UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

Um estudo da aplicação do LASSO para seleção de instâncias

Adriano Cabral Linhares Mourthé

Orientador
Carlos Eduardo Mello

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2017

Catalogação informatizada pelo autor

M929

Mourthe, Adriano Cabral Linhares
Um estudo da aplicação do LASSO para seleção de instâncias / Adriano Cabral Linhares Mourthe. -- Rio de Janeiro, 2017.
42

Orientador: Carlos Eduardo Mello.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2017.

1. LASSO. 2. Regressão linear. 3. Seleção de instâncias. 4. Clusterização. 5. Aprendizado de máquina. I. Mello, Carlos Eduardo, orient. II. Título.

Um estudo da aplicação do LASSO para seleção de instâncias

Adriano Cabral Linhares Mourthé

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

Carlos Eduardo Ribeiro de Mello (UNIRIO)

_____

Ana Cristina Bicharra Garcia (UNIRIO)

_____

Sidney Cunha de Lucena (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2017

Resumo

Esta monografia verifica a possibilidade da utilização do LASSO na tarefa de mineração de dados, conhecida como seleção de instâncias. Para isso, foi realizada uma revisão da literatura de aprendizado de máquina e o desenvolvimento de uma modelagem que explora a propriedade do LASSO de gerar soluções esparsas. A modelagem proposta foi validada em três experimentos computacionais.

Palavras-chave: LASSO, regressão linear, seleção de instâncias, clusterização, aprendizado de máquina.

ABSTRACT

This monograph verifies the possibility of using the LASSO in the task of data mining, known as instance selection. This work reviewed the machine learning literature and developed a model capable of exploiting the properties of the LASSO. The proposed modeling was validated in three computational experiments.

Keywords: LASSO, linear regression, instance selection, clustering, machine learning.

Índice

Lista de Tabelas

Lista de Figuras

Lista de siglas e abreviaturas

IS – Instance Selection.

ML – Machine learning.

LASSO - Least absolute shrinkage and selection operator.

LARS - Least Angle Regression.

L_2 - $\|x\|_p = \left(\sum_i |x_i|^P\right)^{1/p}$, onde $p = 2$, também conhecida como norma euclidiana.

L_1 - $\|x\|_p = \left(\sum_i |x_i|^P\right)^{1/p}$, onde $p = 1$.

KDE - Kernel density estimation.

F.D.P - Função densidade de probabilidade.

MSE - Mean squared error.

RBF – Radial basis function.

1 Introdução

1.1 Motivação

Segundo estimativas do pesquisador em inteligência artificial e aprendizado de máquina, Dr. Randal S. Olson, no prefácio do livro *Python Machine Learning*, 2.5 quintilhões (10^{18}) de bytes de dados são gerados diariamente. Só a internet, conforme Jesse Alpert e Nissan Hajaj, no artigo "We knew the web was big..." contém cerca de 1 trilhão de páginas *web* (ALPERT, HAJAJ, 2008). Portanto métodos tradicionais não são suficientes para realizar análise de grandes volumes de dados. Essa escala demanda técnicas automáticas de análise, e o aprendizado de máquina é o campo disciplinar que pode supri-la (MURPHY, 2012, p. 1). A motivação desse trabalho é explorar técnicas de clustering, instance Selection, regressão linear regularizada para o desenvolvimento de uma modelagem capaz de reduzir o volume de dados.

1.2 Objetivos

O objetivo deste trabalho é verificar a possibilidade de aplicação do *LASSO* na tarefa de I*nstance Selection.* Para isso, é feita uma revisão da literatura sobre: modelos lineares regularizados, *Clustering*, *Feature Selection* e *Instance Selection.* E, em seguida, uma avaliação da modelagem proposta em *data sets* sintéticos.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos da seguinte forma:

- Capítulo II: Fundamentação teórica – é feita uma introdução aos conceitos empregados nesta monografia: regressão linear multivariada, *normal equations*, gradiente descendente, *Least angle regression, least absolute shrinkage and selection Operator, Ridge regression,kernel density estimation*, K-Means, K-Medoids.

- Capítulo III: Formalização da aplicação do LASSO para *instance selection*
- Capítulo IV Apresentação dos experimentos, resultados
- Capítulo V: Conclusões e trabalhos futuros

2 Fundamentação Teórica

2.1 Modelos lineares para regressão

A regressão linear simples é uma abordagem para predizer o valor de uma variável de resposta Y baseada em uma variável explicativa x. Ela assume que existe, aproximadamente, uma relação linear entre X e Y(HASTIE; JAMES; WITTEN; TIBSHIRANI; 2013, p 61).

$$\hat{Y} = \hat{\beta}_0 + x_1\hat{\beta}_1 \quad (2.1)$$

Onde, $\hat{\beta}_0$ e $\hat{\beta}_1$ representam, respectivamente, o intercepto, que é conhecido como *bias* no campo de *machine learning* (HASTIE; TIBSHIRANI; FRIEDMAN, 2008, p 11), e o declive. Além disso, $\hat{\beta}_0$ e $\hat{\beta}_1$também são conhecidas como os coeficientes ou parâmetros do modelo.

A regressão linear simples pode ser expandida para acomodar múltiplas variáveis explicativas – Regressão linear múltipla -. Neste caso, o objetivo da regressão é prever o valor da variável explicativa Y dado um vetor X de dimensão D de variáveis de entrada.

$$\hat{Y} = \theta_0 + \sum_{j=1}^{D}(X_j\hat{\beta}) = \hat{\beta}_0 + x_1\hat{\beta}_1 + x_2\hat{\beta}_2 + \cdots + x_d\hat{\beta}_d \quad (2.2)$$

Onde, x_dé a d-ésima variável de entrada, $\hat{\beta}_d$ é o d-ésimo parâmetro da d-ésima variável de entrada. Geralmente, é adicionado uma variável constante 1 no vetor X e $\hat{\beta}_0$ no vetor de coeficientes. Dessa forma, é possível reescrever a equação (1.2) na forma vetorial como um produto interno (HASTIE; TIBSHIRANI; FRIEDMAN, 2008, p 11):

$$\hat{Y} = X^t\hat{\beta} \ (2.3)$$

Onde:

$$X^t{}_{n,d} = \begin{pmatrix} 1_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ 1_{n,1} & \cdots & x_{n,d} \end{pmatrix}, \hat{\beta}_{\,d,1} = \begin{pmatrix} \hat{\beta}_{\,1} \\ \vdots \\ \hat{\beta}_{\,d} \end{pmatrix}, X^t\hat{\beta} = \begin{pmatrix} (x_1)\hat{\beta}_{\,1} \\ \vdots \\ (x_n)\hat{\beta}_{\,d} \end{pmatrix}, \hat{Y}_{n,1} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Na equação anterior, $X^t\hat{\beta}$ recebe o nome de *Design Matrix* (NG,2016).

Segundo os autores do livro The elements of statistical learning, as variáveis de explicativa podem vir de diversas fontes: entradas quantitativas, entradas quantitativas que sofreram uma transformação, funções base, variáveis de entrada categóricas codificadas por meio de variáveis *dummy* e interações entre variáveis de entrada.

2.1.1 Estimativa dos coeficientes do modelo linear

Geralmente, os coeficientes da regressão que melhor preveem os valores da variável alvo são desconhecidos. Portanto, para o modelo linear gerar previsões é preciso estimar os coeficientes que geram o melhor *fit* no conjunto de dados. A abordagem mais comum é conhecida como mínimos quadrados (HASTIE; JAMES; WITTEN; TIBSHIRANI; 2013, p 61). Para isso, é preciso minimizar o erro entre o valor estimado pelo modelo, $\hat{Y}$, e o valor real Y. A função que precisa ser minimizada é conhecida como *residual sum of squares*.

$$RSS(\hat{\beta}) = \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 = e_1^2 + e_2^2 + \cdots + e_n^2 (2.4)$$

A seguir, duas técnicas que calculam os parâmetros β que minimizam a diferença entre $\hat{Y}$ e $\hat{Y}$ são apresentadas.

2.1.2 Equação normal

Esse método consiste em minimizar a função de custo por meio da equação:

$$\frac{\partial RSS}{\partial \hat{\beta}} = 0 \ (2.5)$$

$$\frac{\partial((Y - X\hat{\beta})^t(Y - X\hat{\beta}))}{\partial\theta} = \frac{\partial((Y^tY) - (Y^tX\hat{\beta}) - (\hat{\beta}^tX^tY) - ((\hat{\beta}^tX^t)(X\hat{\beta}))\,)}{\partial\theta} = 0 \ (2.6)$$

$$2X^tY - 2X^tX\hat{\beta} = 0 \ (2.7)$$

$$X^t X\hat{\beta} = X^t Y \ (2.8)$$

$$\hat{\beta} = (X^t X)^{-1}(X^{'} y) \ (2.9)$$

A solução do sistema linear (1.6) é alcançada por meio do cálculo da matriz inversa $(X'X)^{-1}$. No entanto, nem sempre é possível inverter a matriz $(X'X)$. Nesse caso, a matriz pseudoinversa de Moore-Penrose permiti progredir na solução do sistema. Uma matriz pseudoinversa A de Moore-Penrose é definida como (GOODFELLOW; BENGIO; COURVILLE,2016):

$$A^+ = \lim_{\alpha \to 0} (A^t A + \alpha I)^{-1} A^t$$

Os algoritmos para computar a pseudoinversa não usam essa definição, eles usam a fórmula:

$$A^+ = VD^+ U^t \ (2.10)$$

Onde U, D e V são decomposições singulares de A.

2.1.3 Gradiente descendente

Esse algoritmo iterativo de busca inicia com uma conjectura inicial para $\hat{\beta}$ e, a cada passo, muda o valor de $\hat{\beta}$ de forma que o valor da função de custo seja menor do que a da iteração anterior. No caso da função custo com uma variável, o algoritmo utiliza a derivada da função de custo para alterar o valor de $\hat{\beta}$. A figura abaixo demonstra o funcionamento do algoritmo.

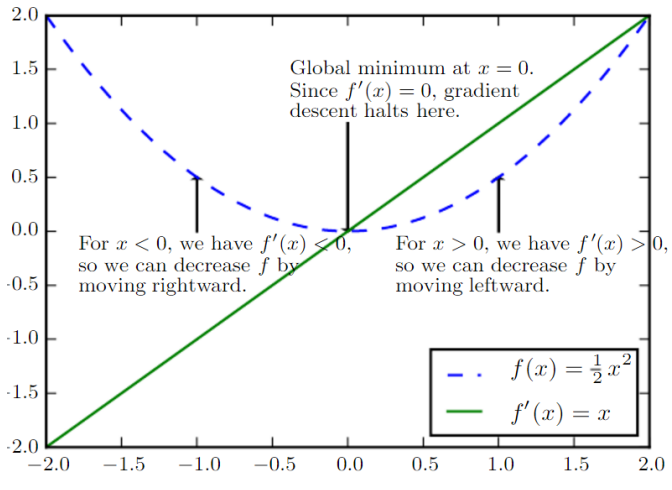

Figura 1 – Gradiente descendente

Fonte: GOODFELLOW; BENGIO; COURVILLE,2016, p.83

Portanto, a saída do algoritmo será um parâmetro que minimiza a função de custo. Entretanto, como pode ser observado na figura abaixo, o gradiente descente não garante que o mínimo encontrado seja global.

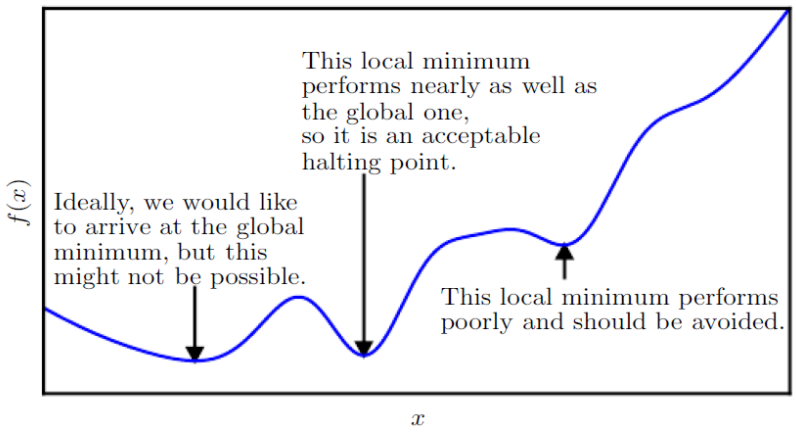

Figura 2 – Mínimo global e mínimo local

Fonte: GOODFELLOW; BENGIO; COURVILLE,2016, p.85

Já no caso multivariado, $f : \mathrm{R}^n \rightarrow \mathrm{R}$, é empregado o vetor gradiente, que contém todas as derivadas parciais e é representado por $\nabla x f\ (x)$. A equação (2.11) é conhecida como *Widrow-Hoff learning rule* (NG,2016). Essa regra é a que atualiza os valores dos parâmetros a cada iteração do algoritmo e é repetida até que todas as derivadas parciais das *features* em relação à $\hat{\beta}$ sejam iguais a zero. A constante α dessa equação recebe o nome de *Learning rate* (BISHOP. 2006), ela é responsável por quantificar o tamanho de cada passo do algoritmo.

$$\hat{\beta} = \hat{\beta} - (\alpha \nabla x f\ (x)) (2.11)$$

2.1.4 Regressão linear regularizada

Uma técnica usada para controlar o fenômeno do *overfitting* é a regularização. Ela envolve adicionar um termo que penaliza grandes coeficientes e, por consequência, diminuem a complexidade do modelo. As técnicas mais populares de regressão linear que usam regularização são: *Ridge regression, Shrinkage, Selection Operator(LASSO)* e *Elastic Net* (SEBASTIAN, Raschka, p. 297).

Ridge regression ou penalized least square encolhe os coeficientes da regressão linear por meio da imposição de uma penalidade aos seus tamanhos. Isso é feito pela

adição de uma regularização L_2 no RSS. No contexto de redes neurais essa prática é conhecida como *weight decay*(MURPHY, 2012, p. 226).

$$\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 + \lambda \sum_{j=1}^{d} {\beta_j}^2 = RSS + \lambda \sum_{j=1}^{d} {\beta_j}^2 \quad (2.12)$$

λ é o coeficiente que governa a importância do termo de regularização em relação à soma dos erros quadrados, e quanto maior o seu valor, maior o fator de *Shrinkage* (TREVOR; TIBSHIRANI; FRIEDMAN,2008, p. 63). A próxima imagem demonstra a interpolação realizada por meio da *ridge regression* com um polinômio de grau 15 em um conjunto de dados, que foram gerados pela função $\cos(1.5e\pi x)$, e o efeito da regularização no erro de validação.

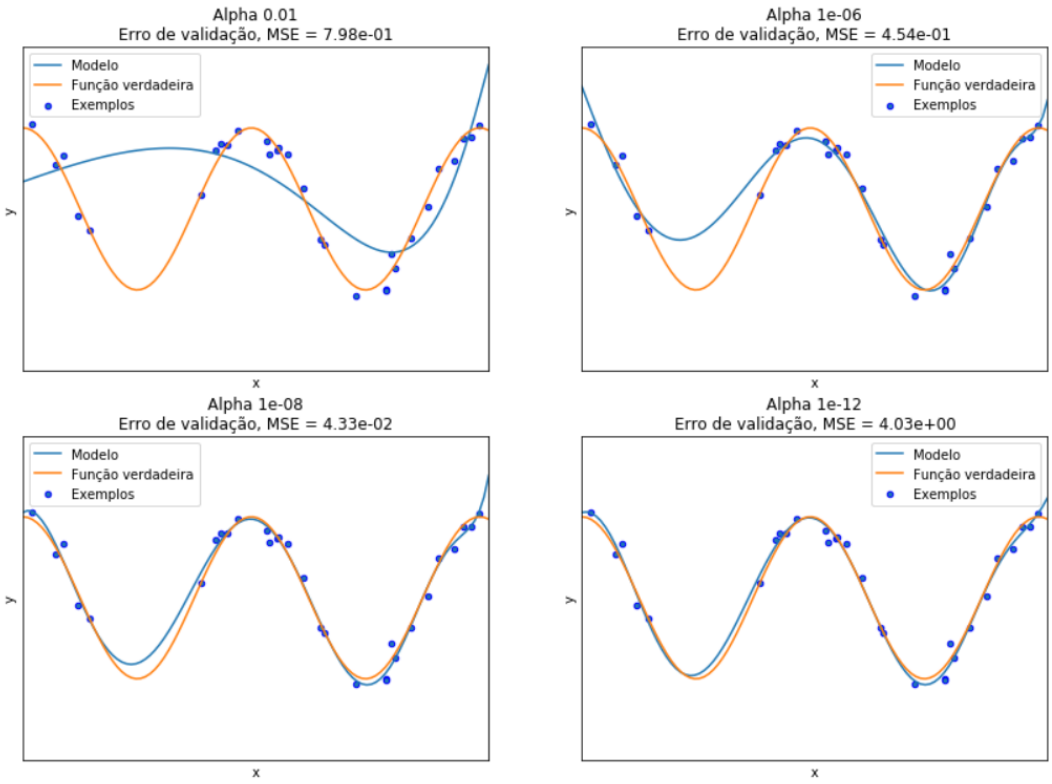

Figura 3 – Variação da regularização

No caso do LASSO, seus coeficientes minimizam a seguinte função (HASTIE; JAMES; WITTEN; TIBSHIRANI; 2013, p 219):

$$\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 + \lambda \sum_{j=1}^{d} |\beta_j| = RSS + \lambda \sum_{j=1}^{d} |\beta_j| \quad (2.13)$$

O *LASSO* e a r*idge regression* possuem formulações semelhantes, a diferença está no fator de regularização. A próxima imagem demonstra o efeito da regularização do *LASSO* nos coeficientes da regressão. Para essa demonstração foi usado o conjunto de dados conhecido com friedman1. Esse conjunto de dados é composto por mil pontos gerados a partir de n variáveis independentes que são uniformemente distribuídas entre [0,1]:

$$y = 10\ sin(\pi x_1 x_2) + 20(x_3 - 0.5)^{\ 2} + 10x_4 + 5x_5 + \varepsilon (2.14)$$

Onde, o termo ε tem uma distribuição $N{\sim}(0,1)$

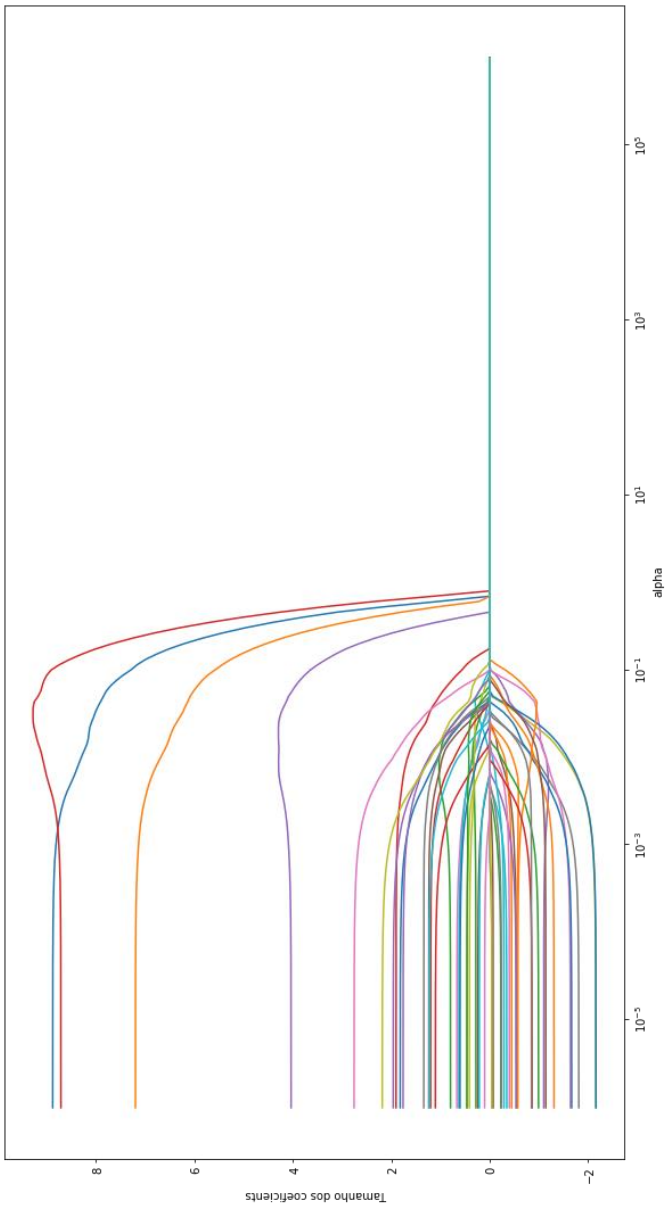

Figura 4 – Caminho do LASSO

Na figura a seguir, a área azul representa a região de restrição, o mapa de contorno em vermelho representa função custo *Least squares* e o ponto de contato entre a elipse e a área azul é o coeficiente da regressão regularizada.

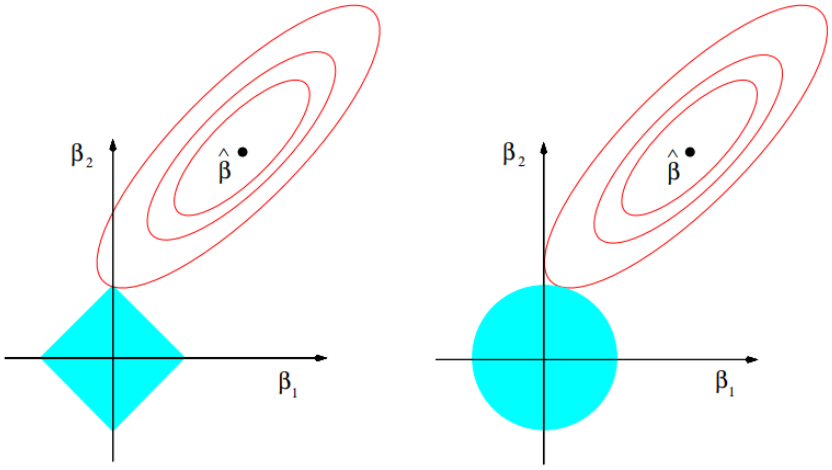

Figura 5 – Região de restrição

TREVOR; TIBSHIRANI; FRIEDMAN,2008, p.71

A figura evidencia uma importante propriedade do *LASSO*, ele é capaz produzir modelos esparsos para determinados valores de λ (BISHOP. 2006, p. 145). No livro *Machine learning: a probabilistic perspective*, o autor explica que de acordo com a teoria da otimização com restrição, a solução ótima ocorre no ponto de interseção do nível mais baixo da função objetivo com a superfície de restrição. Geometricamente, é possível observar que a l_1 *ball* deve crescer até encontrar a função objetivo e que os cantos da l_1 *ball*, que correspondem aos eixos das coordenadas, será a região que primeiro fará contato. Logo, o *LASSO* encontrará soluções em que alguns dos coeficientes são iguais a zero, ou seja, esparsa.

Contudo, é preciso determinar o valor de λ que gera o melhor modelo, isso é conhecido como *parameter tuning*. Uma técnica empregada com esse fim é conhecida como *grid search.* Nela o *score* do método de *Cross-Validation* é usado para selecionar o melhor valor para λ. A próxima figura mostra a evolução do MSE de acordo com o crescimento do valor do fator de regularização.

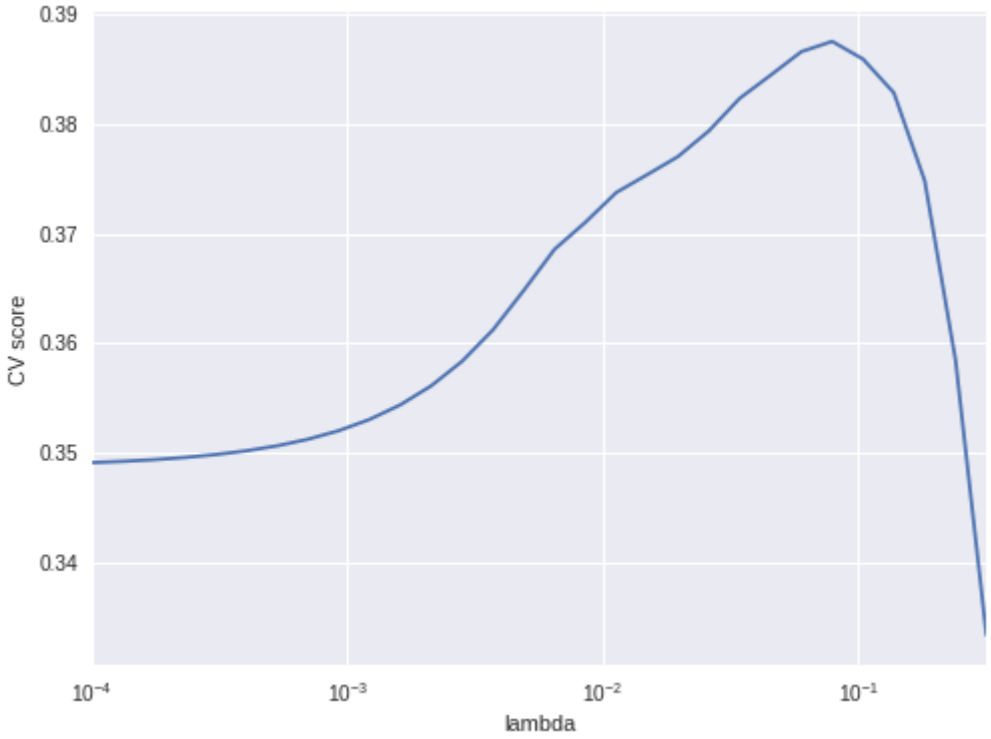

Figura 6 – Cross validation score

Segundo Tibshirani, a solução do *LASSO* por meio do algoritmo *LARS* possui a mesma complexidade computacional dos métodos de solução dos mínimos quadrados. Geralmente, o algoritmo de *OLS* é resolvido com a decomposição Cholesky, que exige $p^3 + np^2/2$ operações, ou pela decomposição QR, que precisa de np^3 operações, onde n é o número de observações e p é o número *features*. (Lawson and Hansen, 1974). Além do *LARS*, existem outras formas de se encontrar os parâmetros do *LASSO*, como, por exemplo, o algoritmo proposto no artigo *Coordinate Descent algorithms for lasso penalized regression*.

2.1.5 LARS e LASSO

De acordo com Tibshirani et al. o *LARS* está intimamente conectado com o *LASSO*, ele provê uma forma extremamente eficiente de se calcular o caminho do *LASSO*. Inicialmente, o algoritmo identifica a variável mais correlacionada com o resíduo. Em seguida, o *LARS* move o coeficiente dessa variável em direção ao seu valor de mínimo quadrado. Quando uma outra variável alcança a mesma correlação da primeira variável, ele para de mover a variável inicial e adiciona a nova variável no conjunto de variáveis ativas. Agora, os coeficientes das duas variáveis são movidos de

forma que suas correlações diminuem de forma conjuta. Esse processo é repetido até que todas as variáveis sejam incluídas no modelo.

Tabela 1 – LARS

1	Normalizar os preditores. Iniciar com o resíduo $r = y - \hat{y}, \beta_1, \beta_2 \ldots, \beta_n = 0$.
2	Encontrar o preditor $x_j$mais correlacionado com r .
3	Mova β_j em direção ao seu valor de Least Squares até que outro x_k tenha tanta correlação com o resíduo quanto x_j.
4	Mova β_je β_k em direção aos seus valores em conjunto de Least Square até que outro x_l tenha tanta correlação com o resíduo quanto x_k e x_j.
5	Continua esse processo até que todos os preditores serem incluídos. Depois de $Min(N-1, D)$ passos, é chegada na solução do Least Squares.

No entanto, o caminho calculado pelo *LARS* não é igual ao do *LASSO*, como pode ser visto no próximo gráfico. Portanto, é preciso fazer uma pequena modificação no *LARS*: caso um coeficiente atinja o valor zero, remova a sua variável do conjunto de variáveis ativas.

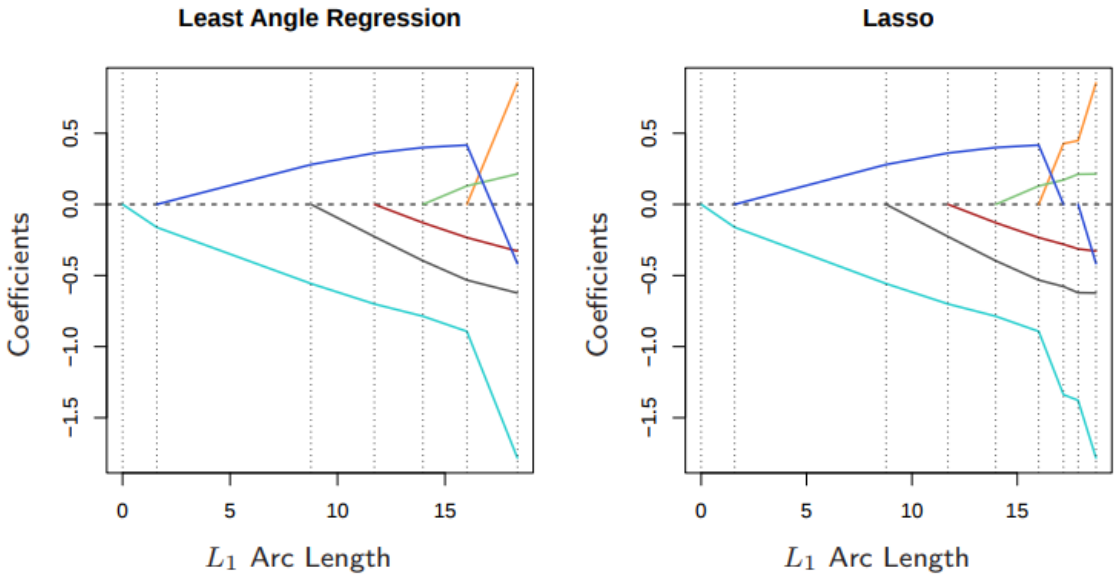

Figura 7 – *LARS* e *LASSO*

2.2 Clustering

A análise de cluster, também conhecida como segmentação de dados, procura segmentar ou agrupar uma coleção de objetos, de forma que os seus elementos sejam mais semelhantes do que os elementos contidos em outros agrupamentos. Além disso, também é possível que essa análise busque por agrupamentos que estejam, naturalmente, hierarquizados. Os algoritmos usados para análise de cluster são categorizados em três tipos: Algoritmos combinatórios, *mixture modeling* e *mode seeking* (HASTIE; TIBSHIRANI; FRIEDMAN, 2008, p 507).

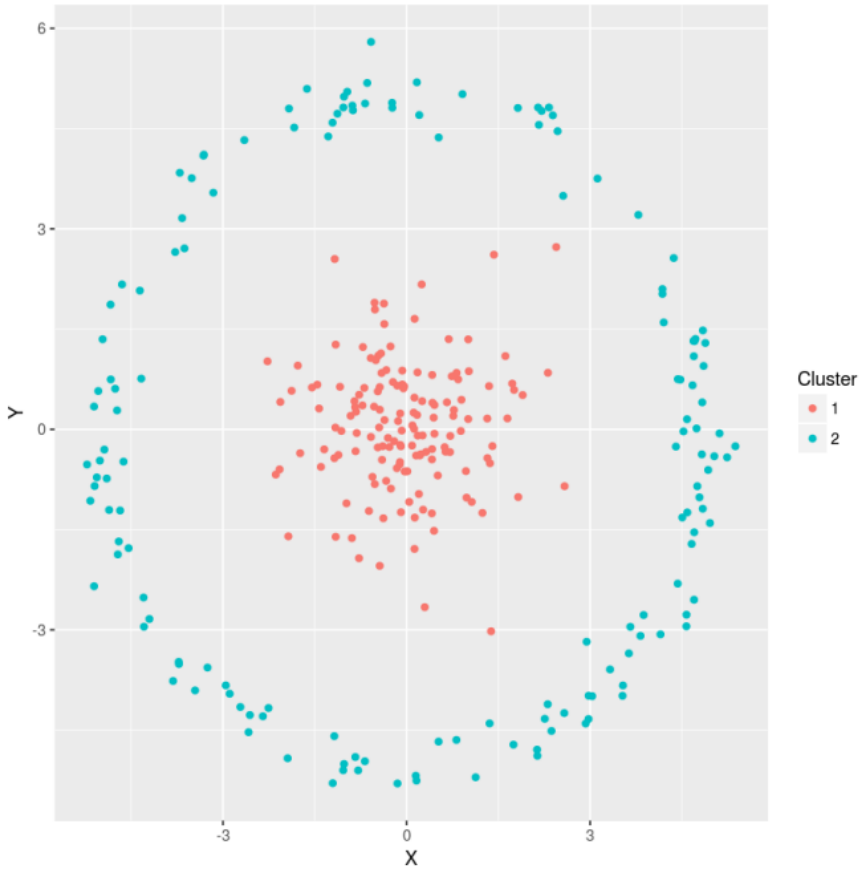

Figura 8 – Clusters

Entretanto, no artigo Why so many clustering algorithms - A Position Paper, Vladimir Estivil Castro, argumenta que, embora a análise de cluster seja uma importante ferramenta de *Data mining*, existem diversas definições na literatura para *cluster*. Por exemplo, ele cita que no livro Cluster Analysis *os* autores definem *clustering* com uma visão *bottom-up*: encontrar grupos em conjunto de dados que sejam definidos por algum critério natural de similaridade. Enquanto isso, R. Duda e P. Har no livro Pattern classification and scene analysis, têm uma definição *top-down* sobre cluster. Segundo

eles, clustering é a segmentação de uma população heterogênica em subgrupos homogêneos.

2.2.1 Algoritmos combinatórios

O algoritmo associa cada observação a um e somente um cluster, e não leva em consideração o modelo probabilístico que descrevem os dados. (HASTIE; TIBSHIRANI; FRIEDMAN, 2008, p 508). Também é preciso que seja predeterminado o número de Clusters.

2.2.1 *K-Means*

Esse algoritmo de *clustering* é aplicado quando todas as variáveis são contínuas e o propósito da tarefa de mineração de dados é particionar um conjunto de dados em *K clusters* distintos, que não se sobrepõem. De acordo com o livro introduction to statistical learning, a ideia fundamental do *K-means* é buscar por agrupamentos que possuam pouca variância *intracluster* $W(C_k)$:

$$\min_{C1,..Ck} \left\{ \sum_{k=1}^{k} W(C_k) \right\} \quad (2.14)$$

Para resolver a equação anterior é preciso definir matematicamente $W(C_k)$. A definição mais comum envolve a distância euclidiana quadrada:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \epsilon C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \quad (2.15)$$

Onde, $|C_k|$ é o número de pontos pertencentes ao k-ésimo cluster. Ou seja, a variância intracluster é a soma de todas as distâncias euclidianas quadradas entre os pontos pertencentes a cada um dos seus respectivos clusters dividido pelo número de elementos de cada cluster. A combinação das duas equações anteriores resulta no problema de otimização do *k-means*.

$$\min_{C1,..Ck} \left\{ \sum_{k=1}^{k} \frac{1}{|C_k|} \sum_{i,i' \epsilon C_k} \sum_{j=1}^{p} \left(x_{ij} - x_{i'j}\right)^2 \right\} \quad (2.16)$$

O algoritmo do *K-means* é capaz de encontrar um ótimo local para esse problema de otimização sem a necessidade de testar todas as K^n formas que podemos particionar n observações em k clusters.

Tabela 2 – K-Means

1	Atribua valores aleatórios entre 1 e *k* para cada observação.
2	Repita até que os *clusters* atribuídos parem de mudar
2.1	Para cada um dos *k clusters*, calcule o seu respectivo centroide. O centroide k-ésimo centroide é um vetor de médias de *d features* das observações que pertencem o *clusters k*.
2.2	Atribua a cada observação o *cluster* que possuir o centroide mais próximo.

A próxima imagem demonstra o resultado do *K-Means* em um conjunto de dados que contém quatro bolhas gaussianas isotrópicas com desvio padrão igual a um.

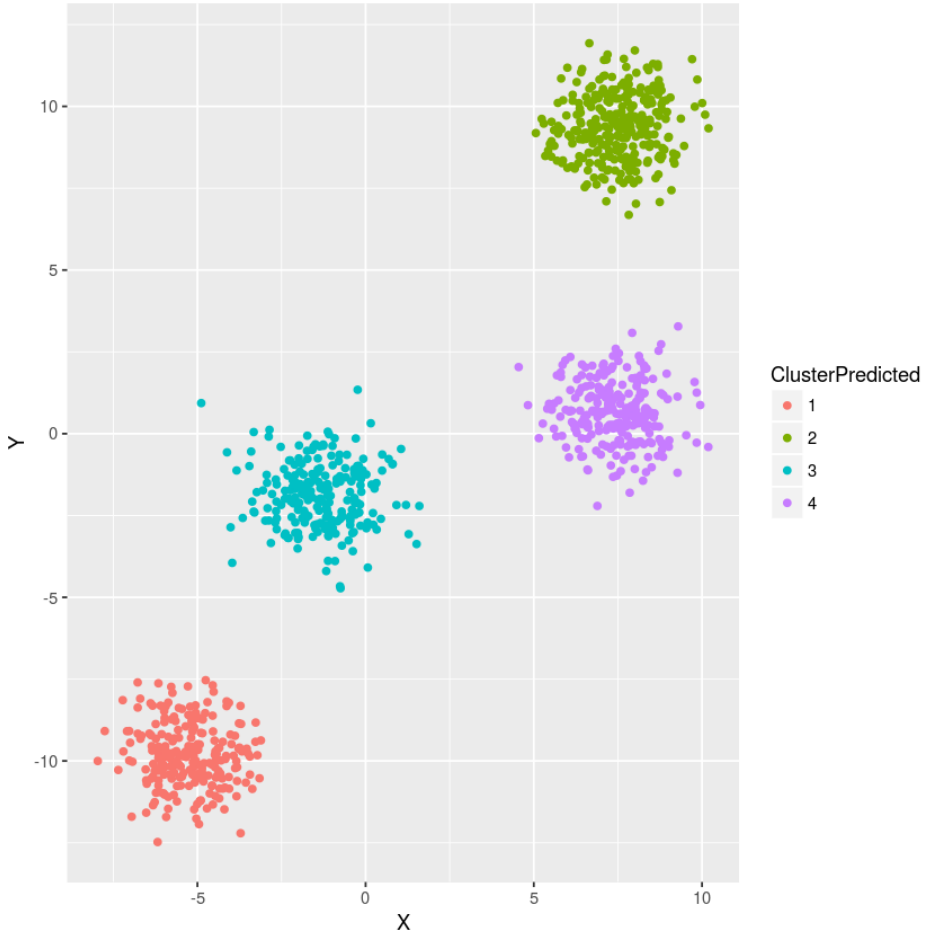

Figura 9 – K-means

2.2.2 K-Medoids

O K-means por usar a média, que é uma estatística não robusta, é sensível a *outliers*. Portanto, pontos que estejam distantes dos *clusters* podem distorcer o valor da variância intra*cluster* e alterar a saída do algoritmo. O K-Medoids procurar resolver essa sensibilidade por meio do uso de instâncias mais representativa no lugar da média **(**HAN**,** KAMBER, PEI, p 455). Desse modo, cada ponto é atribuído ao cluster que possui a instância representativa mais semelhante. Outra modificação é critério de que precisa ser minimizado, no lugar da distância euclidiana quadrada é usado o erro absoluto em relação ao ponto mais representativo do cluster:

$$E(C_k) = \sum_{i=1}^{k} \sum_{p \epsilon c1} |x' - x_{ip}| \quad (2.17)$$

Onde, x' é o elemento mais representativo do *cluster* e $E(C_k)$ a soma de todas as diferenças absolutas entre o elemento mais representativo e os elementos contidos no cluster k.Entretanto, quando $k > 1$ o problema do K-Medoids se torna *NP-Hard.* Um algoritmo guloso popularmente usado para tratar esse problema é *Partitioning Around Medoids – PAM*.

Tabela 3 – K-Medoids

1	Escolha arbitrariamente k pontos como representantes iniciais dos clusters, m.
2	Atribua a cada ponto não escolhido o cluster que possuir o ponto m mais próximo.
3	Enquanto $E(C_k)$ diminuir, repita:
3.1	Para cada ponto m e para cada outro ponto p:
3.1.1	Troque m por p e calcule $E(C_k)$.
3.1.2	Se o novo custo for menor, troque m por p.

2.3 *Model selection e Model assessment*

O poder de generalização de um método de aprendizado de máquina está relacionado com a sua capacidade de previsão em um conjunto de dados de teste. A avaliação desse desempenho é importante, porque ela guia a escolha do modelo adequado, *Model selection*, e mensura a qualidade do modelo, *Model assessment*. Para isso, em situações que o conjunto de dados seja grande, a melhor abordagem é separar o conjunto de dados em três partes: conjunto de treino, conjunto de validação e conjunto

de teste. Nessa divisão, o conjunto de treino é usado para *fit* o modelo, o conjunto de validação serve para M*odel selection* e a validação para M*odel assessment.* (HASTIE; JAMES; WITTEN; TIBSHIRANI; 2013, p 219-222).

Abu-Mostafa e Magdon-Ismail no livro Learning From Data afirmam que o uso mais importante do conjunto de validação é para *model Selection,* porque isso permite que os parâmetros de um modelo possam ser escolhidos. Por exemplo, no caso de uma regressão linear, ele pode ser empregado para escolher o grau do polinômio que será usado para *fit o* modelo linear. A próxima imagem demonstra o impacto do grau do polinômio.

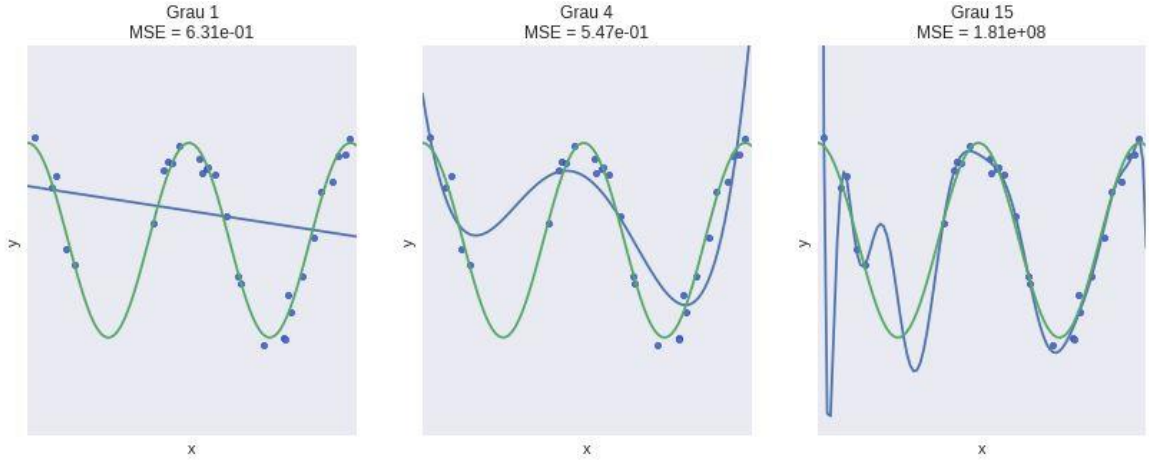

Figura 10 – Polinômios de graus distintos

A imagem também revela dois outros importantes fatores que devem ser considerados, o *overfitting e o underfitting.* O primeiro quadro é um exemplo de *underfitting*, o modelo é demasiadamente simples e não é suficientemente flexível para realizar a interpolação. Já no último quadro, o ocorre a situação oposta, o *overfitting*, o modelo é muito complexo e tem o maior erro de validação.

2.4 *Feature Selection*

Quando algoritmos de *machine learning* e *data minning* são aplicados em dados com alta dimensão, uma questão fundamental é conhecida como *curse of dimensionality* (BELLMAN, 1961). Os dados se tornam esparsos em espaços de alta dimensão, e, portanto, algoritmos que foram feitos para lidar com espaço de baixas dimensões podem ser afetados. Pedro Domingos no artigo A Few Useful Things to Know about Machine Learning expõe um exemplo da *curse of dimensionality:* Um conjunto de dados com cem dimensões e um trilhão de exemplos cobre somente a fração de 10^{-18} do total do

input space. Uma forma de lidar com essa questão é com técnicas de redução de dimensão. Essas técnicas podem ser divididas em duas categorias*: feature extraction* e *feature Selection.* A técnica de *feature extraction* projeta os dados originas em um espaço de dimensão menor. O novo espaço construído é, geralmente, uma combinação linear ou não linear do espaço original das features. *Principal componentes analysis* e *Singular value decomposition* são exemplos de duas técnicas dessa categoria. Já os algoritmos da outra categoria selecionam o subconjunto de *features* mais relevantes para a construção do modelo. *LASSO*, *information gain* e *Fisher score* são membros dessa última categoria.

2.5 Seleção de instancias

Segundo Salvador Garcia et al. no livro Data preprocessing in Data Mining, *instance selection* consiste em selecionar um subconjunto de dados de forma que o resultado da aplicação de data mining seja igual ao resultado obtido com a totalidade dos dados. Ou seja, $P(DM, s) = P(DM, t)$, onde s é o subconjunto mínimo selecionado, P é a performance da aplicação de data mining, t é o conjunto de completo e DM é o algoritmo de data mining.

2.6 Estimativa de densidade

Estimar a função de densidade de probabilidade de uma distribuição contínua a partir de amostra representativa é um problema de importância fundamental nas áreas de aprendizado de máquina e reconhecimento de padrão (GIROLAMI, HE, 2003, p. 1).

Segundo Ian Goodfellow (Goodfellow, 2016, p.99), nesse problema um algoritmo aprende uma função $p_{model}: R^n \rightarrow R$, onde $p_{model}(x)$ pode ser interpretado como uma função de densidade de probabilidade ou como função de probabilidade de massa do espaço em que x foi amostrado.

Bishop, em seu livro *Pattern Recognition and Machine Learning* (p. 68), aborda esse problema como um *ill-posed problem*, pois podem existir infinitas distribuições de probabilidade capazes de gerar os dados observados.

As distribuições que são governadas por parâmetros adaptativos são chamadas de distribuições paramétricas. Por exemplo, no caso da distribuição gaussiana, os parâmetros são: a média e a variância. A aplicação desses modelos para estimar a

densidade necessita de um procedimento capaz de determinar quais são os melhores parâmetros para os dados observados. A seleção dos melhores parâmetros, seguindo uma abordagem Frequentista, é realizada por meio da otimização de um critério, como, por exemplo, a função de verossimilhança. Uma limitação dessa classe de modelos é que a distribuição de probabilidade escolhida pode não ser apropriada para os dados amostrados. A título de exemplo, se o processo que gera os dados é multimodal, então uma distribuição gaussiana nunca irá capturar essa característica, pois ela é necessariamente unimodal (BISHOP, 2006, p.120).

Um método alternativo é a estimativa de densidade não paramétrica.Um exemplo desse método é o histograma. Ele divide a variável aleatória contínua x em *bins* de tamanho Δ_i e, em seguida, conta o número de observações que pertencem a cada *bin*. Portanto, a estimativa da densidade normalizada será:

$$p_i = \frac{n_i}{\mathrm{N}\Delta_\mathrm{i}} (2.18)$$

No livro Advanced Data Analysis from an Elementary Point of View, Cosma Rohilla Shalizi define a escolha do tamanho adequado de bin como uma instância do problema de *bias-variance trade off*. Segundo o autor, a escolha de um número muito grande de pequenos *bins* faz uma estimativa com viés pequeno e com uma grande variância, e, no caso de poucos *bins* de tamanho grande, a estimativa terá um grande viés e uma pequena variância.

2.6.1 Kernel Density Estimation

Essa abordagem assume que N dados foram amostrados a partir de uma função de densidade de probabilidade desconhecida $p(x)$ e é necessário estimar o valor de $p(x)$. Para isso, é considerado o número de amostras que caem em uma Região R, também conhecida como janela. A probabilidade de massa da região R é dada por:

$$P = \int_R p(x)dx \, (2.19)$$

É possível modelar os dados amostrados que pertencem a essa região, K pontos, por meio da distribuição binomial:

$$Binomial(K|N,P) = \frac{N!}{K!\,(N-K)!} P^k (1-P)^{n-K} (2.20)$$

Com a equação do valor esperado da distribuição binomial é possível concluir que a razão média dos pontos que caíram dentro dessa região é $E[K] = nP$, ou seja, para uma grande quantidade de pontos:

$$K \approx \mathrm{NP}(2.21)$$

Além disso, a região R precisa ser suficientemente pequena para que a sua densidade de probabilidade seja constante, é possível concluir que:

$$P \approx \mathrm{p(x)V}(2.22)$$

Para estimar a probabilidade de $p(x)$ é preciso combinar as duas equações anteriores

$$p(x) = \frac{K}{NV}(2.23)$$

O *kernel density estimation* assume que o volume dos *bin* seja o mesmo para todas as regiões.

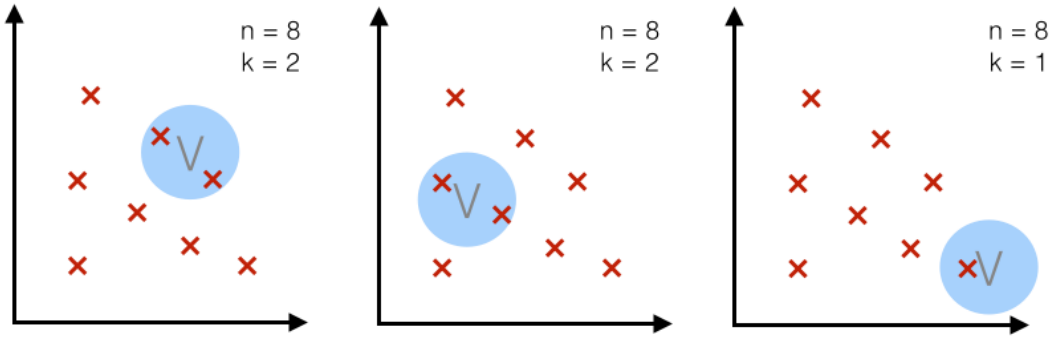

Figura 11 – KDE

Fonte: Python Machine Learning, Sebastian

A região R pode ser considerada como um hipercubo de tamanho unitário e, dessa forma, a probabilidade associada a cada região pode ser calculada a partir do número de pontos que pertencem a essa região.

$$\phi(u) = \begin{cases} 1, & |u_i| \leq 1/2, \quad i = 1, .., D \\ 0, & Caso\ contrário \end{cases} (2.24)$$

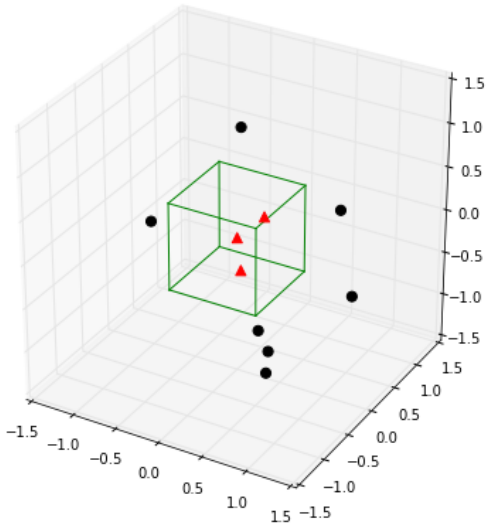

Figura 12 – Pontos que pertencem ao hipercubo R

Fonte: Python Machine Learning, Sebastian

A função de janela $\phi(u)$, que é um exemplo de uma função kernel conhecida como Janela de *parzen*, atribui o valor de 1 para os pontos que caem dentro do hipercubo e 0 caso contrário. A equação anterior pode ser estendida para os casos de hipercubos de qualquer tamanho e que são centrados ao redor do ponto x_i

$$\phi\left(\frac{x - x_i}{h}\right) = \begin{cases} 1, & \frac{x - x_i}{h} \leq 1/2, \quad i = 1,..,D \\ 0, & Caso\ contrário \end{cases} \quad (2.25)$$

Portanto, o número total de k amostras contidas na região R será:

$$k = \sum_{n=1}^{N} \phi\left(\frac{x - x_i}{h}\right) \quad (2.26)$$

Substituindo a equação na equação:

$$p(x) = \frac{1}{N} \sum_{1=1}^{N} \frac{1}{h^d} \phi\left(\frac{x - x_i}{h}\right) \quad (2.27)$$

Onde, h^d é o volume do hipercubo de dimensão D e de aresta de tamanho h. Entretanto, o emprego de um hipercubo gera problemas de continuidade na fronteira do hipercubo. Para contornar essa limitação, e obter uma modelo de densidade mais suave, a função de janela pode ser substituída por uma janela gaussiana – kernel gaussiano. (BISHOP, 2006, p.120).

$$p(x) = \frac{1}{N}\sum_{1=1}^{N} \frac{1}{h^d}\phi\left[\frac{1}{(\sqrt{2\pi}\,)^d h^d} exp\left[-\frac{1}{2}\left(\frac{x-x_i}{h}\right)^2\right]\right] \quad (2.28)$$

3 **Proposta de trabalho**

Neste trabalho de conclusão de curso é proposto usar a propriedade do operador *LASSO* de gerar modelos esparsos para a seleção de instâncias em um conjunto de dados $D_n\{(x_j)\}_{j=1}^{n}$, $x \epsilon \mathbb{R}$. Para isso, foi desenvolvida uma modelagem que trata as densidades estimadas pelo *KDE* como uma combinação linear de janelas de *parzen* com os coeficientes do *LASSO*. Nesta modelagem, a densidade estimada pelo *KDE* é escrita como:

$$\mathrm{p} \;\; = \beta_1\phi(x_1,x_1) + \beta_2\phi(x_1,x_2) + \cdots + \beta_n\phi(x_1,x_n) \quad (3.1)$$

Onde,

$$\phi(x,y) = \exp\left(\frac{(x-y)^2}{2\,\sigma^2}\right) \quad (3.2)$$

É possível escrever a matriz de design como:

$$\phi_{n,n} = \begin{bmatrix} \beta_1\phi(x_1,x_1) & \cdots & \beta_n\phi(x_1,x_n) \\ \vdots & \ddots & \vdots \\ \beta_1\phi(x_n,x_1) & \cdots & \beta_n\phi(x_n,x_n) \end{bmatrix} \quad (3.3)$$

Organizando as equações, o problema pode ser escrito da seguinte maneira:

$$\begin{bmatrix} \phi(x_1,x_1) & \cdots & \phi(x_1,x_n) \\ \vdots & \ddots & \vdots \\ \phi(x_n,x_1) & \cdots & \phi(x_n,x_n) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \quad (3.4)$$

Em seguida, é possível escrever a função de custo que precisa ser minimizada para obtenção dos coeficientes.

$$\left(\sum_{i=1}^{n}(P(x_i) - \sum_{j=1}^{n}(\beta_j\,\phi(x_i,x_{ij}))\right)^2 + \lambda\sum_{j=1}^{n}|\beta_j| \quad (3.5)$$

Esse problema de otimização pode ser resolvido por meio do algoritmo *LARS*. Como discutido nos capítulos sobre o *LASSO* e *LARS*, a regularização L_1 tem a propriedade de

gerar modelos esparsos. As variáveis não nulas são as que possuem a maior correlação absoluta com o resíduo, nesse caso a diferença a densidade estimada e a densidade verdadeira. Dessa maneira, os pontos selecionados serão aqueles que seus respectivos coeficientes não foram zerados.

Tabela 4 – Seleção de instâncias com o LASSO

1	Montar a matriz de kernel com uma banda h.
2	Usar *KDE*, com uma banda $2h$, para estimar a F.D.P.
3	Executar o *LASSO* com a F.D.P estimada como variável alvo e a matriz de kernel como a matriz contendo as variáveis regressoras.
4	Selecionar as variáveis que não tiveram seus respectivos coeficientes zerados.

É importante destacar que a banda do kernel para estimar o *KDE* tem o dobro da largura empregada na matriz de kernel. Esse aumento acontece, devido a convolução de gaussianas. A gaussiana resultante possui uma variância igual à soma das variâncias das gaussianas originais. Esse fenômeno é denominado de Autossimilaridade (Moo K. Chung, 2007).

$$g_{resultante}(\mu, \sigma_1^2 + \sigma_2^2) = g_1(\mu, \sigma_1^2) g_1(\mu, \sigma_2^2) \quad (3.6)$$

Como a gaussiana é uma função autossimilar e a convolução de uma gaussiana é uma operação linear, a convolução de um kernel gaussiano seguido da convolução com outro kernel gaussiano é equivalente a uma convolução com um kernel mais largo (Moo K. Chung, 2007).

4 **Resultados**

Para a realização dos experimentos foi utilizada uma máquina virtual alugada no serviço *Amazon Web Services*. Essa *VM* conta com um processador Intel Xeon® E5-2686 v4, que pertence à família Broadwell, de 2,3 GHz, 16 Gigabytes de memória ram e com o sistema operacional *Ubuntu* 16.04.3 *LTS*. Os scripts usados para gerar os experimentos foram escritos na linguagem Python 3.6. As implementações empregadas dos algoritmos *LASSO, K-Mean, PAM e Mean Shift* são as da biblioteca *Scikit Learn.* Já a implementação do *KDE* multivariado pertence ao *Scipy.*

4.1 Experimento 1

Os primeiros experimentos foram realizados em um conjunto de dados de 2 mil pontos, que são constituídos por quatro gaussianas de duas dimensões com desvio padrão igual a um. Com esse experimento, é esperado demonstrar que o algoritmo proposto seja capaz de selecionar a moda das gaussianas.

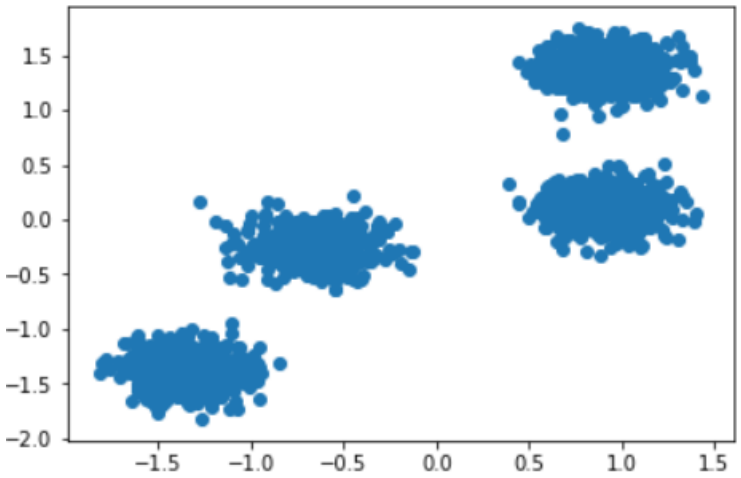

Figura 13 – Experimento 1, Scatter plot

Além disso, a tabela a seguir apresenta os parâmetros que foram usados como entrada do algoritmo:

Tabela 5 – Parâmetros experimento 1

Fator de regularização do *LASSO*	1e-3
Banda do *kernel RBF*	1.5
Banda do *kernel* usado no *KDE*	0.75

Os resultados obtidos estão na próxima tabela:

Tabela 6 – Instâncias selecionadas

LASSO	***K-Means***	***PAM***	***Mean Shift***
[-0.64261341, 0.26519153]	[0.90203742,1.3800148]	[0.90913615,1.38256955],	[0.90203742 ,1.3800148]
[0.86952835, 0.15551689]	[0.89998468,0.11752002]	[0.9026445, 0.12157498],	[0.89998468 ,0.11752002]
[-1.32872037, -1.37335779]	[-1.33841474,-1.3844636]	[-1.35495821,-1.38859737],	[-1.33841474 -1.38446365]
[0.92046701, 1.34700841]	[-0.65608153,-0.2525081]	[-0.65005061, -0.25196188]	[-0.65608153 -0.25250819]

Os pontos em vermelho foram as instâncias selecionadas pelo *LASSO.*

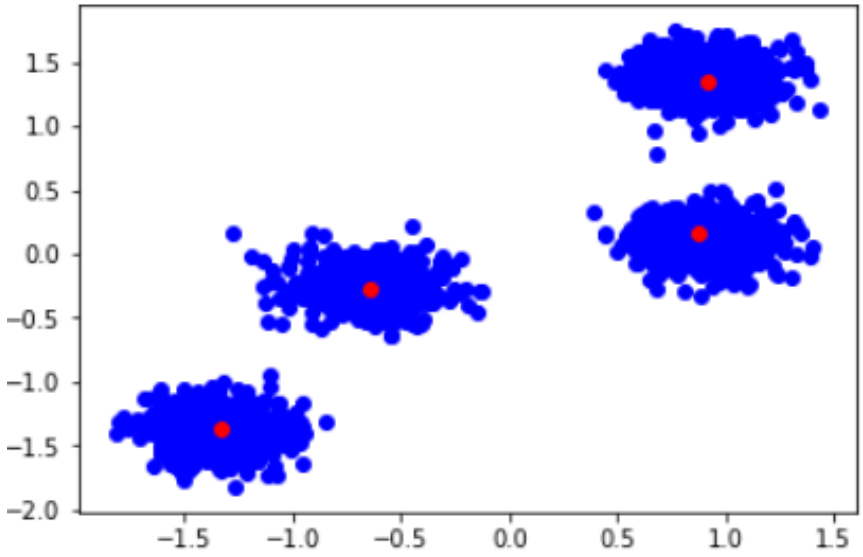

Figura 14 – Experimento 1, instâncias selecionadas

O gráfico a seguir é um *scatter plot* com os pontos selecionados por cada algoritmo. Nele os pontos pretos representam o *Mean shift,* o vermelho, o *LASSO,* o amarelo, o *PAM* e o verde, o *K-means.*

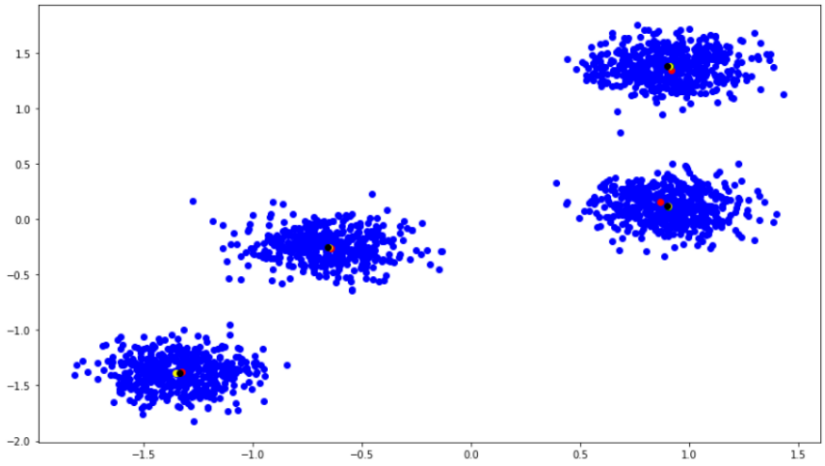

Figura 15 – Experimento 1, algoritmos de clustering

Para os algoritmos do *K-Means* e *PAM* foi dado como entrada o número correto de *clusters*, e para o *meanshift* foi usada a mesma largura de banda do *KDE*. Também é possível analisar a evolução do valor dos coeficientes do *LASSO* em relação à magnitude da norma $l1$ da restrição, e verificar que, inicialmente, aparece um coeficiente e, logo em seguida, três outros coeficientes.

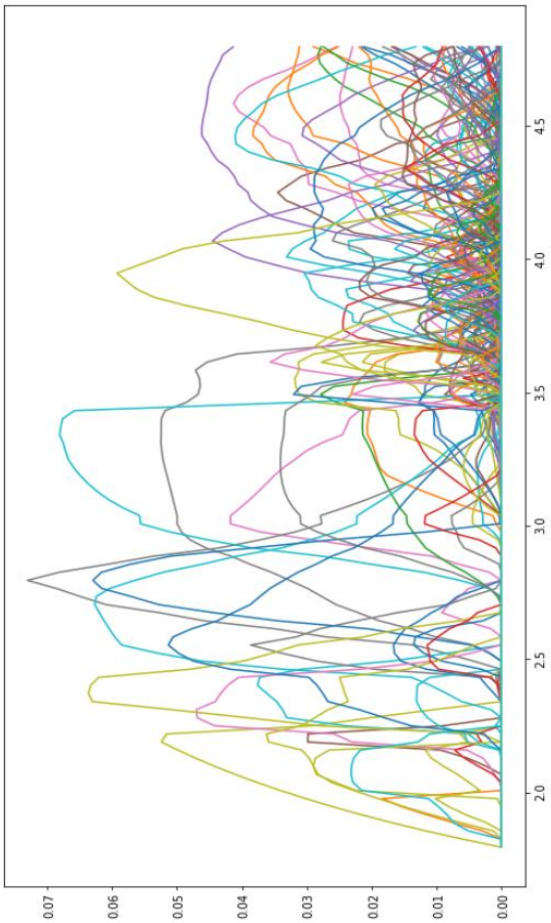

Figura 16 – Experimento 1, caminho dos coeficientes

4.2 Experimento 2

No experimento seguinte, os 3 mil pontos foram gerados com as funções:

$$x = tcos(t),$$

$$y = tsin(t),$$

onde t foi amostrado de uma distribuição uniforme com limites entre 0 e 1. Para isso, foi usada a função do pacote *numpy* $random.rand$:

$$t = 1.5\,\pi(1 + 3numpy.random.rand(1, (número\ de\ pontos))$$

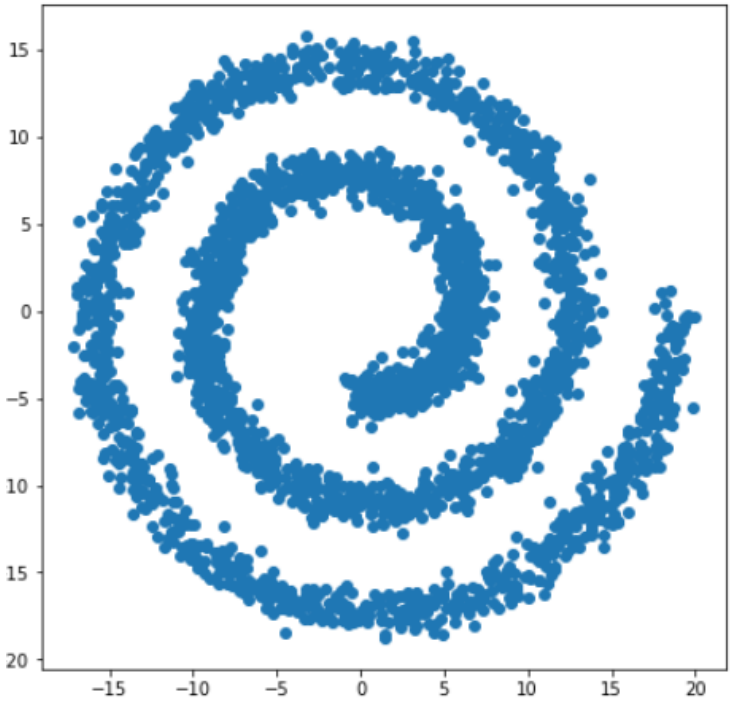

Figura 17 – Experimento 2, Scatter plot

Nesse exemplo foi verificado se a banda usada no *KDE* é adequada para que seja possível capturar a estrutura do dado. A próxima imagem demonstra o resultado do *KDE* com a banda de tamanho igual a 0.1.

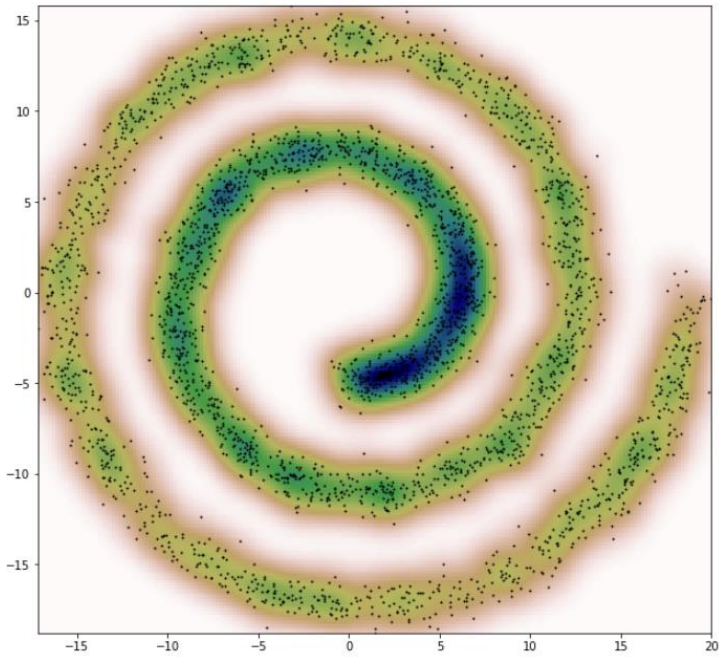

Figura 18 – Experimento 2, densidade estimada

Após definir o tamanho da banda para o *KDE*, o algoritmo foi executado com os seguintes parâmetros:

Tabela 7 – Parâmetros experimento 2

Fator de regularização do *LASSO*	1e-6
Banda do *kernel RBF*	0.2
Banda do *kernel* usado no *KDE*	0.1

O resultado alcançado mostra que para o algoritmo é capaz de reduzir o número de pontos, mas mantendo a estrutura original do conjunto de dados, foram selecionado s somente 81 pontos do total de 4 mil. Também é importante destacar que todos os coeficientes resultantes, diferentes de zero, foram positivos.

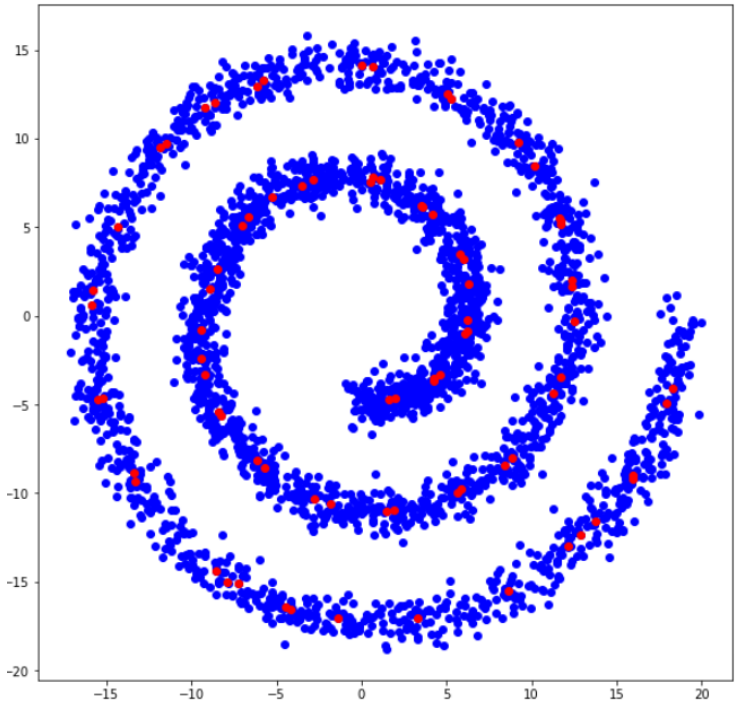

Figura 19 – Experimento 2, pontos selecionados

A imagem a seguir mostra os pontos selecionados sobrepostos à densidade estimada pelo KDE.

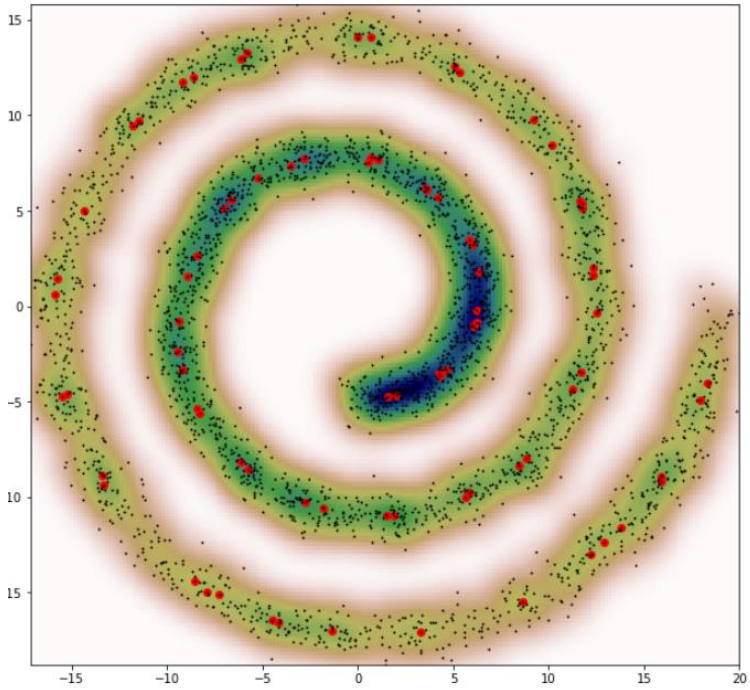

Figura 20 – Experimento 2, pontos selecionados e densidade estimada

Enquanto isso, o M*ean shift* fez a seleção de 2518 centroides, que estão em verde na próxima imagem. O valor da banda foi igual ao usado para estimar o *KDE*.

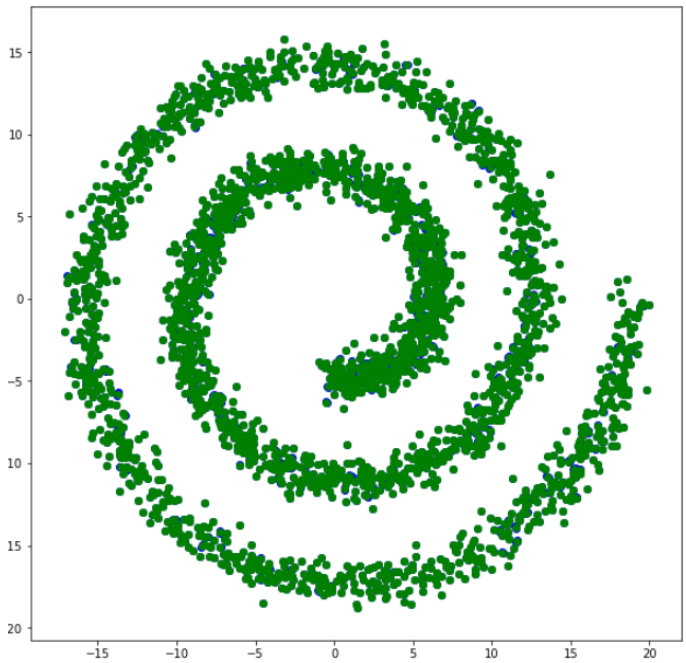

Figura 21 – Experimento 2, pontos selecionados pelo Mean shift

Outra comparação interessante é verificar o resultado do *PAM*, os *medoids* selecionados estão em verde no próximo gráfico, quando o parâmetro referente ao número de clusters é supervisionado pelo número de pontos selecionados pelo *LASSO*.

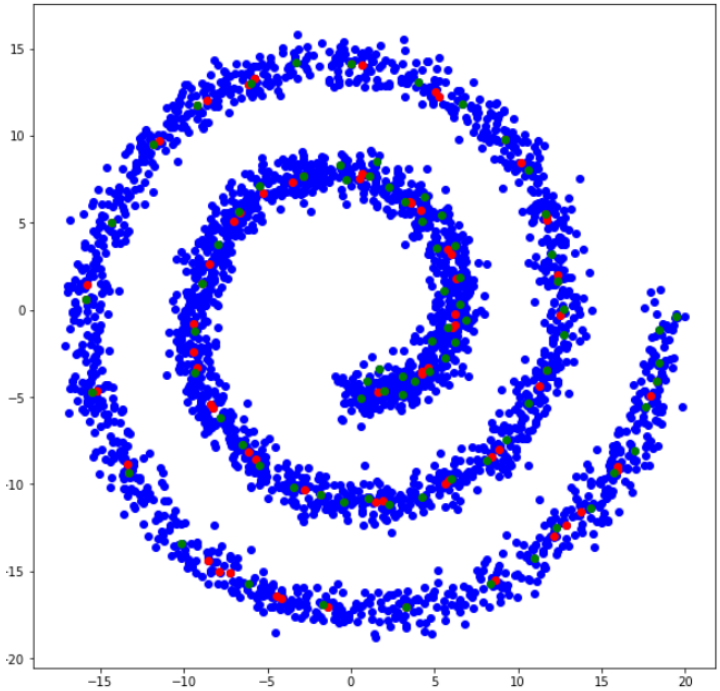

Figura 22 – Experimento 2, pontos selecionados pelo PAM e LASSO

4.3 Experimento 3

Para o terceiro experimento foram gerados dados a partir de duas distribuições gaussianas.

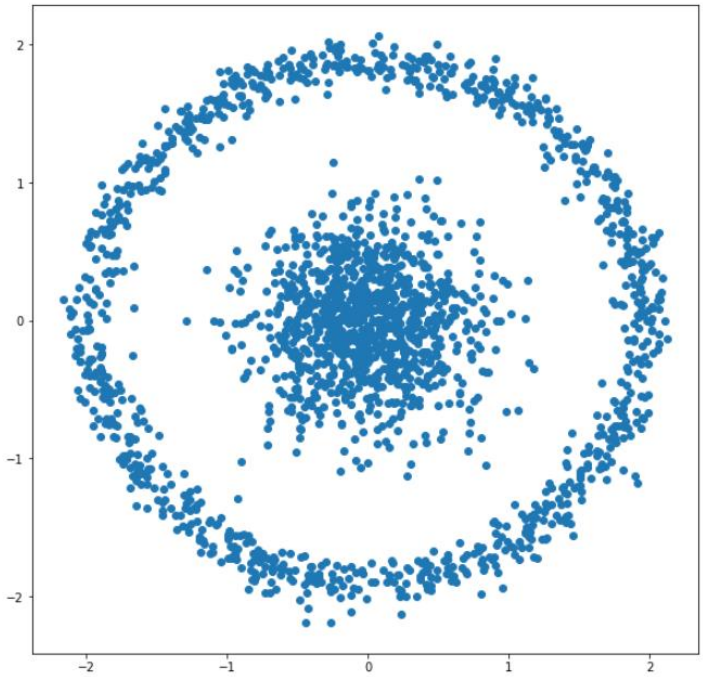

Figura 23 – Experimento 3, Scatter plot

A próxima imagem demonstra o resultado do *KDE* com a banda de tamanho igual a 0.6

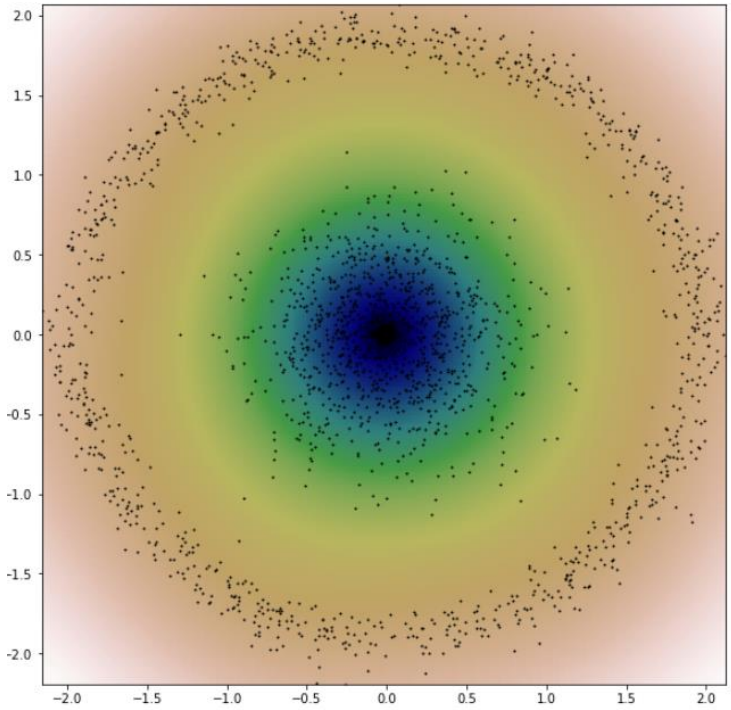

Figura 24 – Experimento 3, KDE com banda = 0.6

Em seguida, o algoritmo foi executado com os parâmetros

Tabela 8 – Parâmetros experimento 3

Fator de regularização do *LASSO*	1e-6
Banda do *kernel RBF*	1.2
Banda do *kernel* usado no *KDE*	0.6

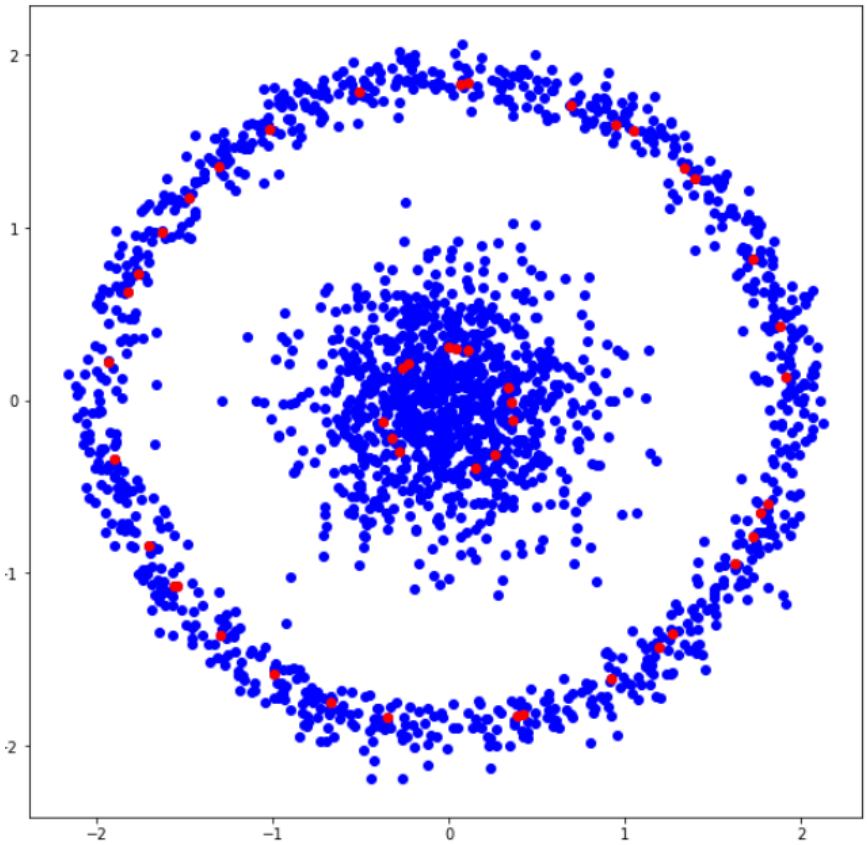

Figura 25 – Experimento 3, pontos selecionados

O algoritmo selecionou 48 pontos dos 2 mil pontos iniciais. Nesse último experimento também foi avaliado a influência da variação da banda de kernel mantendo o valor da regularização do *LASSO* fixa. Já a imagem a seguir, projeta os pontos selecionados sobre a densidade estimada pelo *KDE*.

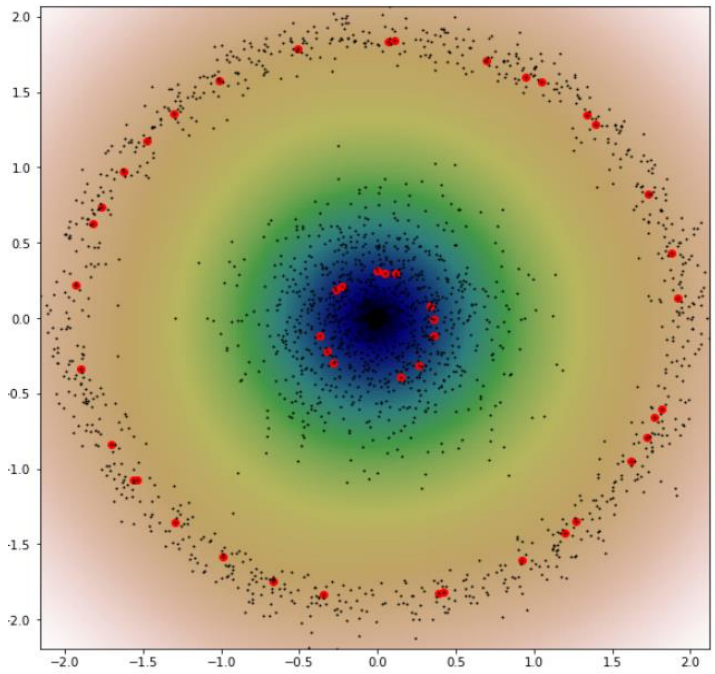

Figura 26 – Experimento 3, pontos selecionados e densidade

Também foi verificada, nesse experimento, a influência da largura de banda quando o fator de regularização do *LASSO* é mantido de forma fixa. A próxima figura demonstra o resultado do *LASSO* com a metade, 0.3, da largura da banda usado para KDE. Como esperado, o número de pontos diminuiu, caiu para 35.

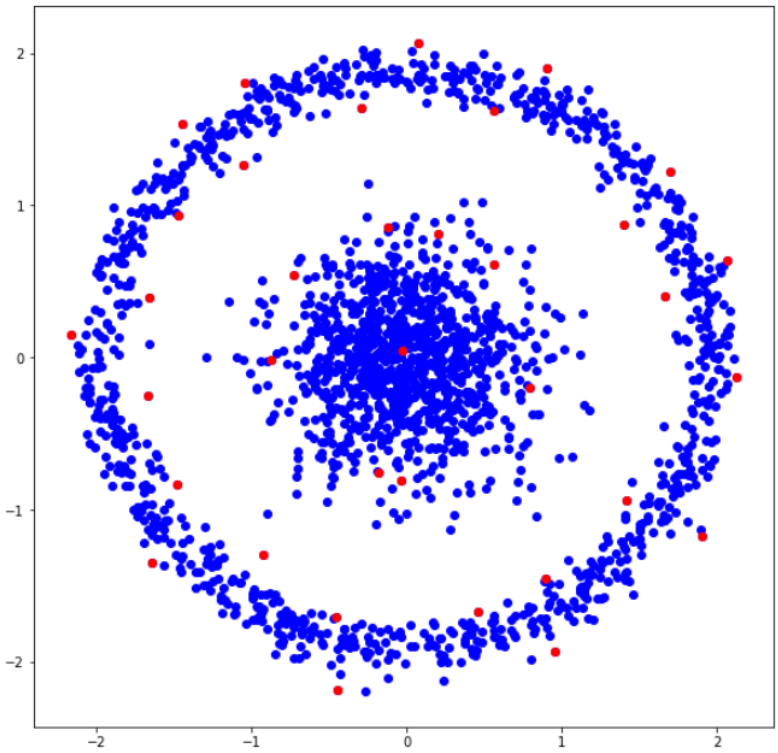

Figura 27 – Experimento 3, pontos selecionados com banda curta

Quando o valor da banda foi reduzido pela metade, os pontos selecionados mudaram. Agora, parece que o algoritmo está priorizando pontos que estão nas bordas.

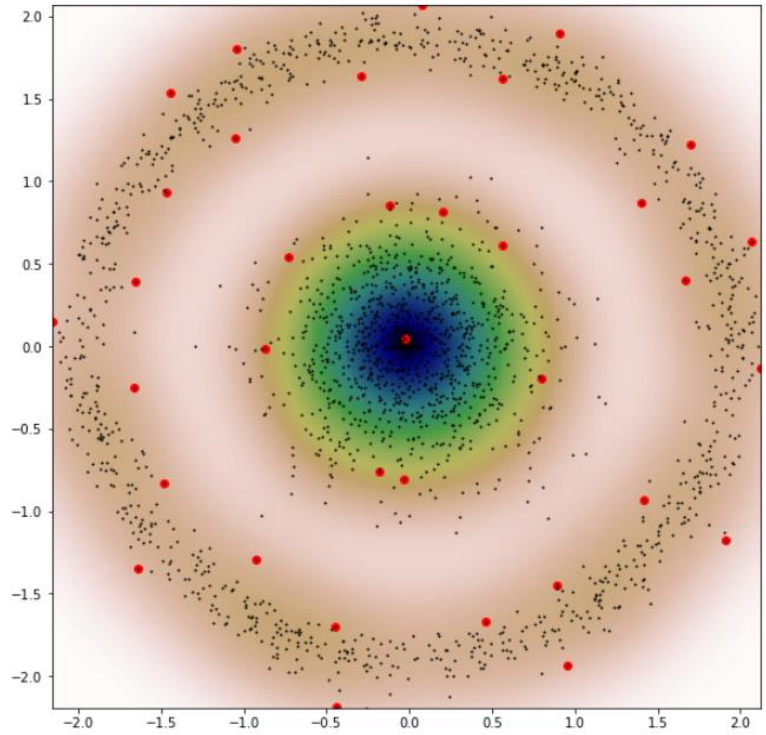

Figura 28 – Experimento 3, pontos selecionados e densidade com banda curta

Para tentar esclarecer esse comportamento, foi gerada mais uma visualização, dessa vez, com as curvas de nível e os pontos selecionados.

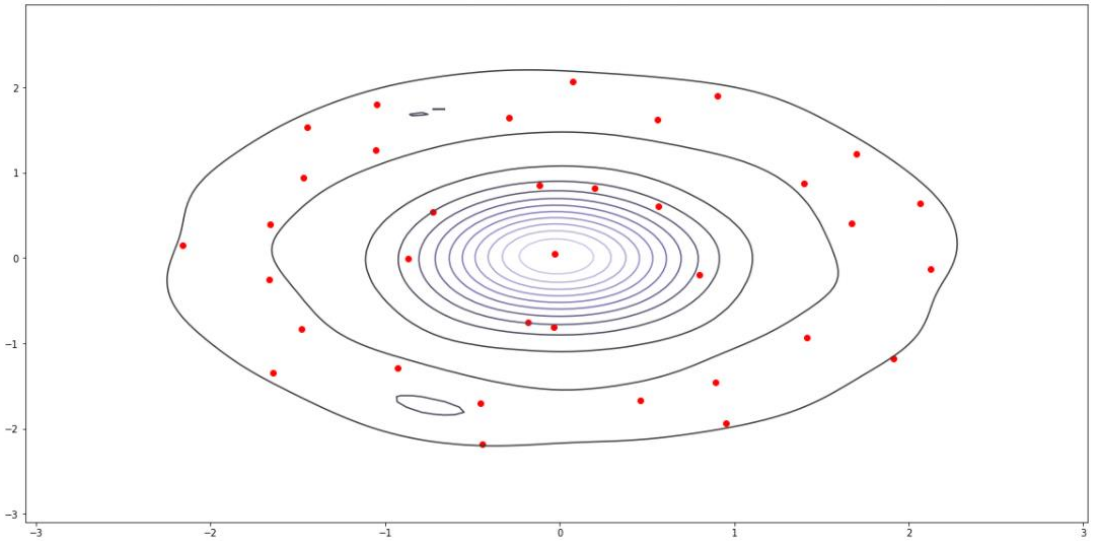

Figura 29 – Experimento 3, curvas de nível com banda excessivamente curta

Também foi feita uma imagem com os valores iniciais de banda. A diferença nos resultados ocorre devido ao valor do *bin* do *KDE* ser muito pequeno, e isso resulta em uma grande variância.

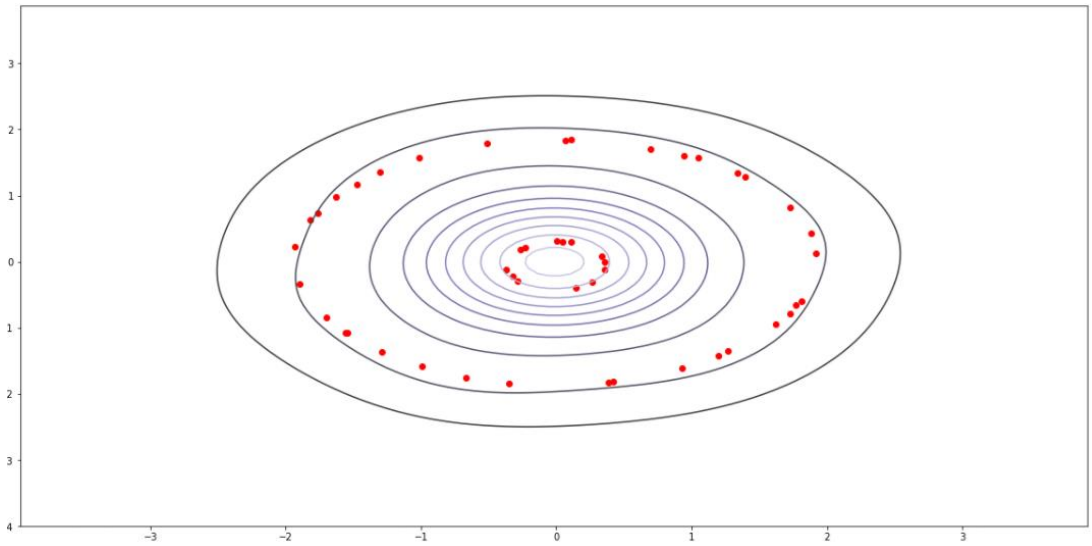

Figura 30 – Experimento 3, curvas de nível com tamanho de banda adequado

Já os resultados com uma banda maior, 1.2, demonstram um desfecho oposto. O algoritmo seleciona mais pontos, 82, e suavizando excessivamente a distribuição.

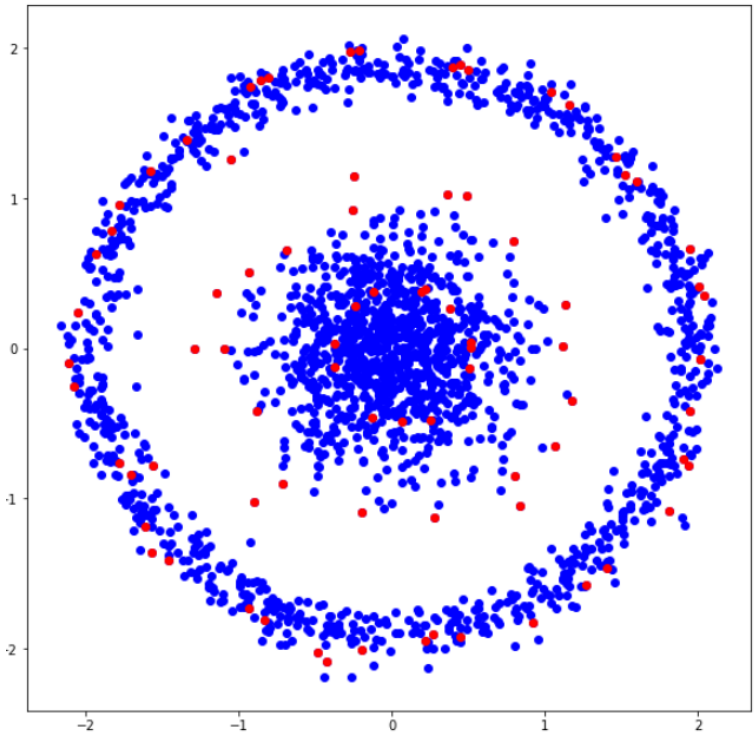

Figura 31 – Experimento 3, pontos selecionados com banda larga

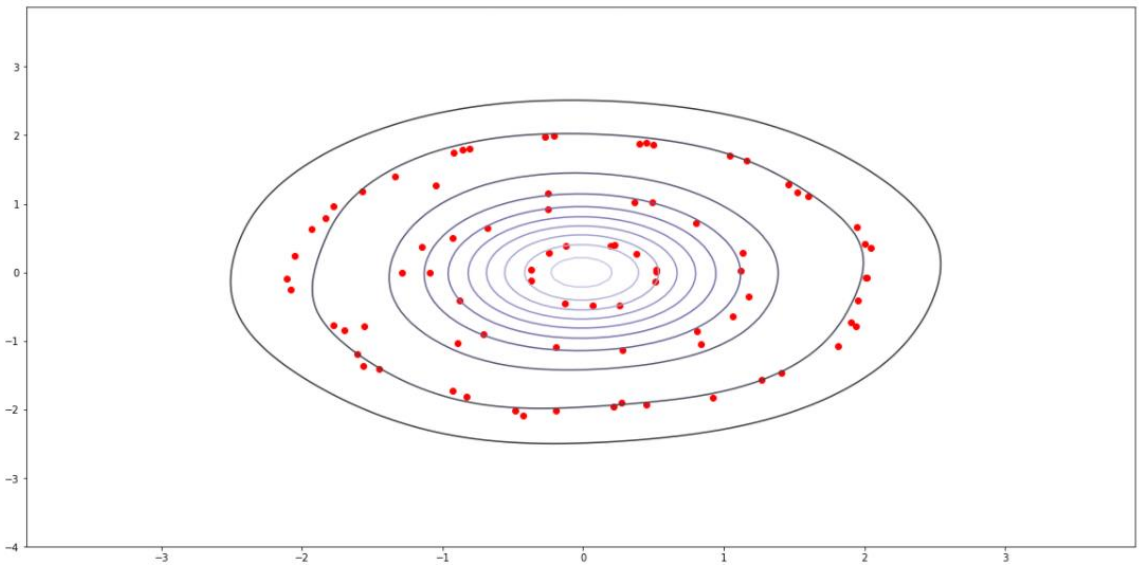

Figura 32 – Experimento 3, curvas de nível com banda excessivamente larga

4.4 Discussão final

Conforme apresentado nos experimentos anteriores, o algoritmo proposto nesse trabalho consegue realizar, de forma consistente, a seleção de instâncias. Também foi demonstrado o impacto exercido pelo tamanho da banda na seleção de instâncias.

Entretanto, não foi viável realizar um experimento envolvendo segmentação de imagem, visto que a matriz de kernel que precisava ser alocada superava a quantidade de memória principal disponível. O estouro de memória sempre ocorreu quando foram usadas imagens de resoluções superiores ou iguais a 320x200.Só foi possível alocar a matriz de kernel quando foi utilizada uma imagem com uma resolução de 120x120, no entanto, com essa resolução a matriz de kernel se degenerou em uma matriz identidade, ou seja, o kernel gaussiano atribuiu semelhança máxima aos pares de pontos da diagonal principal da matriz de kernel e, nas outras posição da matriz, nenhuma semelhança.

É importante ressaltar que mesmo após um levantamento na literatura não foi encontrado um trabalho que fizesse uso do *LASSO* aplicado no problema de *instance Selection*.

5 Conclusão e futuros trabalhos

A elaboração desta monografia teve como objetivo o desenvolvimento de uma proposta de aplicação do *LASSO* no problema de seleção de instâncias. Esta proposta faz uso de diversos métodos, técnicas e conceitos da área de *machine learning*. Portanto, foi necessária uma revisão sistemática da literatura de machine learning, que envolveu modelos lineares, clustering e modelos para *density estimation*. Os experimentos apresentaram evidências que a abordagem proposta é capaz de realizar IS.

Contudo, também foi verificada uma limitação, a matriz de kernel pode crescer de forma que torne inviável a execução do algoritmo.

Trabalhos futuros podem investigar se as diferentes versões do LASSO, como por exemplo, *adaptive LASSO* e *BOLASSO*, são mais adequadas para o problema de IS. Outro aspecto que também pode ser pesquisado é a aplicação do *Akaike information criterion* e *Bayes Information criterion* para a seleção automática de modelos do *LASSO*.

Referências Bibliográficas

ALDENDERFER, Mark S; BLASHFIELD, Roger K. *Cluster analysis*. Newbury: SAGE Publications, 1984.

ALPERT, Jesse; HAJAJ, Nissan. *We knew the web was big...* Disponível em: <https://googleblog.blogspot.com.br/2008/07/we-knew-web-was-big.html> Acesso em: 2 dez. 2017.

BISHOP, Christopher. *Machine Learning and Pattern Recognition.* v.1. Berlin: Springer, 2006.

DOMINGOS, Pedro. *A Few Useful Things to Know about Machine Learning*. Disponível em: < https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf> Acesso em: 27 dez. 2017.

DUDA, Richard O.; HART, Peter E.; STORK, David G.. *Pattern Classification and Scene Analysis*. Disponível em: < http://ww.svms.org/classification/DuHS95.pdf> Acesso em: 29 dez. 2017.

FRIEDMAN, Jerome; HASTIE, Trevor; TIBSHIRANI, Robert. *The Elements of Statistical Learning*. v.7. Berlin: Springer, 2008.

GIROLAMI, Mark; HE, Chao. *Probability Density Estimation from Optimally Condensed Data Samples*. Disponível em: <http://ieeexplore.ieee.org/document/1233899/?reload=true> Acesso em: 5 nov. 2017.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. v. 1. Massachussets: M.I.T. Press, 2016.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. *Data mining* - concepts and techniques. 3ed. Waltham: Morgan Kaufmann, 2011.

HASTIE, Trevor; TIBSHIRANI, Robert; WAINWRIGHT, Martin. Prefácio. In: *Statistical Learning with Sparsity*. Disponível em: <https://web.stanford.edu/~hastie/StatLearnSparsity_files/SLS_corrected_1.4.16.pdf> Acesso em: 10 nov. 2017.

JAMES, Gareth; WITTEN, Daniela; FRIEDMAN, Jerome; HASTIE, Trevor; TIBSHIRANI, Robert. *An Introduction to Statistical Learning*. v.1. Berlin: Springer, 2013.

KEVIN, Murphy P. *Machine Learning*: A Probabilistic Perspective. V.1 London: The Mit Press, 2012.

KOTSIANTIS, S.B. *Supervised Machine Learning: A Review of Classification* Techniques, 2007. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9683&rep=rep1&type=pdf> Acesso em: 29 nov. 2017.

MAGDON-ISMAIL: Malik, TIEN; Hsuan L. *Learning From data – A short Course*. v.1. Pasadena, AMLBook , 2012.

Moo K. Chung. *Notas de Aula da Disciplina STAT 692 - Medical Image Analysis*. Wisconsin-Madison University, 01/10/2007. Disponível em: <http://www.stat.wisc.edu/~mchung/teaching/MIA/reading/diffusion.gaussian.kernel.pdf.pdf> Acesso em: 25 jan. 2018.

NG, Andrew. *Notas de Aula da Disciplina CS 229-Machine Learning*. California: Stanford University, 01/10/2016.

RASCHKA, Sebastian. *Python Machine Learning*. v.1. Birmingham: Packt Publishing, 2016.

SHALIZI, Comas Rohilla. *Advanced Data Analysis from an Elementary Point of View*. Disponível em: <http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf> Acesso em: 10 nov. 2017.

WU, T. Tong; LANGE, Kenneth. *Coordinate Descent algorithms for Lasso Penalized Regression*. Disponível em: <https://arxiv.org/pdf/0803.3876.pdf> Acesso em: 25 nov, 2017.