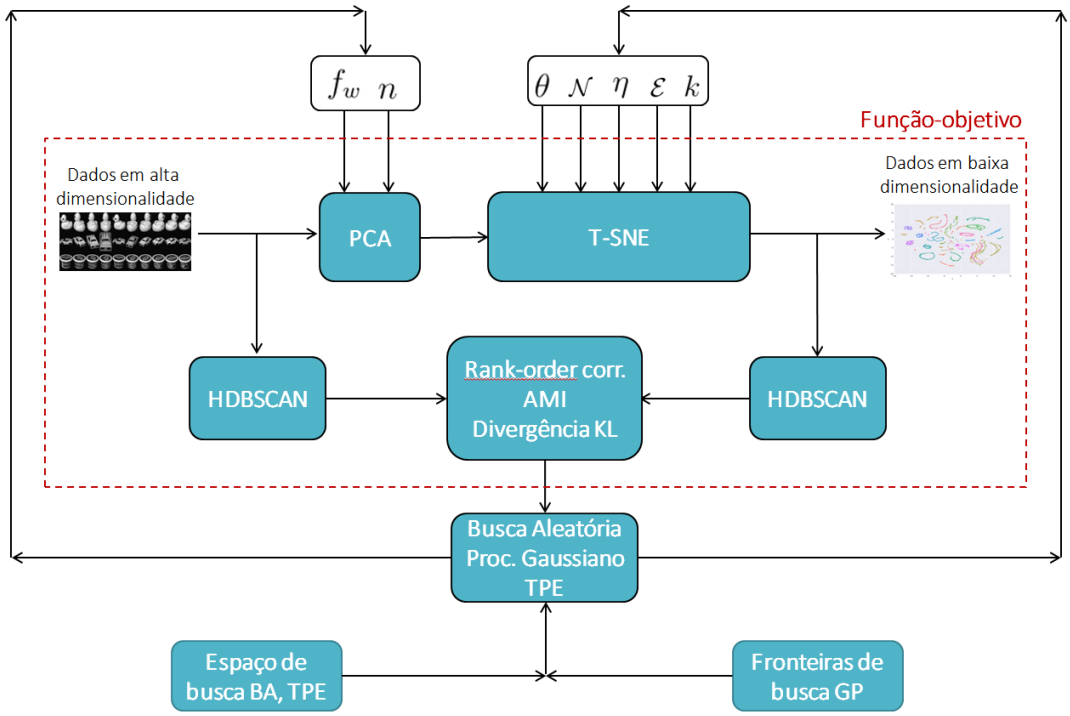

Figura 7. Visão geral do *pipeline* de otimização utilizado.

4. Resultados

Nesta seção, são apresentados os resultados dos experimentos conduzidos, divididos em três análises, de acordo com os objetivos do projeto: (a) representação de estrutura global, (b) consistência de agrupamento e (c) otimização da divergência KL. Mostramos os resultados das três metodologias de otimização (Busca Aleatória, TPE e GP) em seis conjuntos de dados diferentes.

4.1. Representação de estrutura global

Modelamos a qualidade da representação da estrutura global dos dados através da métrica de *rank-order correlation*, discutida na seção anterior. Na Figura 8, apresentamos os resultados de otimização para os conjuntos `well_sep`, `well_sep_noise` e `coil_20` para nossas três metodologias de seleção de hiperparâmetros (`rand`, `tpe` e `gp`). Não foi possível obter resultados para os conjuntos `gaussian_noise`, `topology` e `olivetti`, uma vez que nos dois primeiros não há mais de um cluster, condição necessária para computar nossa métrica. Para o último caso, os clusters são muito pequenos para serem identificados.

Mostramos os resultados na forma de histogramas, indicando quantas vezes cada resultado foi alcançado. Para *rank-order correlation*, o resultado é melhor à medida que se aproxima de 1.0. Ou seja, quanto maior a concentração de medições **para a direita** dos

histogramas, melhor foi a performance do algoritmo de otimização (alcançou mais vezes bons resultados).

Os resultados foram relativamente balanceados, porém com uma vantagem do algoritmo TPE. Na Tabela 4, mostramos o resultado médio, além do resultado da pior e da melhor configuração encontrada pelas metodologias em termos de *rank-order correlation*. Em cinco das nove combinações possíveis de conjunto de dados e métrica, TPE mostrou melhores resultados (em destaque), contra duas combinações para Busca Aleatória e duas para Processos Gaussianos.

Além disso, mostramos na Figura 9 uma comparação qualitativa entre o melhor mapeamento em termos de *rank-order correlation* atingido pelos algoritmos contra um mapeamento realizado utilizando os parâmetros sugeridos por [Maaten and Hinton 2008]. É notável a diferença entre o ordenamento de proximidade entre os grupos. No segundo caso (`well_sep_noise`) os elementos de ruído se distribuíram entre os clusters, mostrando que essa configuração conseguiu capturar a estrutura globalmente dispersa desses objetos. No caso de `coil_20`, os clusters ficaram mais evidentes, e vemos que o tamanho de cada cluster e as distâncias relativas entre os clusters muda drasticamente. Com parâmetros sugeridos, há uma uniformidade de tamanhos e distâncias.

Finalmente, mostramos na Tabela 5 os valores da função-objetivo (`loss`) e dos hiperparâmetros nas 5 melhores configurações obtidas nos experimentos de otimização para cada conjunto de dados, e também na configuração sugerida. Obtivemos escolhas de perplexidades altas, mostrando que existem boas configurações fora do intervalo $I = [5, 50]$ recomendado como boa prática. Porém, este resultado é alinhado com a intuição do algoritmo: perplexidades mais altas capturam estrutura global com maior assertividade. *Whitening* não foi utilizado na maioria das vezes. O fator de exagero quase sempre esteve acima de 4, que é o valor recomendado. Busca Aleatória e TPE foram os métodos mais presentes entre os 5 melhores mapeamentos. Os valores atingidos de *rank-order correlation* foram razoáveis, porém não muito altos: a estrutura global em baixa dimensionalidade continua sendo uma aproximação da estrutura global no espaço original.

Tabela 4. Comparação entre metodologias de otimização para a métrica de *rank-order correlation*. TPE e Busca aleatória mostraram melhores resultados.

`well_sep`			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
`rand`	0.013750	0.455357	-0.410714
`tpe`	**0.131250**	**0.464286**	**-0.330357**
`gp`	-0.024888	0.321429	-0.343750
`well_sep_noise`			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
`rand`	0.007589	**0.308036**	-0.250000
`tpe`	-0.023393	0.281250	-0.276786
`gp`	**0.029464**	0.156250	**-0.165179**
`coil_20`			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
`rand`	0.137164	0.332194	**-0.027591**
`tpe`	**0.153791**	**0.341741**	-0.054992
`gp`	0.035610	0.326673	-1.000000

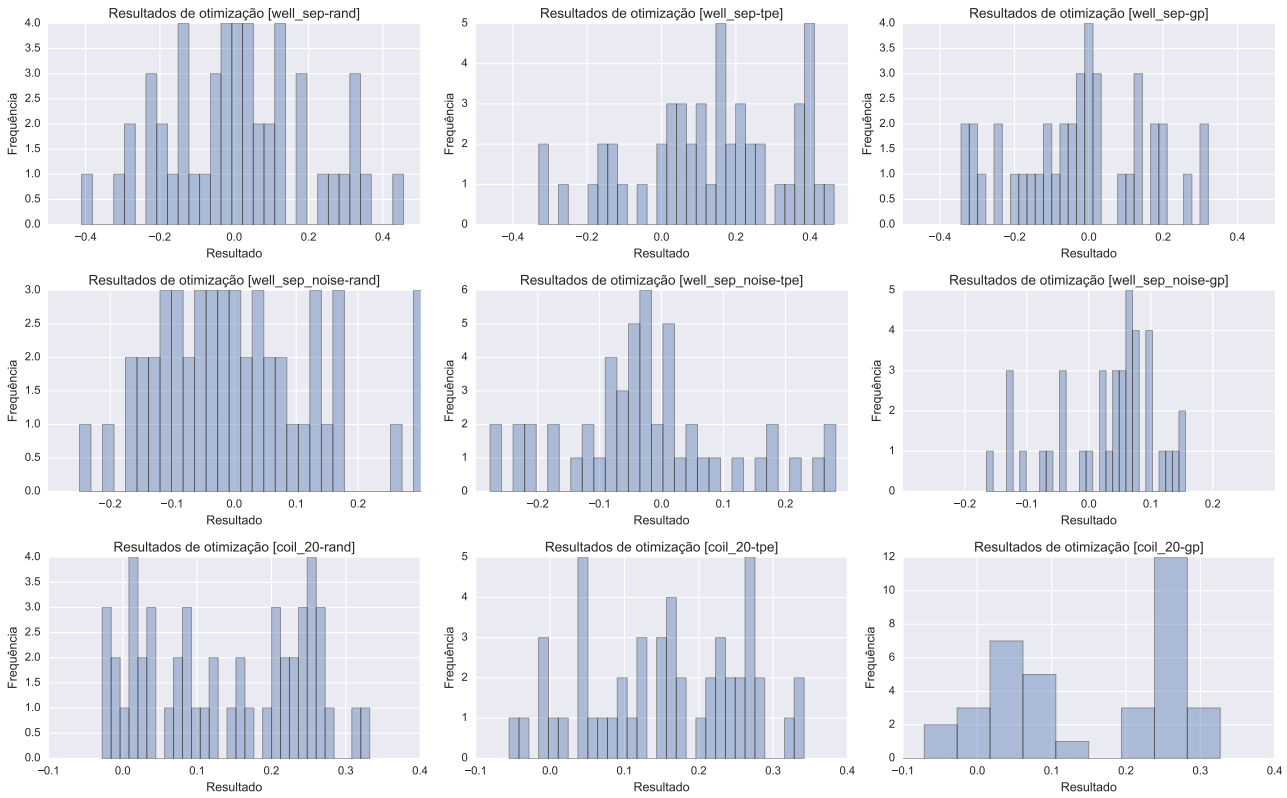

Figura 8. Histogramas de otimização. Vemos que TPE e Busca Aleatória mostraram bom desempenho, fazendo o *pipeline* alcançar bons resultados em termos de representação de estrutura global com maior frequência.

Tabela 5. Configurações sugeridas por [Maaten and Hinton 2008] e para os 5 melhores mapeamentos de baixa dimensionalidade, para cada conjunto de dados, em termos de *rank-order correlation*. É notável o uso de perplexidades consideravelmente mais altas do que é tido como boa prática.

well_sep									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.464	0.269	5.114	849.90	4800	40	178	False	tpe
2	0.455	0.216	6.022	623.88	2200	20	115	False	rand
3	0.429	0.202	7.451	422.11	3900	20	148	False	tpe
4	0.411	0.212	7.257	433.86	3800	35	144	False	tpe
5	0.411	0.219	5.139	456.09	3800	30	144	False	tpe
-	**0.022**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
well_sep_noise									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.308	0.672	11.574	420.37	4300	55	8	True	rand
2	0.308	0.479	6.040	533.46	3400	20	50	False	rand
3	0.304	0.597	5.180	649.96	5000	15	153	False	rand
4	0.281	0.574	7.738	661.52	4300	25	183	False	tpe
5	0.268	0.280	5.678	797.30	3500	90	188	True	tpe
-	**-0.205**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
coil_20									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.342	0.489	11.009	739.74	3900	100	115	False	tpe
2	0.339	0.750	3.923	677.63	3400	80	124	False	tpe
3	0.332	0.695	6.510	524.25	400	30	171	False	rand
4	0.327	0.200	12.000	1000.00	2707	10	200	False	gp
5	0.327	0.200	1.000	1000.00	1734	10	200	False	gp
-	**-0.002**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**

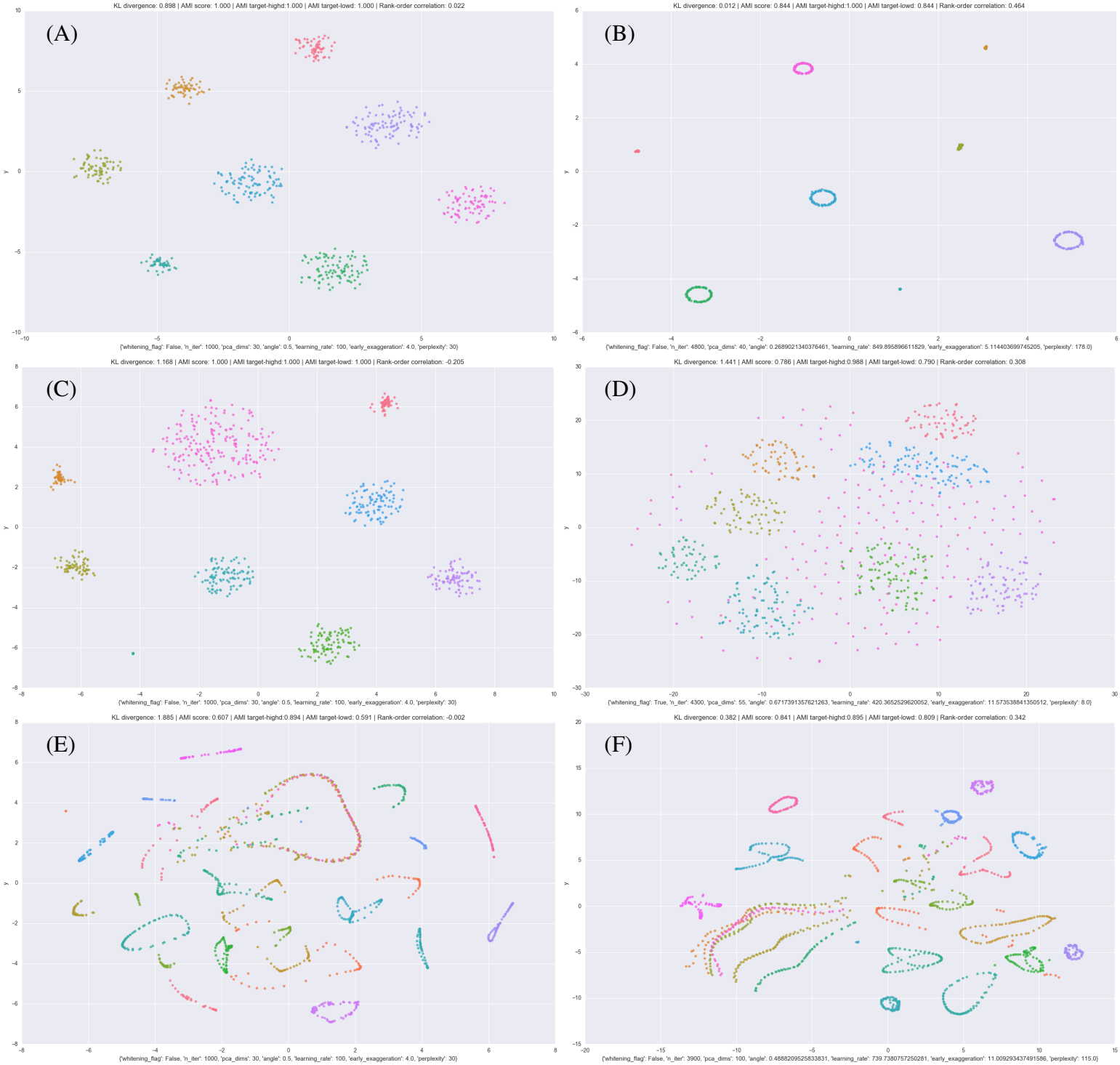

Figura 9. Comparação entre mapeamentos com parâmetros sugeridos por [Maaten and Hinton 2008] (esquerda), e melhor mapeamento encontrado pelos algoritmos de otimização (direita), em termos de *rank-order correlation*. Na sequência, de cima para baixo: `well_sep` (A e B), `well_sep_noise` (C e D) e `coil_20` (E e F).

4.2. Consistência de agrupamento

Para medir consistência de agrupamento, aplicamos o algoritmo HDBSCAN tanto nos dados em alta dimensionalidade quanto nos dados em baixa dimensionalidade, e comparamos os resultados através de Informação Mútua Ajustada (AMI), como descrito na Seção 3. Como na subseção anterior, não foi possível obter resultados para os conjuntos de dados `gaussian_noise`, `topology` e `olivetti`, já que não é possível extrair agrupamentos significativos desses conjuntos. Mostramos os resultados para todos os algoritmos de otimização nos conjuntos `well_sep`, `well_sep_noise` e `coil_20` na Figura 10, na forma de histogramas. Um resultado de AMI próximo de 1 nos indica que o HDBSCAN produziu o mesmo resultado de agrupamento usando os dados tanto em alta dimensionalidade quanto em baixa dimensionalidade, mostrando que os agrupamentos naturais dos dados foram preservados. Por outro lado, AMI próximo de 0 evidencia que o t-SNE distorceu a distribuição de dados de tal forma que não é mais possível extrair seus agrupamentos naturais. Portanto, temos bons resultados se as medições se concentrarem **à direita** e **próximas de 1** nos histogramas.

Em geral, o algoritmo TPE mostrou melhores resultados, seguido da Busca Aleatória. Na Tabela 6, mostramos uma síntese dos resultados. Nos conjuntos `well_sep` e `well_sep_noise`, todos os métodos de otimização conseguiram configurar o *pipeline* de forma a conseguir o melhor resultado possível pelo menos uma vez, apesar de TPE ter chegado mais vezes em bons resultados (resultado médio maior). No caso de `coil_20`, os resultados de TPE e Busca Aleatória foram próximos, com TPE obtendo melhores resultados médios mas Busca Aleatória obtendo a melhor configuração. Em todos os conjuntos, os melhores mapeamentos obtiveram bons resultados de consistência de agrupamento. Porém, em rodadas de otimização intermediárias, os algoritmos também chegaram em resultados ruins, mostrando que o problema não é trivial e que os hiperparâmetros têm um grande impacto na qualidade dos mapeamentos produzidos pelo t-SNE.

Na Figura 11, comparamos o melhor mapeamento produzido pelas metodologias de otimização com os mapeamentos produzidos pelo t-SNE com parâmetros sugeridos, em termos de informação mútua ajustada. Nos conjuntos `well_sep` e `well_sep_noise`, temos resultados parecidos, mostrando robustez nos parâmetros sugeridos em [Maaten and Hinton 2008]. Porém, no caso de `coil_20`, temos resultados bastante diferentes. O resultado de AMI do t-SNE configurado com parâmetros sugeridos foi de 0.607 contra um valor de 0.858 obtido com parâmetros ótimos. Nesse caso, é possível notar melhorias, como uma menor sobreposição entre pontos de diferentes clusters, além de não haver entrelaçamento entre clusters.

Por fim, na Tabela 7, comparamos as configurações dos cinco melhores mapeamentos obtidos, contra a configuração sugerida. Para `well_sep` e `well_sep_noise`, observamos configurações diversas levando a bons resultados, inclusive com parâmetros sugeridos. Em `coil_20`, temos um conjunto de hiperparâmetros mais restrito, com `learning_rate` entre 650 e 1000, `pca_dims` acima de 50, e perplexidades entre 50 e 120. Neste caso, mostramos que existem configurações boas fora dos intervalos tidos como boas práticas.

Tabela 6. Comparação entre metodologias de otimização para a métrica de informação mútua ajustada. Busca Aleatória se mostrou uma metodologia de otimização competitiva, com resultados próximos aos obtidos por TPE.

well_sep			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
rand	0.628055	**1.000000**	-0.001554
tpe	**0.764075**	**1.000000**	-0.001077
gp	0.423185	**1.000000**	**0.000677**
well_sep_noise			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
rand	0.786561	**0.988115**	0.000404
tpe	**0.847412**	**0.988115**	**0.003484**
gp	0.498063	**0.988115**	0.000000
coil_20			
Metodologia	**Resultado médio**	**Melhor Resultado**	**Pior Resultado**
rand	0.734528	**0.858499**	**0.575072**
tpe	**0.744464**	0.844431	0.553622
gp	0.504571	0.823488	0.000000

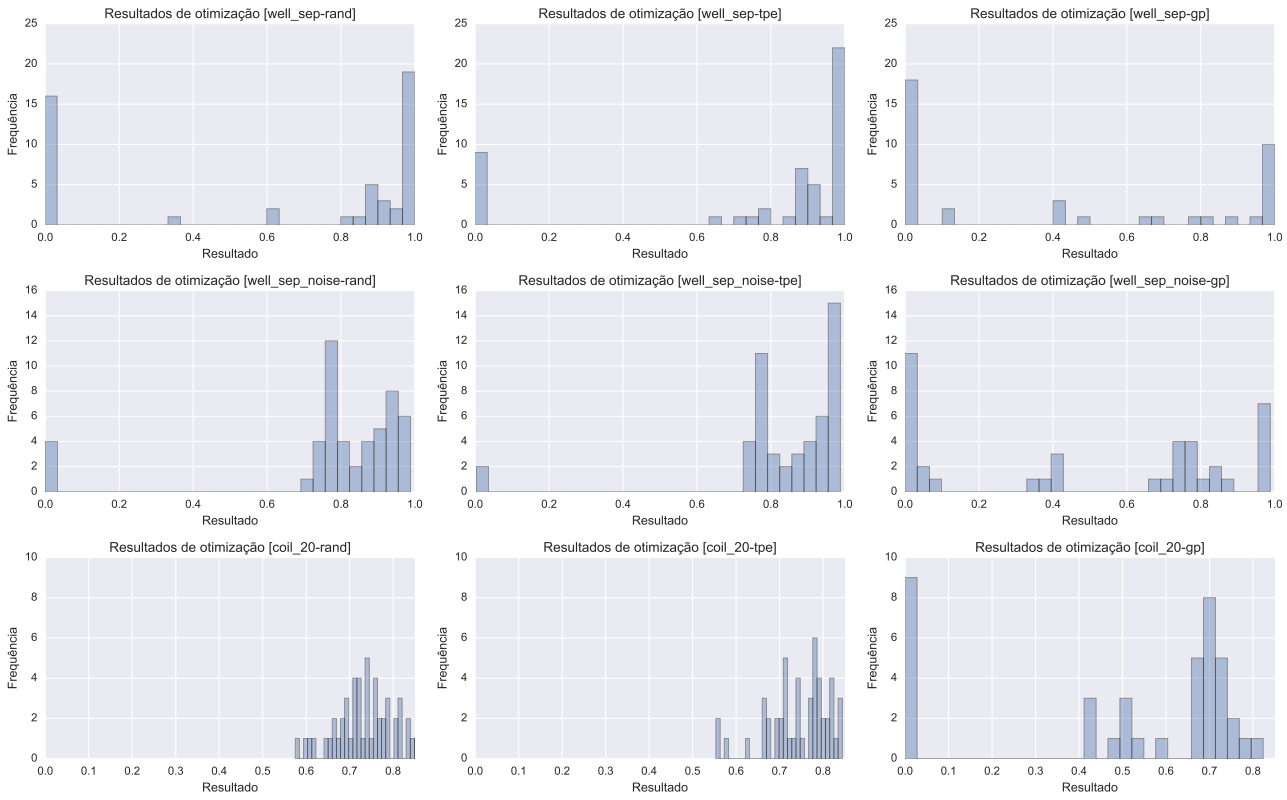

Figura 10. Histogramas de otimização. Vemos que o algoritmo TPE mostrou o melhor desempenho, fazendo o *pipeline* alcançar bons resultados em termos de consistência de agrupamento com maior frequência.

Tabela 7. Configurações sugeridas por [Maaten and Hinton 2008] e para os 5 melhores mapeamentos de baixa dimensionalidade, para cada conjunto de dados, em termos de informação mútua ajustada. Para `well_sep`, `well_sep_noise`, diversas configurações levaram a bons resultados, incluindo a configuração padrão. Contudo, em `coil_20` um conjunto de parâmetros mais restrito foi observado, marcado por taxas de aprendizado e perplexidades altas.

well_sep									
rank	**result**	**angle**	**early_exaggeration**	**learning_rate**	**n_iter**	**pca_dims**	**perplexity**	**whitening_flag**	**optim**
1	1.000	0.762	5.642	69.61	2000	40	199	False	tpe
2	1.000	0.379	7.707	561.78	4300	75	162	False	tpe
3	1.000	0.701	2.889	267.71	400	90	14	False	rand
4	1.000	0.797	8.159	366.31	2000	60	68	False	rand
5	1.000	0.800	1.000	1000.00	1934	10	200	True	gp
-	**1.000**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
well_sep_noise									
rank	**result**	**angle**	**early_exaggeration**	**learning_rate**	**n_iter**	**pca_dims**	**perplexity**	**whitening_flag**	**optim**
1	0.988	0.800	1.000	50.00	842	10	200	False	gp
2	0.988	0.481	7.497	979.22	3600	45	38	False	tpe
3	0.988	0.480	6.075	186.56	3900	60	36	False	tpe
4	0.988	0.800	12.000	50.00	2363	10	200	False	gp
5	0.988	0.800	12.000	50.00	3833	10	200	True	gp
-	**0.988**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
coil_20									
rank	**result**	**angle**	**early_exaggeration**	**learning_rate**	**n_iter**	**pca_dims**	**perplexity**	**whitening_flag**	**optim**
1	0.858	0.330	7.574	882.68	2700	75	113	False	rand
2	0.846	0.438	5.824	867.48	2300	55	54	False	rand
3	0.844	0.564	1.295	661.88	2100	60	80	True	tpe
4	0.840	0.313	1.512	762.17	3400	95	55	False	tpe
5	0.835	0.617	1.277	990.52	1400	50	83	True	tpe
-	**0.607**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**

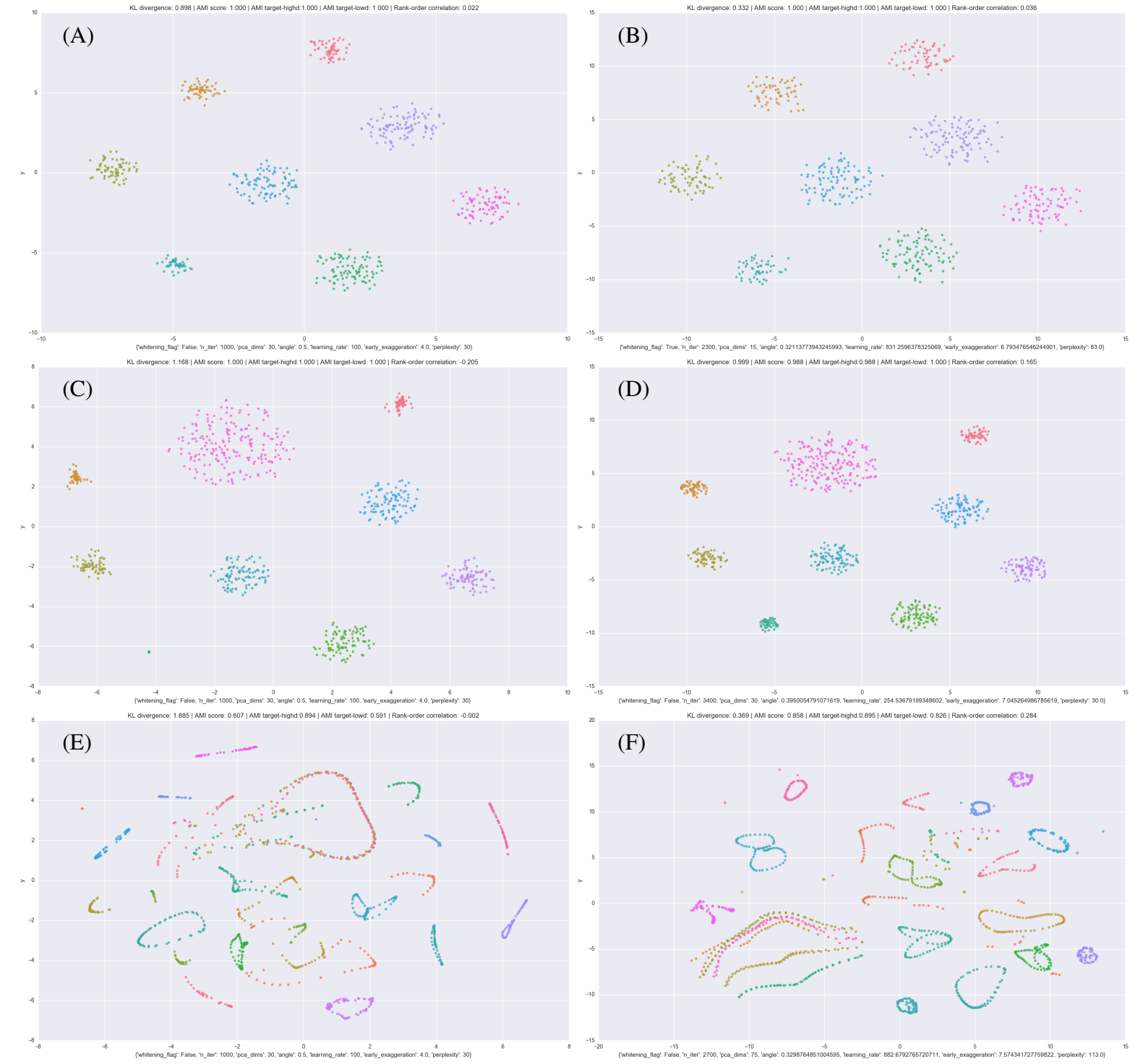

Figura 11. Comparação entre mapeamentos com parâmetros sugeridos por [Maaten and Hinton 2008] (esquerda), e melhor mapeamento encontrado pelos algoritmos de otimização (direita), em termos de informação mútua ajustada. Na sequência, de cima para baixo: `well_sep` (A e B), `well_sep_noise` (C e D) e `coil_20` (E e F).

4.3. Otimização de divergência KL

Nossa métrica final é o valor da função-objetivo do t-SNE na última iteração de otimização. A divergência de Kullback-Leibler, calculada para medir a semelhança entre as distribuições de probabilidade associadas aos mapeamentos de baixa e alta dimensionalidade, é reconhecida por estar sujeita a mínimos locais. Para aliviar este problema, otimizamos os hiperparâmetros do algoritmo, com exceção da perplexidade, que faz parte da definição das distribuições de probabilidade. Taxa de aprendizado, número de iterações e fator de exagero são fundamentais para este processo.

Como esta métrica não depende de nenhum agrupamento, temos resultados para todos os conjuntos de dados considerados. Um resultado de divergência KL próximo de 0 nos mostra que as distribuições em análise são muito parecidas. Quanto maior a divergência KL, maior é a diferença entre as distribuições. Não há um limite superior para esta métrica, podendo atingir valores infinitos. Dessa forma, nos histogramas de resultados, mostrados na Figura 12, bons resultados se localizam **à esquerda** e **próximos de 0**. Na Tabela 8, mostramos os resultados médios, melhores e piores dos métodos de otimização, como nas seções anteriores.

Processos Gaussianos e TPE mostraram os melhores resultados, com TPE mostrando maior consistência (4 de 6 melhores resultados médios) mas GP com os melhores resultados únicos (5 de 6 melhores mapeamentos). Nos histogramas, podemos notar que as medições se concentraram mais à esquerda com TPE, uma vez que mostrou o melhor resultado médio, com GP mostrando padrões similares.

Nas Figuras 13 e 14, realizamos uma comparação qualitativa das visualizações geradas pelos parâmetros recomendados por [Maaten and Hinton 2008] e pelos melhores parâmetros encontrados nos experimentos. Nos casos de `well_sep`, `well_sep_noise` e `gaussian_noise` não foram evidenciadas grandes diferenças, com mapeamentos qualitativamente similares. Contudo, em `topology`, observamos que os pontos da distribuição densa (em verde) ficaram mais unidos entre si e mais separados da distribuição dispersa (em azul). Em `coil_20`, vemos que há uma maior coesão entre os clusters, também evidenciado pelo aumento de AMI, sugerindo que esta métrica também pode ser beneficiada com a otimização da divergência KL. Também observamos este padrão em `olivetti`.

Finalmente, mostramos na Tabela 9 a comparação dos resultados e parâmetros obtidos pelos 5 melhores mapeamentos contra o mapeamento com parâmetros sugeridos. Em todos os casos observamos valores de `learning_rate` e `n_iter` consideravelmente mais altos que os sugeridos, mostrando processos de otimização mais agressivos. O parâmetro `pca_dims` se manteve baixo, além de `whitening_flag` estar ativa, reduzindo a complexidade de entrada dos dados. Os outros parâmetros variaram bastante, mostrando que o foco da seleção de hiperparâmetros foi o processo de otimização.

Tabela 8. Comparação entre metodologias de otimização para a métrica de divergência KL. Processos de otimização mais agressivos foram favorecidos, com `n_iter` e `learning_rate` assumindo valores altos.

well_sep			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	1.306324	1.920038	0.682447
tpe	1.001414	**1.895389**	0.414864
gp	**0.869961**	2.089794	**0.409589**
well_sep_noise			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	0.950799	1.458431	0.716205
tpe	**0.821274**	**1.181619**	0.473567
gp	0.835586	1.459449	**0.466183**
gaussian_noise			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	1.816119	**1.962306**	1.455277
tpe	1.780158	2.075961	1.449643
gp	**1.662728**	2.06278	**1.439478**
topology			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	1.438342	**1.487019**	1.301611
tpe	**1.40030**	1.587147	1.274059
gp	1.467642	1.911154	**1.233788**
coil_20			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	1.091031	**2.524966**	0.728873
tpe	**0.938274**	2.607371	0.693599
gp	1.08498	2.634907	**0.656221**
olivetti			
Metodologia	**Resultado médio**	**Pior Resultado**	**Melhor Resultado**
rand	0.674381	0.974427	0.567711
tpe	**0.642205**	**0.886099**	**0.566233**
gp	0.759249	1.189383	0.577439

Figura 12. Histogramas de otimização para divergência KL. Vemos bons resultados para Processos Gaussianos e para o TPE.

Tabela 9. Configurações sugeridas por [Maaten and Hinton 2008] e para os 5 melhores mapeamentos de baixa dimensionalidade, para cada conjunto de dados, em termos de divergência KL. Processos de otimização mais agressivos foram favorecidos, com altas taxas de aprendizado e muitas iterações.

well_sep									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.409589	0.800000	12.000000	687.368818	3827.126811	10.0	30.0	True	gp
2	0.413368	0.800000	12.000000	635.315748	1750.303093	10.0	30.0	True	gp
3	0.414172	0.800000	12.000000	735.239535	4536.013782	10.0	30.0	True	gp
4	0.414864	0.409643	8.575245	690.963907	3100.000000	10.0	30.0	True	tpe
5	0.414878	0.380031	2.270071	682.489103	2600.000000	10.0	30.0	True	tpe
-	**0.898**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
well_sep_noise									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.466183	0.792731	1.000000	776.330291	1257.033112	10.0	30.0	True	gp
2	0.473567	0.327058	10.990380	679.015381	2400.000000	10.0	30.0	True	tpe
3	0.474683	0.799988	1.000000	637.894665	3197.549523	10.0	30.0	True	gp
4	0.474854	0.390576	11.794448	683.782117	1000.000000	10.0	30.0	True	tpe
5	0.475556	0.200000	1.000000	1000.000000	2054.448873	10.0	30.0	True	gp
-	**1.168**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
gaussian_noise									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	1.439478	0.20000	1.000000	1000.000000	3489.976458	10.0	30.0	False	gp
2	1.439478	0.20000	12.000000	1000.000000	1521.488773	10.0	30.0	False	gp
3	1.441191	0.20000	12.000000	827.882048	2838.335732	10.0	30.0	True	gp
4	1.444441	0.20000	12.000000	602.355950	3538.196029	10.0	30.0	True	gp
5	1.449643	0.33292	7.657583	661.284024	2300.000000	10.0	30.0	True	tpe
-	**1.904**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
topology									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	1.233788	0.200000	12.0	1000.0	2034.168703	10.0	30.0	False	gp
2	1.247027	0.200000	12.0	1000.0	3168.754345	10.0	30.0	False	gp
3	1.252653	0.800000	1.0	1000.0	5000.000000	10.0	30.0	True	gp
4	1.256570	0.674570	12.0	1000.0	4224.460489	10.0	30.0	True	gp
5	1.257400	0.777289	1.0	1000.0	2465.610121	10.0	30.0	False	gp
-	**1.446**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
coil_20									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.656221	0.8	1.0	1000.0	5000.000000	10.0	30.0	True	gp
2	0.656221	0.8	12.0	1000.0	2424.622639	10.0	30.0	True	gp
3	0.656221	0.8	12.0	1000.0	4083.241391	10.0	30.0	True	gp
4	0.656221	0.8	12.0	1000.0	3288.463122	10.0	30.0	True	gp
5	0.656221	0.8	1.0	1000.0	1815.592000	10.0	30.0	True	gp
-	**0.694**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**
olivetti									
rank	result	angle	early_exaggeration	learning_rate	n_iter	pca_dims	perplexity	whitening_flag	optim
1	0.566233	0.516768	5.373478	455.185754	2400.0	30.0	30.0	False	tpe
2	0.567711	0.226195	3.928807	613.077460	800.0	90.0	30.0	False	rand
3	0.568599	0.796125	10.322247	482.272149	2100.0	40.0	30.0	False	tpe
4	0.568670	0.443713	3.778983	600.519895	3300.0	20.0	30.0	False	tpe
5	0.570581	0.795021	8.560583	454.335895	2200.0	35.0	30.0	False	tpe
-	**0.793**	**0.500**	**4.000**	**100.00**	**1000**	**30**	**30**	**False**	**Maaten et. al.**

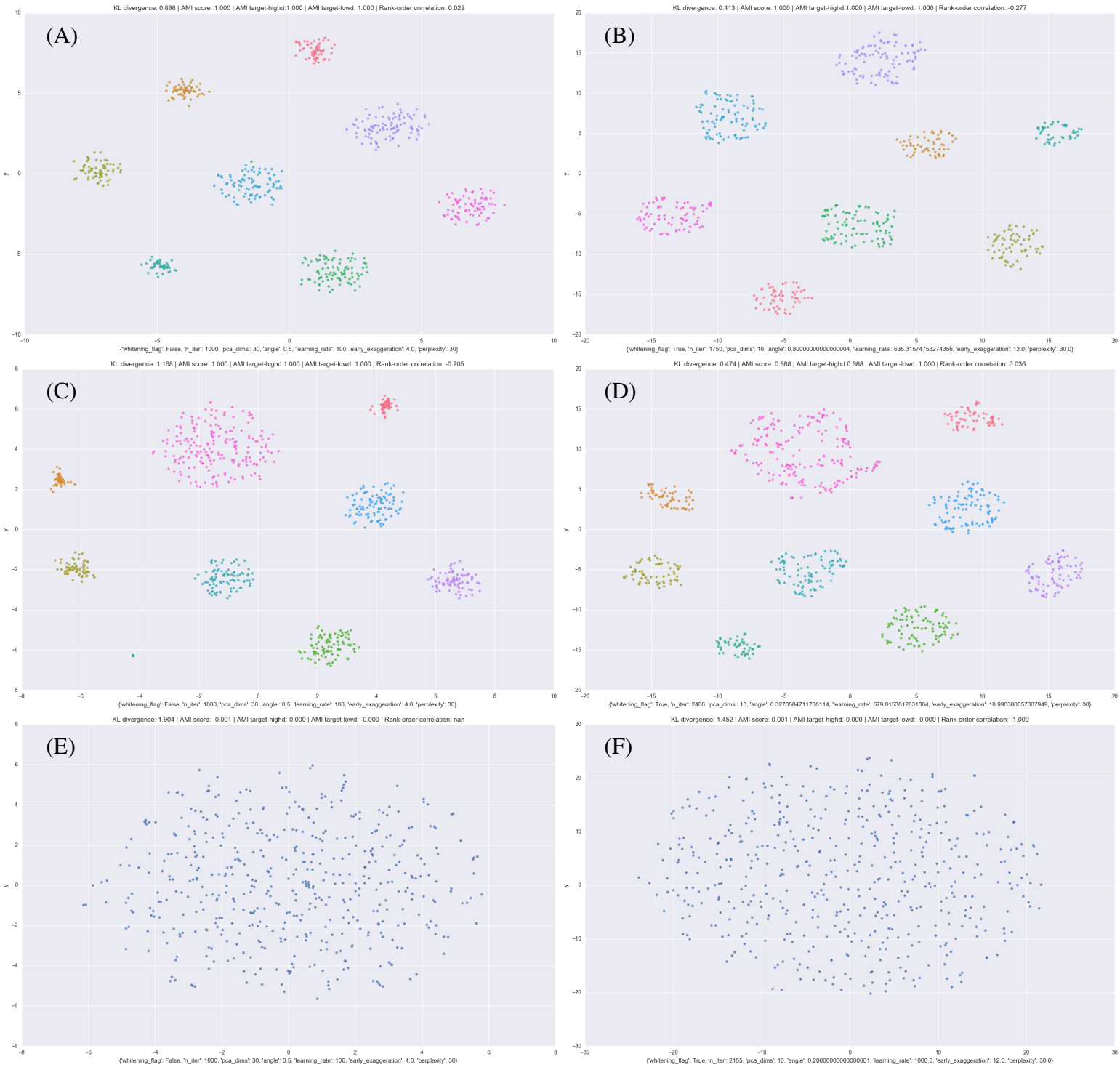

Figura 13. Comparação entre mapeamentos com parâmetros sugeridos por [Maaten and Hinton 2008] (esquerda), e melhor mapeamento encontrado pelos algoritmos de otimização (direita), em termos de divergência KL. Na sequência, de cima para baixo: `well_sep` (A e B), `well_sep_noise` (C e D) e `gaussian_noise` (E e F).

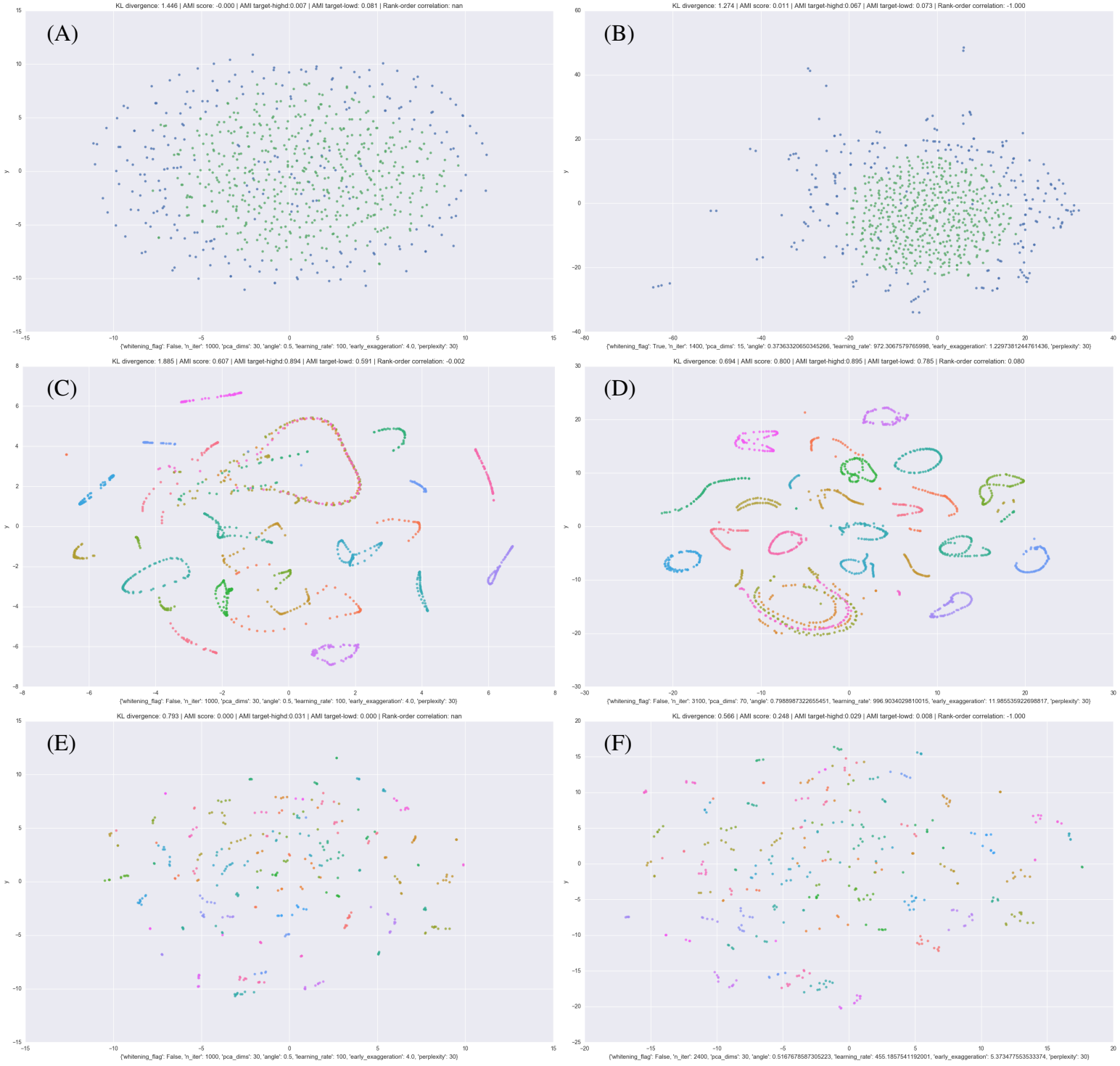

Figura 14. Comparação entre mapeamentos com parâmetros sugeridos por [Maaten and Hinton 2008] (esquerda), e melhor mapeamento encontrado pelos algoritmos de otimização (direita), em termos de divergência KL. Na sequência, de cima para baixo: `topology` (A e B), `coil_20` (C e D) e `olivetti` (E e F).

5. Conclusão

Neste trabalho, mostramos os resultados da aplicação de três metodologias de otimização de hiperparâmetros (TPE, Processos Gaussianos e Busca Aleatória) na configuração do t-SNE, visando gerar visualizações otimizadas no que diz respeito a três objetivos diferentes: (a) representação de estrutura global, através da métrica de *rank-order correlation*; (b) consistência de agrupamento, através da métrica de informação mútua ajustada e (c) otimização de divergência KL, através do resultado final da otimização do t-SNE. Seis conjuntos de dados foram utilizados para validar as abordagens, com quatro conjuntos sintéticos derivados do trabalho de [Wattenberg et al. 2016] e dois conjuntos reais, também utilizados em [Maaten and Hinton 2008], para fins de comparação.

No que diz respeito à representação de estrutura global e consistência de agrupamento, TPE mostrou melhores resultados, seguido de perto por Busca Aleatória. Em otimização de divergência KL, o método de Processos Gaussianos mostrou melhores resultados, com uma pequena margem de diferença. Em geral, recomendamos o uso de TPE, que mostrou resultados mais consistentes. Contudo, Busca Aleatória se mostrou um método bastante poderoso dada a sua simplicidade, atingindo resultados próximos (e em alguns casos, melhores) em comparação aos outros dois métodos. Nesta linha, um caminho de exploração futura pode ser a metodologia conhecida por *Hyperband* [Li et al. 2016], que otimiza os recursos cedidos a cada iteração de uma busca aleatória, aproveitando a natureza paralelizável deste algoritmo para conseguir bons resultados em menor tempo que as outras metodologias.

O método de Processos Gaussianos, apesar de ter mostrado bons resultados no experimento de otimização da divergência KL, não foi competitivo de uma maneira geral. Um caminho futuro de exploração para melhorar esta metodologia é de experimentar Thompson Sampling como função de aquisição para o algoritmo, uma técnica que tem mostrado bons resultados em tarefas de aprendizado por reforço [Chapelle and Li 2011]. Uma função de aquisição baseada em Thompson Sampling não requer hiperparâmetros, diferente de UCB, em que devemos ajustar `kappa` para definir o compromisso entre *exploration* e *exploitation*. Dessa forma, poderíamos simplificar o uso da metodologia de otimização, enquanto mantemos um processo de otimização balanceado.

As métricas desenvolvidas atingiram seu objetivo de aumentar a qualidade das representações em baixa dimensionalidade geradas pelo t-SNE. Otimizando *rank-order correlation*, chegamos a visualizações mais representativas da estrutura global dos dados. Em `well_sep` e `well_sep_noise`, o posicionamento relativo dos clusters foi alterado. Para `coil_20`, além da posição relativa, o tamanho relativo dos clusters se alterou. Mostramos, através dos experimentos, que usar valores mais elevados de perplexidade do que é tido como boa prática é essencial para atingir este objetivo. Contudo, *rank-order correlation* tem duas fraquezas: (1) dependência de uma metodologia de agrupamento e (2) utilizar o centróide dos clusters para computar os rankings de distância (o que favorece clusters circulares). Em trabalhos futuros, ao invés de utilizar clusters e seus centróides, pode ser interessante utilizar os *core points* como referência para esses cálculos. Dessa forma, poderíamos avaliar a qualidade da representação de estrutura global em dados onde não há clusters ou estes são difíceis de encontrar, como em `gaussian_noise`, `topology` e `olivetti`.

Com a métrica de informação mútua ajustada, foi possível obter configurações interes-

santes. Nos conjuntos `well_sep` e `well_sep_noise`, não foram obtidos grandes ganhos, já que com parâmetros recomendados temos resultados robustos. Porém, no conjunto `coil_20` observamos uma grande diferença na qualidade das visualizações obtidas. Foi observada uma melhor separação entre os clusters, além de menor entrelaçamento. Também foram observadas perplexidades mais altas em comparação com as boas práticas nos melhores mapeamentos, porém em geral menores que as observadas nos experimentos de *rank-order correlation*.

Em seguida, nos experimentos utilizando a métrica de divergência KL, pudemos observar melhorias nos mapeamentos, apesar de menos expressivas do que foi observado nos experimentos com as duas outras métricas. Neste caso, sugerimos utilizar o procedimento de otimização de divergência KL como uma maneira mais agressiva de evitar mínimos locais. Por ser um algoritmo dependente de uma inicialização aleatória, a comunidade aconselha executar o t-SNE algumas vezes com os mesmos parâmetros e publicar a visualização com a melhor divergência KL. Recomendamos, então, o mesmo procedimento, mas otimizando os hiperparâmetros do t-SNE (com exceção da perplexidade) utilizando os métodos propostos. Assim, os recursos computacionais são bem aproveitados para melhorar a convergência do algoritmo, gerando ganhos mais expressivos.

Finalmente, planejamos disponibilizar para a comunidade uma ferramenta de otimização do t-SNE, baseada nos resultados deste projeto. O código utilizado para gerar os resultados mostrados neste documento será disponibilizado em `https://github.com/gdmarmerola/hyper-optim-thesis`, por motivos de reprodutibilidade.

Referências

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Bergstra, J., Yamins, D., and Cox, D. D. (2013a). Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. Citeseer.

Bergstra, J., Yamins, D., and Cox, D. D. (2013b). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *ICML (1)*, 28:115–123.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Campello, R. J., Moulavi, D., and Sander, J. (2013). Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172. Springer.

Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Hinton, G. E. and Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 833–840.

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2016). Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nene, S. A., Nayar, S. K., Murase, H., et al. (1996). Columbia object image library (coil-20).

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Samaria, F. S. and Harter, A. C. (1994). Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 138–142. IEEE.

Van Der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245.

Wattenberg, M., Viégas, F., and Johnson, I. (2016). How to use t-SNE effectively. http://distill.pub/2016/misread-tsne/.