UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

ALISSON SAMPAIO DE CARVALHO ALENCAR

MÁQUINA DE APRENDIZAGEM MÍNIMA: ASPECTOS TEÓRICOS E PRÁTICOS

FORTALEZA

2017

ALISSON SAMPAIO DE CARVALHO ALENCAR

MÁQUINA DE APRENDIZAGEM MÍNIMA: ASPECTOS TEÓRICOS E PRÁTICOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Orientador: Prof. Dr. João Fernando Lima Alcântara

Co-Orientador: Prof. Dr. João Paulo Pordeus Gomes

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A353m Alencar, Alisson Sampaio de Carvalho.
Máquina de Aprendizagem Mínima: aspectos teóricos e práticos / Alisson Sampaio de Carvalho Alencar. – 2017.
71 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2017.
Orientação: Prof. Dr. João Fernando Lima Alcântara.
Coorientação: Prof. Dr. João Paulo Pordeus Gomes.

1. Aprendizagem de máquina. 2. Máquina de aprendizagem mínima. 3. Capacidade de interpolação. 4. Multilateração. I. Título.

CDD 005

ALISSON SAMPAIO DE CARVALHO ALENCAR

MÁQUINA DE APRENDIZAGEM MÍNIMA: ASPECTOS TEÓRICOS E PRÁTICOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Inteligência Artificial

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. João Fernando Lima Alcântara (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Paulo Pordeus Gomes (Co-Orientador)

Prof. Dr. Amauri Holanda de Souza Júnior
Instituto Federal de Educação Ciência e Tecnologia do Ceará (IFCE)

Prof. Dra. Ananda Lima Freire
Instituto Federal de Educação Ciência e Tecnologia do Ceará (IFCE)

Prof. Dr. Francesco Corona
Universidade Federal do Ceará (UFC)

Aos meus avós Elso e Osvaldo, in memoriam.

AGRADECIMENTOS

Agradeço inicialmente a Deus, fonte de toda sabedoria, que me abençoou com a oportunidade de fazer este trabalho.

À minha família, pelo carinho nos momentos que compartilhamos e compreensão nas inúmeras ausências. Em especial ao meu filho, Tito Júlio, minha esposa, Juliana, meus pais, Edna e Givaldo, minhas queridas irmãs, Aline e Alane e minha avó, Lourdinha. Sem vocês essa jornada teria sido inviável.

Agradeço aos professores João Fernando, João Paulo Pordeus e Amauri Holanda pela valiosa orientação que me permitiu evoluir além do que eu mesmo acreditava.

Aos colegas do laboratório LIA, pelas valiosas discussões.

A todos os professores e demais funcionários do departamento de computação, por suas importantes contribuições.

À CAPES, pelo apoio financeiro.

“Tudo aquilo que se compartilha se multiplica!”

(Papa Francisco)

RESUMO

A máquina de aprendizagem mínima (*Minimal Learning Machine*, MLM) é um método de aprendizado supervisionado cujo treinamento se dá a partir da construção de um mapeamento linear entre matrizes de distâncias obtidas nos espaços de entrada e saída dos dados. Após o treinamento, a saída correspondente a um novo padrão é encontrada pela solução de um problema de multilateração. Dada a sua formulação simples, treinamento rápido e boa acurácia, a MLM tem sido utilizada com sucesso em várias aplicações. Entretanto, deve-se destacar que tais modelos ainda carecem de uma análise detalhada de algumas das suas etapas. Este trabalho apresenta três contribuições referentes a análise MLMs. Inicialmente será apresentada a prova da capacidade de interpolação de um modelo MLM. Posteriormente serão apresentadas análises experimentais das etapas de seleção de pontos de referência e estimação da saída.

Palavras-chave: Aprendizado de máquina. Máquina de aprendizagem mínima. Capacidade de interpolação. Multilateração.

ABSTRACT

Minimal learning machine (MLM) is a supervised learning algorithm based on a linear mapping between input and output distance matrices. After the learning step, the corresponding output prediction for new incoming inputs is achieved by solving a multilateration problem. Given its simple formulation, quick training and good accuracy, MLM has been used successfully in several applications. However, it should be noted that such models still need a detailed analysis of some of their steps. This paper presents three contributions regarding the MLM analysis. Initially will be presented the proof of the interpolation capacity of an MLM model. Afterwards, experimental analyzes of the steps of selection of reference points and estimation of the output will be presented.

Keywords: Machine Learning. Minimal Learning Machine. Interpolation Capacity. Multilateration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

C-MLM	*Cubic Equation Minimal Learning Machine*
ELM	*Extreme Learning Machine*
MLM	*Minimal Learning Machine*
MLP	*Multilayer Perceptron*
NN-MLM	*Nearest Neighborhood Minimal Learning Machine*
RBF-NN	*Radial Bases Function Neural Network*

LISTA DE SÍMBOLOS

$\mathscr{X}$	Espaço de entrada do conjunto de dados
$\mathscr{Y}$	Espaço de saída do conjunto de dados
$\mathscr{D}$	Conjunto de dados formado por pares entrada x saída
δ_k	Função que computa a distância euclidiana entre um ponto do espaço de entrada e o k-ésimo ponto de referência
d_k	Função que computa a distância euclidiana entre um ponto do espaço de entrada e o k-ésimo ponto de referência (no espaço da entrada)
δ_k	Função que computa a distância euclidiana entre um ponto do espaço de saída e o k-ésimo ponto de referência (no espaço da saída)
g_k	Mapeamento que leva as distâncias de um dado ponto no espaço da entrada na distância entre sua saída e o k-ésimo ponto de referência no espaço da saída
$\mathbf{D}_x$	Matriz de distâncias entre os pontos do conjuntos de dados e os pontos de referência no espaço da entrada
$\mathbf{\Delta}_y$	Matriz de distâncias entre os pontos do conjuntos de dados e os pontos de referência no espaço da saída
$\mathbf{B}$	Matriz formada pelos coeficientes de regressão de cada ponto de referência

SUMÁRIO

1 INTRODUÇÃO

O aprendizado de máquina é um ramo da inteligência artificial que se dedica a construir programas de computador que possam "aprender"a se comportar de maneira desejada a partir de dados obtidos de experiências anteriores. Por não possuir dados suficientes para determiná-los com exatidão ou por possuírem algum componente aleatório, essas experiências podem fornecer apenas uma informação parcial sobre o problema. As recorrentes situações em que nos deparamos com esses problemas podem nos permitir entender, razoavelmente, seu funcionamento e assim prever o resultado que provavelmente teríamos em futuras experiências. Destaco o "provavelmente"para enfatizar que, por razões já mencionadas, é possível que nunca tenhamos a condição de encontrar a solução com absoluta certeza.

Na verdade, essa impossibilidade é que torna úteis os algoritmos de aprendizado de máquina. Como vemos em Abu-Mostafa *et al.* (2012), o aprendizado de máquina é usado em situações onde não temos uma solução exata, mas temos dados que nos permitam construir uma solução empírica. Isso nos traz dois conceitos importantes da área: 1) o aprendizado de máquina não é, normalmente, aplicado para "reinventar a roda"construindo soluções razoáveis para problemas que possuem uma solução exata viável. Ela é aplicada para oferecer uma solução razoável a problemas para os quais a obtenção de uma solução melhor seria muito cara ou até impossível. 2) Dados obtidos de experiências anteriores são a base para o funcionamento dos algoritmos de aprendizado de máquina. É dos dados que se busca obter informações relevantes para a solução do problema.

Fixados esses conceitos fundamentais, temos uma infinidade de problemas que se busca resolver. Estes problemas se dividem em dois principais paradigmas: aprendizado supervisionado, onde os dados foram previamente rotulados, e aprendizado não supervisionado, onde os dados apresentam apenas os atributos de entrada. Neste trabalho vamos tratar de aprendizado supervisionado. Mais informações sobre aprendizado supervisionado, não supervisionado e outros paradigmas podem ser encontradas em Haykin (1998), Bishop (2006) e Braga (2007).

Além da variação existente nos conjuntos de dados dos diferentes problemas, há ainda diferenças de objetivos, ou tarefas que desejamos realizar. As principais delas são: aproximação de funções (ou regressão) e reconhecimento de padrões (também conhecido como classificação). Ambas serão objeto de estudo deste trabalho.

Existem diversos métodos que se propõem a solucionar o problema de reconhecimento de padrões e de aproximação de funções, como Perceptron de Múltiplas Camadas

(*Multilayer Perceptron* (MLP)) Haykin (1998), Máquina de Aprendizado Extremo (*Extreme Learning Machine* (ELM)) Huang *et al.* (2012), Rede Neural de Funções de Base Radial (*Radial Bases Function Neural Network* (RBF-NN)) Haykin (1998) e mais recentemente a Máquina de Aprendizagem Mínima (*Minimal Learning Machine* (MLM)) de Souza Junior *et al.* (2013b).

A máquina de aprendizagem mínima é um método de aprendizado supervisionado proposto em de Souza Junior *et al.* (2013b), cujo treinamento se dá a partir da construção de um mapeamento linear entre matrizes de distâncias obtidas nos espaços de entrada e saída dos dados. As matrizes de distâncias são construídas computando a distância euclidiana entre todos os pontos da base de dados e um subconjunto dos mesmos, que chamaremos de pontos de referência. A seleção dos pontos de referência pode ser feita de forma aleatória. Após esse processo, constrói-se um modelo linear de regressão entre as distâncias na entrada para as distâncias na saída. Após o treinamento, a saída correspondente a um novo padrão é encontrada computando-se as distâncias para os pontos de referência no espaço da entrada e aplicando o modelo linear encontrado no treinamento para chegar a uma estimativa das distâncias no espaço da saída. A posição do ponto no espaço da saída é encontrada pela solução de um problema de multilateração.

1.1 Objetivos

Alguns trabalhos propostos na literatura mostram que a MLM apresenta bom desempenho em muitas aplicações, como em de Souza Junior *et al.* (2013b), de Souza Junior *et al.* (2013a), Oliveira *et al.* (2016), Alencar *et al.* (2015), Caldas *et al.* (2016), Gomes *et al.* (2015) e Gomes *et al.* (2017). Apesar disso, alguns aspectos do método carecem de uma análise mais detalhada. Nesta dissertação iremos abordar os seguintes aspectos:

1. estudar a capacidade de interpolação da MLM. Visto que a capacidade de aproximação universal de funções pode ser encontrada através do estudo da capacidade de interpolação e da dimensão VC dos modelos, este estudo se mostra relevante, uma vez que a aproximação universal é uma das principais garantias teóricas que se busca em classificadores.
2. estudar métodos alternativos de métodos de seleção de pontos de referência e seu impacto no desempenho da rede.
3. estudar a capacidade dos métodos de estimação da saída (dada pela multilateração) de recuperar a informação desejada além de um estudo experimental que busca avaliar sua performance quando submetida a dados ruidosos.

4. apresentar uma *toolbox* para Matlab da MLM que foi desenvolvida durante a pesquisa e reúne as principais extensões da MLM propostas na literatura.

A seguir iremos detalhar o objetivo geral desta pesquisa, bem como seus objetivos específicos.

1.1.1 Objetivos Gerais

Neste trabalho temos como objetivo principal o estudo do método Máquina de Aprendizagem Mínima , proposto em de Souza Junior *et al.* (2013b). Aqui abordaremos uma garantia teórica desejável, dada pela capacidade de interpolação do métodos e também questões de ordem prática, como a influencia dos métodos de seleção de pontos de referência no desempenho do método e a capacidade de estimar a saída via multilateração. Além disso, traremos também uma *toolbox* para Matlab desenvolvida durante as pesquisas.

1.1.2 Objetivos Específicos

De forma mais detalhada, podemos listar os objetivos dessa dissertação como:

1. Avaliar a capacidade da rede MLM de interpolar um conjunto de pontos arbitrário, o que inclui:
    a) Avaliar a inversibilidade da matriz de distâncias dos dados no espaço da entrada.
    b) Avaliar a capacidade da correta estimação das saídas.
2. Avaliar a influencia do método de seleção de pontos de referência no desempenho da rede
3. Avaliar a capacidade de estimação da saída dos métodos aqui abordados e sua tolerância a presença de ruído
4. Implementar uma *toolbox* em matlab para a rede MLM e algumas variações.

1.2 Estrutura da Dissertação

O restante dessa seção está organizado da seguinte forma:

No capítulo 2 apresentamos a rede mlm.

No capítulo 3 avaliamos alguns aspectos teóricos da rede para mostrar que ela possui a capacidade de aproximar funções arbitrariamente complexas.

O capítulo 4 traz um estudo experimental sobre a influencia do método de seleção de pontos de referência na performance da MLM e sobre a capacidade de estimação da saída pelos

métodos abordados.

A *toolbox* pra matlab é apresentada no capítulo 5.

O capítulo 6 apresenta a conclusão do trabalho.

2 MÁQUINA DE APRENDIZAGEM MÍNIMA

A máquina de aprendizagem mínima é um método de regressão proposto em de Souza Junior *et al.* (2013b) que busca relacionar as distâncias entre os pontos no espaço da entrada com as distâncias entre eles no espaço da saída.

Para isso, seleciona-se K pontos do conjunto de dados para serem os nossos **pontos de referência**. A priori, consideremos que essa seleção é aleatória. E construímos um mapeamento linear entre as distâncias aos pontos de referência na entrada e as distâncias aos pontos de referência na saída. A predição de novas amostras se dá usando as distâncias no espaço da entrada e o modelo computado para gerar uma estimativa das distâncias aos pontos de referência no espaço da saída, e aplica-se, então, algum método que permita recuperar uma estimativa para a saída a partir das distâncias computadas.

2.1 Formulação básica

Neste trabalho vamos considerar um problema de aprendizagem como a tarefa de aproximar uma função alvo, contínua, $f : \mathscr{X} \to \mathscr{Y}$ a partir de um conjunto de dados $\mathscr{D} = \{(\mathbf{x}_n, \mathbf{y}_n = f(\mathbf{x}_n)) + \varepsilon_n\}_{n=1}^{N}$, com $\mathbf{x}_n \in \mathscr{X}$ e $\mathbf{y}_n \in \mathscr{Y}$. Neste trabalho chamaremos $\mathscr{X} = \mathbb{R}^D$ de espaço da entrada e $\mathscr{Y} = \mathbb{R}^S$ de espaço da saída.

Para aproximar uma função objetivo f, a MLM usa transformações $\delta_k : \mathscr{Y} \to \mathbb{R}_+$ e $d_k : \mathscr{X} \to \mathbb{R}_+$ dadas pelas funções de distâncias computadas para pontos fixos $\{(\mathbf{m}_k, \mathbf{t}_k = f(m_k)) \in \mathscr{D}\}_{k=1}^{K}$ que chamamos de **pontos de referência**. Formalmente, temos

$$\mathrm{d}_k(\mathbf{x}) = d(\mathbf{x}, \mathbf{m}_k)$$

$$\delta_k(\mathbf{y}) = \delta(\mathbf{y}, \mathbf{t}_k)$$

onde $d(\cdot,\cdot)$ e $\delta(\cdot,\cdot)$ são dadas pela distância euclidiana.

Considerando que exista um mapeamento entre os espaços induzidos pelas funções δ e d, isto é, existe $g_k : \mathbb{R}_+^K \to \mathbb{R}_+$, nós desejamos computar as distâncias de cada ponto em $\mathscr{D}$ para cada ponto de referência do espaço da entrada, construindo a matriz $\mathbf{D} \in \mathbb{R}_+^{N \times K}$. Analogamente, podemos computar as distâncias no espaço da saída entre todos os pontos do conjunto de dados e as saídas dos pontos de referência, gerando assim a matriz $\mathbf{\Delta} \in \mathbb{R}_+^{N \times K}$. Com isso, passamos a procurar um mapeamento g_k tal que $\mathbf{\Delta}_{n,k} = g_k(\mathbf{D}_{n,\cdot}) + \varepsilon_n \; \forall \; n = 1, \ldots, N$, onde ε_n é o erro residual e $\mathbf{D}_{n,\cdot}$ é a n-ésima linha da matriz $\mathbf{D}$.

No algoritmo MLM nós assumimos que o mapeamento entre as distâncias na entrada

e as distâncias na saída, dado por g_k, é linear. Com isso, podemos estimar as distâncias no espaço da saída através de uma combinação linear das distâncias no espaço da entrada, ou seja, temos $\mathbf{\Delta}_{n,k} = \mathbf{D}_{n,\cdot}\mathbf{b}_k + \varepsilon_n$, onde $\mathbf{b}_k \in \mathbb{R}^K$ representa os coeficientes do mapeamento linear. Assim, o mapeamento para todos os vetores em $\mathscr{D}$ é dado por $\mathbf{\Delta} = \mathbf{DB} + \varepsilon$, sendo a matriz $\mathbf{B} \in \mathbb{R}^{K\times K}$ composta pelos vetores de coeficientes de cada saída individual.

Após esse processo, a MLM recupera a estimativa de saída de um novo dado desconhecido através da função $h_{\mathbf{B}}(\mathbf{x}) : \mathscr{X} \to \mathscr{Y}$ dada por:

$$h_{\mathbf{B}}(\mathbf{x}) = \arg\min_{\mathbf{y}} \sum_{k=1}^{K} \left[\delta_k^2(\mathbf{y}) - \left(\sum_{i=1}^{K} d_i(\mathbf{x}) B_{i,k} \right)^2 \right]^2, \tag{2.1}$$

onde $\delta_k(\mathbf{y}) = \|\mathbf{y} - \mathbf{t}_k\|$ é a distância euclidiana entre $\mathbf{y}$ e a saída do k-ésimo ponto de referência $\mathbf{t}_k$, $d_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_i\|$ é a distância euclidiana entre $\mathbf{x}$ e o i-ésimo ponto de referência $\mathbf{m}_i$ e K é o número de pontos de referência.

2.1.1 Treinamento

O treinamento da rede MLM se divide em duas etapas:

1. seleção dos pontos de referência $\{(\mathbf{m}_k; \mathbf{t}_k)\}$;
2. encontrar a matriz de regressão $\mathbf{B}$.

Em relação à seleção dos pontos de referência, a proposta original sugere que seja feita aleatoriamente. Neste trabalho, estudamos algumas alternativas no Capítulo 4. Outras técnicas de seleção de pontos de referência de redes MLM podem ser encontradas em Oliveira e Souza (2017).

A segunda parte do algoritmo, referente a encontrar a matriz de regressores lineares $\mathbf{B}$, pode ser feita através do método dos mínimos quadrados, cuja estimativa é dada por

$$\hat{\mathbf{B}} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\Delta, \tag{2.2}$$

Onde $\mathbf{D}$ e $\mathbf{\Delta}$ são as matrizes de distância entre os pontos do conjunto de dados e os pontos de referência no espaço de entrada e de saída, respectivamente.

2.1.2 Teste

Assim como o treinamento, usar uma rede MLM treinada para estimar a saída de uma nova amostra $\mathbf{x}$, que conhecemos apenas sua representação no espaço da entrada, também é feita em

duas etapas

1. estimativa das distâncias para os pontos de referência no espaço da saída;
2. recuperação da saída $\hat{\mathbf{y}}$ a partir das estimativas das distâncias.

Para isso, usamos o novo ponto $\mathbf{x}$ e computamos a distância dele para cada um dos pontos de referência $\{\mathbf{m}_k\}$ formando o vetor $[d_1(\mathbf{x}),\ldots,d_K(\mathbf{x})]$. Com isso, estimamos as distâncias entre a nova amostra e os pontos de referência no espaço da saída através da equação

$$\hat{\delta}_k(\mathbf{y}) = \sum_{i=1}^{K} d_i(\mathbf{x})\hat{B}_{i,k}, \qquad \forall k = 1,\ldots,K. \tag{2.3}$$

Estas estimativas são então usadas para encontrar a posição da saída desejada (ou uma estimativa desta). Existem na literatura vários métodos que se propõe a resolver essa tarefa. Neste trabalho veremos um deles no capítulo 3. A proposta original usa o algoritmo de Levenberg-Marquardt para encontrar uma estimativa de saída $\hat{\mathbf{y}}$ com a propriedade a seguir

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \sum_{k=1}^{K} \Big((\mathbf{y}-\mathbf{t}_k)^T(\mathbf{y}-\mathbf{t}_k) - \hat{\delta}_k^2(\mathbf{y}) \Big)^2. \tag{2.4}$$

2.2 Extensões da MLM

Algumas pesquisas, como em Garcia-Pedrajas *et al.* (2005), têm mostrado que um comitê de classificadores apresenta resultados melhores que o uso de um único classificador. Um comitê de classificadores consiste em treinar vários classificadores e em seguida combinar os resultados para gerar uma única saída.

Uma tentativa inicial de criar um comitê de MLMs esbarraria no problema de estimar a saída usando métodos de otimização. Este processo, que já é problemático em um classificador, seria ampliado no uso de vários classificadores. Assim, vemos em Mesquita *et al.* (2017) uma proposta para tornar mais rápida a estimação da saída da rede MLM, bem como uma proposta de treinamento e uso de comitês de MLM.

A seguir veremos a versão rápida do MLM para classificação e uma versão para regressão aplicável quando a saída for unidimensional. Na seção seguinte veremos todas as propostas de treinamento e combinação dos resultados em um comitê de MLMs.

2.2.1 fast MLM

Aqui veremos propostas para acelerar o uso do MLM encontradas em Mesquita *et al.* (2017). As propostas se aplicam a classificação e a regressão, quando a saída tem apenas uma dimensão (isto é, $y = f(x) \in \mathbb{R}$).

2.2.1.1 NN-MLM

Aqui iremos apresentar o método *Nearest Neighborhood Minimal Learning Machine* (NN-MLM), que explora características inerentes ao problema de classificação para acelerar o processo de estimação de saída da rede MLM.

Uma característica particular dos problemas de classificação é que a saída pertence a um conjunto finito de opções, que são as classes. Assim, na estimação da saída não seria necessário explorar todas as saídas possíveis para determinar aquela que minimizaria o erro entre a distância e as estimativas obtidas. Aqui, estaremos interessados apenas em saber qual das classes possui a menor diferença entre as distâncias estimadas e a distância que teríamos em cada caso.

Para ilustrar melhor essa abordagem, considere o conjunto de dados de exemplo da Figura 1. Na abordagem tradicional do MLM escolhemos aleatoriamente os pontos de referência no espaço da entrada e da saída, o que podemos ver na Figura 2.

Após o treinamento convencional da MLM, aplicá-la a um novo ponto desconhecido nos dará estimativas para a distância entre a saída desse novo ponto e as saídas dos pontos de referência, como podemos ver na Figura 3. Do lado esquerdo da imagem, vemos um ponto desconhecido, marcado em amarelo, e suas distâncias aos pontos de referência no espaço da entrada. Do lado direito temos os pontos de referência no espaço da saída e os círculos que representam as distâncias estimadas pelo MLM. Aqui vale destacar alguns fatos importantes:

1. temos vários pontos de referência no espaço da saída, mas eles estão sobrepostos sobre os pontos que representam as duas únicas classes presentes no problema;
2. as estimativas de distâncias para pontos da mesma classe são idênticas, como consequência o gráfico só nos permite ver dois círculos;
3. o círculo da classe azul é bem menor que o círculo da classe vermelha, o que já era esperado, já que o ponto amarelo encontra-se "dentro"da nuvem de pontos vermelhos.

Assim, a proposta feita foi não usar otimização, mas sim o algoritmo 1NN no espaço

Figura 1 – Base de dados artificial

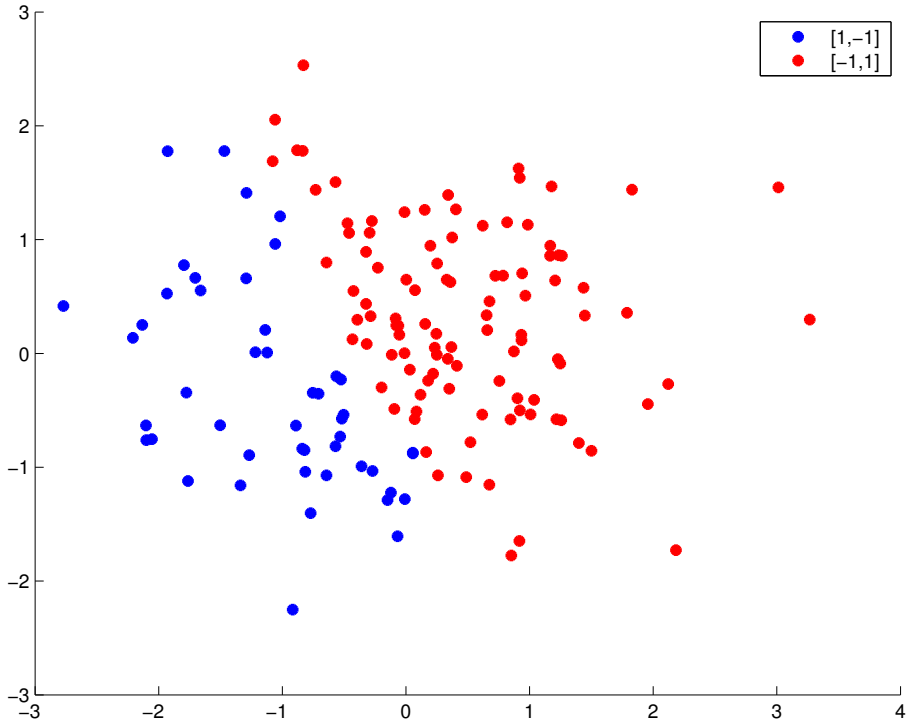
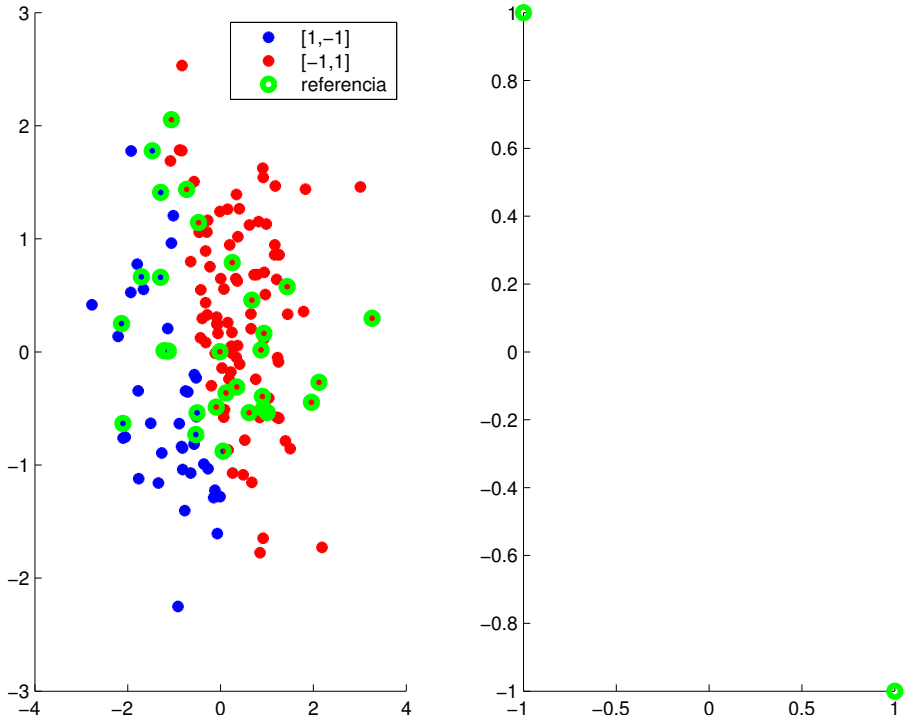
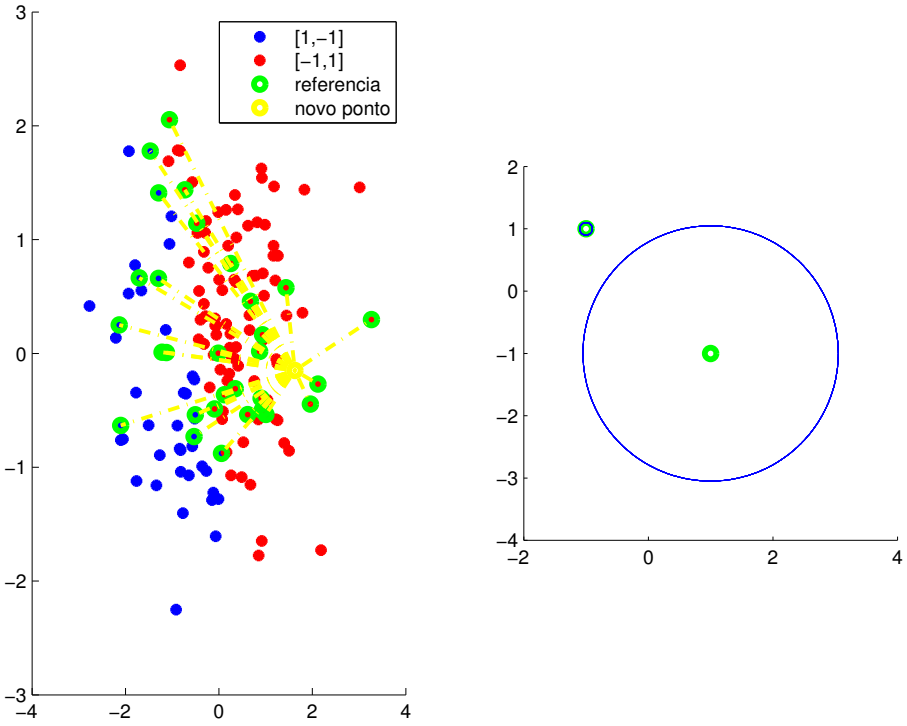


Figura 2 – Seleção aleatória dos pontos de referência no espaço da entrada (à esquerda) e da saída (à direita)



da saída. Ou seja, atribuir ao novo ponto a classe do ponto cuja estimativa da distância na saída for a menor possível. Em Mesquita *et al.* (2017), o resultado que ilustramos empiricamente é

Figura 3 – Estimação da saída da MLM



comprovado de forma analítica.

2.2.1.2 C-MLM

O método *Cubic Equation Minimal Learning Machine* (C-MLM), que descreveremos a seguir, apresenta uma solução mais rápida para problemas de regressão com saída unidimensional. Neste caso, a função custo da estimação da saída pode ser reescrita conforme a equação 2.5. Como temos $y \in \mathbb{R}$, temos que J é uma função de $\mathbb{R}$ em $\mathbb{R}$ dada por um polinômio de ordem 4.

$$J(y) = \sum_{k=1}^{K} ((y - t_k)^2 - \hat{\delta}^2(y, t_k))^2 \tag{2.5}$$

Assim, podemos calcular a derivada de $J(y)$ e igualar a zero para encontrar os pontos críticos de forma analítica resolvendo a equação 2.6.

$$\underbrace{K}_{A} y^3 \underbrace{-3 \sum_{k=1}^{K} t_k}_{B} y^2 + \underbrace{\sum_{k=1}^{K} (3t_k^2 - \hat{\delta}^2(y, t_k))}_{C} y + \underbrace{\sum_{k=1}^{K} (\hat{\delta}^2(y, t_k) t_k - t_k^3)}_{D} = 0 \tag{2.6}$$

Como vemos, a equação é dada por um polinômio de ordem 3 do tipo $Ay^3 + By^2 + Cy + D = 0$. Essa equação possui fórmula fechada e pode ser resolvida de várias formas. No

trabalho em questão ela foi resolvida pelo método de Cardano.

A solução da equação 2.6 pode conduzir a três casos, que devem ser tratados conforme descrito na Tabela 2.

Tabela 2 – C-MLM (Proposta)

	situação	**abordagem**
caso 1	três raízes reais, não todas iguais	calcula-se o valor $J(y)$ de cada uma delas e usamos aquela que apresentar o menor resultado
caso 2	três raízes reais e iguais	usamos o valor obtido nas raízes como resultado
caso 3	uma raiz real e duas raízes complexas	usamos a raiz real como resultado

2.2.2 Comitê de MLMs

O uso de comitê de máquinas, de maneira geral, pode ser dividido em duas etapas:

1. **Geração:** criação dos vários modelos que irão compor o comitê, onde se espera um resultado melhor que cada máquina individual.
2. **Integração:** combinação dos resultados dos membros do comitê de modo a gerar uma saída única.

A seguir descreveremos os métodos de geração e integração propostos em (MESQUITA *et al.*, 2017).

2.2.2.1 Geração

Durante o processo de geração dos modelos que irão compor o comitê devemos atentar para dois itens principais: acurácia de cada modelo e diversidade entre os modelos. Caso os modelos tenham uma acurácia ruim, a combinação de suas saídas não deve levar a um resultado muito melhor. Por outro lado, se os modelos forem muito semelhantes entre si, a saída resultante será praticamente a repetição das saídas, quase idênticas, dos modelos, tornando o uso do comitê dispensável.

Para buscar diversidade aos modelos sem comprometer sua acurácia duas propostas foram feitas:

1. construir M MLMs com pontos de referências distintos, selecionados aleatoriamente. Aqui cada modelo precisaria usar menos pontos de referências do que o total de dados disponível para treinamento, caso contrário os modelos seriam idênticos.
2. construir M MLMs, cada uma usando uma fração P dos dados disponíveis para treinamento.

Aqui todos os pontos disponibilizados para cada modelo seriam usados como pontos de referência.

Na primeira técnica a diversidade vem do uso de diferentes pontos de referência, enquanto na segunda vem do uso de diferentes pontos de treinamento.

2.2.2.2 Integração

A tarefa de integração depende do tipo de atividade que se está trabalhando: classificação ou regressão. Para cada atividade duas técnicas de integração foram propostas, como veremos a seguir.

2.2.2.2.1 Classificação

A primeira proposta, e a mais simples, é a do voto majoritário. Aqui simplesmente contamos quantos classificadores do comitê rotularam a nova amostra como pertencente a cada classe e retornamos como saída a classe mais votada.

A segunda estratégia considera um voto ponderado, onde cada classificador tem seu voto multiplicado por um fator que depende da confiança na sua predição. No caso do NN-MLM essa ponderação pode ser baseada na relação entre a menor distância e as somas de todas as distâncias. Formalmente, temos:

$$w_m = \frac{max_{s=1,\ldots,S}\hat{\delta}^{-1}_{(m)}(y,e_s)}{\sum_{s=1}^{S}\hat{\delta}^{-1}_{(m)}(y,e_s)} \tag{2.7}$$

Onde e_s é o rótulo da s-ésima classe do problema.

2.2.2.2.2 Regressão

No caso de tarefas de regressão, a abordagem mais simples seria retornar a média das estimativas de cada modelo do comitê. Ou seja, fazer $\hat{y} = \frac{1}{M}\sum_{m=1}^{M}\hat{y}(m)$, onde $\hat{y}(m)$ é a saída estimada do m-ésimo MLM.

Uma outra abordagem seria combinar as estimativas das distâncias de todos os modelos na função objetivo, como vemos a seguir:

$$J(y) = \sum_{m=1}^{M}\sum_{k=1}^{K(m)}((y-t_m k)^2 - \hat{\delta}_{(m)}(y,t_{(m)k}))^2 \tag{2.8}$$

Onde $K(m)$ é o número de pontos de referência do m-ésimo modelo, $t_{(m)k}$ é a saída do k-ésimo ponto de referência do m-ésimo modelo e $\hat{\delta}_{(m)}$ é a estimativa da distância dada pelo m-ésimo modelo. Note que caso a saída seja unidimensional ainda poderemos usar o C-MLM simplesmente derivando a equação 2.8 e igualando a zero.

2.3 Conclusão

Neste capítulo apresentamos a rede MLM, um método de classificação e regressão proposto em de Souza Junior *et al.* (2013b) que é o principal objeto de estudo deste trabalho. Vimos como se dá o treinamento de uma rede e como é feita a estimação da saída.

Além disso, vimos algumas propostas que visam acelerar a etapa de estimação de saída em casos específicos e uma proposta de comitê de MLMs, encontrada em Mesquita *et al.* (2017).

Notamos que podemos explorar a MLM em relação aos métodos alternativos de seleção de pontos de referência e a capacidade de interpolação de um conjunto de dados, assuntos que serão objeto de estudo dos próximos capítulos.

3 ASPECTOS TEÓRICOS DA MÁQUINA DE APRENDIZAGEM MÍNIMA

Neste capítulo iremos demonstrar uma importante garantia teórica da rede MLM. Ela se trata da capacidade de interpolação. Essa propriedade é um passo importante na direção de verificar que a rede possui a capacidade de aproximação universal de funções.

Mostrar que o MLM consegue interpolar os dados é uma tarefa que realizaremos em duas etapas. Primeiro, iremos mostrar que a matriz de distâncias $\mathbf{D}_x$, construída usando todos os pontos da base de dados como pontos de referência, admite uma inversa. Considerando que seja verdade (e iremos mostrar que é), saberemos que a estimativa das distâncias serão todas corretas. A partir daí, provaremos que, sob certas condições a serem descritas, a estimação da saída irá recuperar a posição exata dos pontos com erro zero.

3.1 Inversibilidade da matriz de distâncias

Durante o treinamento da rede MLM nós precisamos resolver um sistema linear cuja matriz de coeficientes é formada pelas distâncias entre os pontos do conjunto de dados e os pontos de referência, ou seja, uma matriz $\mathbf{D}$ tal que o elemento $d_{i,j}$ é dado por $d(\mathbf{x}_i, \mathbf{r}_j)$, i.e., a distância entre o i-ésimo ponto do conjunto de dados e o j-ésimo ponto de referência. Se considerarmos o caso particular em que todos os pontos do conjunto de dados são pontos de referência, a matriz de coeficientes será então uma matriz quadrada de ordem igual ao número de pontos N. Podemos reordenar os pontos de modo a ter $x_i = r_i \forall i \in [1, N]$, fazendo com que a matriz de coeficientes seja tal que cada elemento $d_{i,j}$ seja dado por $d(\mathbf{x}_i, \mathbf{x}_j)$. Uma matriz com essa característica é chamada de **Matriz de Distâncias**. Para que o sistema possua solução exata precisamos mostrar que toda matriz de distâncias admite inversa, e esse é o objetivo desta seção.

Este resultado foi demonstrado primeiramente em (MICCHELLI, 1986), usando recursos matemáticos avançados e fazendo uso de diversos resultados de álgebra e análise. Uma demonstração alternativa foi proposta em (AUER, 1995), usando apenas alguns conceitos de cálculo e álgebra linear. É esse trabalho que iremos discutir nesta seção.

Iremos provar que, sendo $[\mathbf{D}]_N$ uma matriz de distâncias de ordem N, teremos $det([\mathbf{D}]_N) > 0$ quando N for ímpar e $det([\mathbf{D}]_N) < 0$ para N par. Em particular, concluiremos que $det([\mathbf{D}]_N) \neq 0$, o que garante que $[\mathbf{D}]_N$ admite inversa. Faremos isso usando indução no tamanho da matriz (N).

3.1.1 *Base da indução ($N = 2$ e $N = 3$)*

Verificar o caso base é bastante simples. Sabendo que toda matriz de distâncias é simétrica e formada por números positivos exceto os elementos da diagonal principal, que são todos zeros, fazemos:

$$[\mathbf{D}]_2 = \begin{bmatrix} 0 & d_{12} \\ d_{12} & 0 \end{bmatrix} \qquad [\mathbf{D}]_3 = \begin{bmatrix} 0 & d_{12} & d_{13} \\ d_{12} & 0 & d_{23} \\ d_{13} & d_{23} & 0 \end{bmatrix}$$

e assim vemos que $det([\mathbf{D}]_2) = -d_{12}^2 < 0$ e $det([\mathbf{D}]_3) = 2d_{12}d_{13}d_{23} > 0$.

3.1.2 *Passo indutivo*

A nossa hipótese de indução será de que o resultado desejado é válido para qualquer conjunto de N ou menos pontos em $\mathbb{R}^L$. Teremos que mostrar que isso implica que o resultado também vale para qualquer conjunto de $N+1$ pontos em $\mathbb{R}^L$.

Inicialmente vamos reescrever o determinante de $[\mathbf{D}]_{N+1}$, fixando a matriz de distâncias $[\mathbf{D}]_N$, que relaciona os N primeiros pontos (e, pela hipótese de indução, deve possuir a propriedade desejada) e acrescentando um vetor $\boldsymbol{\alpha} = [d_{1,N+1}, d_{2,N+1}, ..., d_{N,N+1}]$, que contém as distâncias entre o (N+1)-ésimo ponto e os demais. Assim, podemos escrever o determinante de $[\mathbf{D}]_{N+1}$ como uma função de $\boldsymbol{\alpha}$, dada por

$$f(\alpha) = det([\mathbf{D}]_{N+1}) = det \begin{bmatrix} & & & [\mathbf{D}]_N & & & \\ 0 & d_{12} & ... & d_{1j} & ... & d_{1n} & \alpha_1 \\ d_{12} & 0 & ... & d_{2j} & ... & d_{2n} & \alpha_2 \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ d_{i1} & d_{i2} & ... & d_{ij} & ... & d_{in} & \alpha_i \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ d_{n1} & d_{n2} & ... & d_{nj} & ... & 0 & \alpha_n \\ \alpha_1 & \alpha_2 & ... & \alpha_j & ... & \alpha_n & 0 \end{bmatrix} \quad \text{vetor } \alpha \tag{3.1}$$

Vamos calcular o determinante de $[\mathbf{D}]_{N+1}$ em relação a última coluna. Assim, ficamos

com

$$det([\mathbf{D}]_{N+1}) = \sum_{i=1}^{N} \alpha_i * (-1)^{i+N+1} * det([\mathbf{D}]_{N+1}^{i,N+1}) \tag{3.2}$$

Onde $[\mathbf{D}]_{N+1}^{i,N+1}$ é a matriz $[\mathbf{D}]_{N+1}$ após a remoção da i-ésima linha e da última coluna. Note que o somatório pode iterar apenas até n pois o último elemento da última coluna é zero.

Agora vamos calcular o determinante de $[\mathbf{D}]_{N+1}^{i,N+1}$ em relação a última linha. Por simplicidade de notação, aqui chamaremos a matriz $[\mathbf{D}]_{N+1}^{i,N+1}$ de $\mathbf{A}$. Assim, temos:

$$det([\mathbf{D}]_{N+1}^{i,N+1}) = |\mathbf{A}| = \sum_{j=1}^{N} \alpha_j (-1)^{j+N} |\mathbf{A}^{N,j}|$$

onde $\mathbf{A}^{N,j}$ é a matriz $[\mathbf{D}]_{N+1}^{i,N+1}$ após a remoção da última linha e da j-ésima coluna. Note que $\mathbf{A}^{N,j} = [\mathbf{D}]_N^{ij}$, ou seja, a matriz $[\mathbf{D}]_N$ após a remoção da i-ésima linha e da j-ésima coluna. Assim,

$$det([\mathbf{D}]_{N+1}^{i,N+1}) = \sum_{j=1}^{N} \alpha_j (-1)^{j+N} det([\mathbf{D}]_N^{ij}) \tag{3.3}$$

Substituindo a equação 3.3 em 3.2, ficamos com

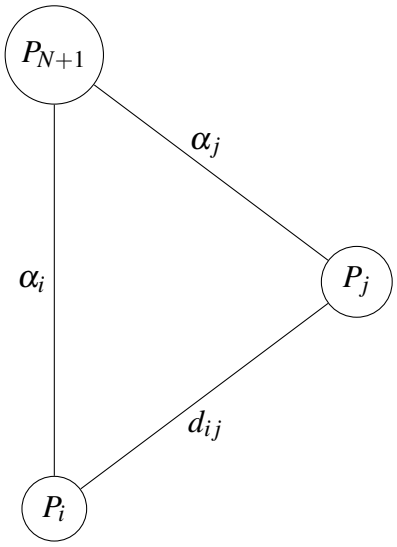
$$\begin{aligned}
det([\mathbf{D}]_{N+1}) &= \textstyle\sum_{i=1}^{N} \alpha_i (-1)^{i+N+1} \sum_{j=1}^{N} \alpha_j (-1)^{j+N} det([\mathbf{D}]_N^{i,j}) \\
&= \textstyle\sum_{i=1}^{N} \alpha_i (-1)^{i+N+1} \sum_{j=1}^{N} \alpha_j (-1)^{j+N-i+i} det([\mathbf{D}]_N^{i,j}) \\
&= \textstyle\sum_{i=1}^{N} \alpha_i (-1)^{i+N+1} \sum_{j=1}^{N} \alpha_j (-1)^{i+j} (-1)^{N-i} det([\mathbf{D}]_N^{i,j}) \\
&= \textstyle\sum_{i=1}^{N} \alpha_i (-1)^{i+N+1} (-1)^{N-i} \sum_{j=1}^{N} \alpha_j (-1)^{i+j} det([\mathbf{D}]_N^{i,j}) \\
&= \textstyle\sum_{i=1}^{N} \alpha_i (-1)^{i+N+1+N-i} \sum_{j=1}^{N} \alpha_j (-1)^{i+j} det([\mathbf{D}]_N^{i,j}) \\
&= \textstyle\sum_{i=1}^{N} \alpha_i (-1)(-1)^{2N} \sum_{j=1}^{N} \alpha_j (-1)^{i+j} det([\mathbf{D}]_N^{i,j}) \\
&= -\textstyle\sum_{i=1}^{N} \alpha_i \sum_{j=1}^{N} \alpha_j (-1)^{i+j} det([\mathbf{D}]_N^{i,j}) \\
&= -\boldsymbol{\alpha}^T adj([\mathbf{D}]_N) \boldsymbol{\alpha}
\end{aligned} \tag{3.4}$$

Onde $adj(.)$ representa a matriz adjunta. Assim, concluímos que $f(\boldsymbol{\alpha}) = -\boldsymbol{\alpha}^T adj([\mathbf{D}]_n)\boldsymbol{\alpha}$. Ou seja, temos uma função quadrática com termos lineares e independentes iguais a zero. Sabemos também que o gradiente de f é dado por $\nabla f(\boldsymbol{\alpha}) = -2adj([\mathbf{D}]_n)\boldsymbol{\alpha}$.

A priori, poderíamos calcular o valor dessa função para qualquer vetor $\boldsymbol{\alpha} \in \mathbb{R}^N$, entretanto, a matriz $[\mathbf{D}]_{n+1}$ só será uma matriz de distância, e portanto, interessante ao nosso

estudo, quando o vetor $\boldsymbol{\alpha}$ for tal que exista $P_{n+1} \in \mathbb{R}^L$ tal que $\alpha_i = d(P_i, P_{N+1})\ \forall i \in [1,N]$, onde P_i são os pontos que deram origem a matriz $[\mathbf{D}]_n$. Chamaremos esse conjunto de $\mathbb{S}' \subset \mathbb{R}^N$. Considere agora o conjunto $\mathbb{S} \subset \mathbb{R}^N$ de todos os pontos $\boldsymbol{\alpha} \in \mathbb{R}^N$ tais que para quaisquer $i \neq j$, as distâncias entre os pontos P_i, P_j e um **possível** ponto P_{N+1} não violariam a desigualdade triangular. A Figura 4 ilustra esse conceito. Note que essa condição não garante que exista P_{N+1} cujas distâncias para os outros n pontos sejam dadas pelo vetor $\boldsymbol{\alpha}$. Note também que, como toda função de distância deve satisfazer a desigualdade triangular, temos $\mathbb{S}' \subset \mathbb{S}$. Assim, provaremos então a seguinte proposição:

Proposição 3.1.1 *Para todo $\boldsymbol{\alpha} \in \mathbb{S}$, $f(\alpha)$ terá um valor máximo em 0 quando $N+1$ for par e um valor mínimo em 0 quando $N+1$ for ímpar. Além disso, O valor ótimo será atingido apenas nas colunas de $[\mathbf{D}]_N$, que são os vértices de $\mathbb{S}$, o que é equivalente a fazer $P_{N+1} = P_i$ para algum $i \in [1,N]$.*



$$\alpha_i \leq \alpha_j + d_{ij}$$
$$\alpha_j \leq \alpha_i + d_{ij}$$
$$d_{ij} \leq \alpha_i + \alpha_j$$

Figura 4 – Desigualdade triangular

Para provar essa proposição, vamos inicialmente verificar que a função f não possui pontos críticos em $\mathbb{S}$. Para isso, note que um ponto crítico $\boldsymbol{\alpha}^*$ teria $\nabla f(\boldsymbol{\alpha}^*) = -2adj([\mathbf{D}]_N)\boldsymbol{\alpha}^* = 0$. Perceba que caso $\boldsymbol{\alpha}^*$ possua alguma componente diferente de zero, as colunas de $adj([\mathbf{D}]_N)$ serão linearmente dependentes. Aqui destacamos que $\boldsymbol{\alpha} = [0,0,...,0] \notin \mathbb{S}$. Sabendo que $det([\mathbf{D}]_N) \neq 0$, pela hipótese de indução, e sabendo que, para qualquer matriz $\mathbf{A}$, $\mathbf{A} * adj(\mathbf{A}) =$

$|\mathbf{A}| * \mathbf{I}$, onde I é a matriz identidade, temos:

$$\begin{aligned} [\mathbf{D}]_N * adj([\mathbf{D}]_N) &= det([\mathbf{D}]_N) * I \\ \frac{1}{det([\mathbf{D}]_N)} * [\mathbf{D}]_N * adj([\mathbf{D}]_N) &= I \\ adj([\mathbf{D}]_N)^{-1} &= \frac{1}{det([\mathbf{D}]_N)} * [\mathbf{D}]_N \end{aligned} \tag{3.5}$$

Ou seja, a matriz $adj([\mathbf{D}]_N)$ é inversível. Isso quer dizer que suas colunas são LI, logo não pode haver $\boldsymbol{\alpha}^* \neq 0$ tal que $\nabla f(\boldsymbol{\alpha}^*) = 0$. O que conclui a prova de que f não possui pontos críticos em $\mathbb{S}$.

Agora vamos mostrar que quando o vetor $\boldsymbol{\alpha}$ for igual a uma das colunas de $[\mathbf{D}]_N$ o valor de $f(\boldsymbol{\alpha})$ será zero. Para isso, basta observar que fazer $\boldsymbol{\alpha}$ igual a i-ésima coluna de $[\mathbf{D}]_N$ faria a matriz $[\mathbf{D}]_{N+1}$ ter duas linhas iguais, como vemos na equação 3.6.

$$f(\boldsymbol{\alpha}) = det([\mathbf{D}]_{N+1}) = det \begin{bmatrix} 0 & d_{12} & \dots & d_{1j} & \dots & d_{1n} & d_{i1} \\ d_{12} & 0 & \dots & d_{2j} & \dots & d_{2n} & d_{i2} \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ d_{i1} & d_{i2} & \dots & d_{ij} & \dots & d_{in} & 0 \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nj} & \dots & 0 & d_{in} \\ d_{i1} & d_{i2} & \dots & d_{ij} & \dots & d_{in} & 0 \end{bmatrix} = 0 \tag{3.6}$$

Veremos a seguir que os vértices de $\mathbb{S}$ (colunas de $[\mathbf{D}]_N$) são máximos ou mínimos locais em $\mathbb{S}$, de acordo com a paridade de N. Faremos isso usando expansão de Taylor. Precisaremos então calcular o gradiente de f em cada vértice $\mathbf{v_i}$. Para isso, usaremos novamente a relação $\mathbf{A} * adj(\mathbf{A}) = |\mathbf{A}| * I$. Assim, como $\nabla f(\boldsymbol{\alpha}) = -2adj([\mathbf{D}]_N)\boldsymbol{\alpha}$, temos que $\nabla f(\mathbf{v_i}) = -2adj([\mathbf{D}]_N)\mathbf{v_i}$, mas como $[\mathbf{D}]_N * adj([\mathbf{D}]_N) = |[\mathbf{D}]_N| * I$ e $\mathbf{v_i}$ é a i-ésima coluna de $[\mathbf{D}]_N$, então teremos que $\nabla f(\mathbf{v_i}) = -2adj([\mathbf{D}]_N)\mathbf{v_i}$ será igual a (-2) vezes a i-ésima coluna da matriz $|[\mathbf{D}]_N| * I$. Ou seja:

$$\nabla f(\mathbf{v_i}) = \begin{bmatrix} \frac{\partial f}{\partial \alpha_1}(\mathbf{v_i}) \\ \vdots \\ \frac{\partial f}{\partial \alpha_i}(\mathbf{v_i}) \\ \vdots \\ \frac{\partial f}{\partial \alpha_n}(\mathbf{v_i}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ -2|[\mathbf{D}]_N| \\ \vdots \\ 0 \end{bmatrix} \tag{3.7}$$

Assim, podemos afirmar que para todo $\mathbf{h} \in \mathbb{R}^N$ suficientemente pequeno, temos $f(\mathbf{v_i}+\mathbf{h}) = \underbrace{f(\mathbf{v_i})}_{0} + \sum_{j=1}^{N} h_i \frac{\partial f}{\partial \alpha_j}(\mathbf{v_i}) = -2det([\mathbf{D}]_N)h_i$. Note que, para atender a nossa proposição nós precisamos que esse resultado tenha sinal oposto ao de $det([\mathbf{D}]_N)$ independente da paridade de N. Assim, temos que mostrar que $h_i \leq 0$ implica que $\mathbf{v_i}+\mathbf{h} \notin \mathbb{S}$. Para isso, perceba que se $h_i \leq 0$, o ponto $\mathbf{v_i}+\mathbf{h}$ representaria as distâncias entre um possível ponto P_{N+1} e os demais, sendo que a distância entre este e o ponto P_i seria dada por $d(P_{N+1},P_i) = v_{i,i}+h_i \leq v_{i,i} = 0$, mas a distância entre dois pontos distintos precisa ser positiva. Assim concluímos que dentro de $\mathbb{S}$, os vértices são pontos de máximo ou mínimo local, dependendo da paridade de N conforme apresentado na proposição 3.1.1.

Para concluir a demonstração, restaria descartar a possibilidade de haver algum outro máximo ou mínimo local de f em $\mathbb{S}$. Isso significa explorar os pontos das fronteiras dadas pelas inequações da Figura 4. Como temos 3 inequações, teríamos 3 casos para explorar. Em cada caso teríamos que tratar a desigualdade como igualdade e reescrever a função f de modo a garantir que tal igualdade seja satisfeita, encontrar o gradiente desta função e verificar se existe algum ponto crítico. Em (AUER, 1995) vemos que os pontos críticos, quando existem, são os próprios vértices de $\mathbb{S}$, permitindo concluir a demonstração.

A demonstração que acabamos de reproduzir, mostra que toda matriz de distâncias formada por N pontos distintos é inversível. No contexto do treinamento da rede MLM, isso traz a importante consequência de que, caso usemos todos os pontos do conjunto de dados como pontos de referência, a matriz $\mathbf{D}_x$, com as distâncias entre os pontos no espaço da entrada, será inversível e, com isso, teremos a matriz de regressores $\mathbf{B}$ dada por $\mathbf{D}_x^{-1}\mathbf{\Delta}_y$, logo, $\mathbf{D}_x\mathbf{B}$ será exatamente igual a $\mathbf{\Delta}_y$.

3.2 Estimação da saída

O resultado da seção anterior é importante por mostrar que a MLM consegue recuperar com erro zero as distâncias entre os pontos de referência do conjunto de dados usado no treinamento. Entretanto, isso não é suficiente para afirmar que ela é capaz de interpolar esse conjunto de dados. Para isso, precisamos avaliar a capacidade do modelo estimar a saída, i.e., recuperar a posição dos pontos no espaço da saída, a partir das distâncias perfeitamente encontradas. Essa etapa será discutida nesta seção.

Aqui faremos a estimação da saída será feita através da trilateração. Essa técnica é

usada para determinar a posição de um ponto a partir da estimativa de distancias para pontos fixos conhecidos. A rigor, trilateração trabalha no $\mathbb{R}^2$ e usa 3 pontos para determinar a localização desconhecida. Neste trabalho consideraremos o caso mais geral, onde os dados estarão em $\mathbb{R}^L$ e usaremos N pontos fixos conhecidos, técnica chamada de multilateração. Essa generalização é quase imediata e não trará nenhum dano às técnicas aqui empregadas.

Uma forma de resolver o problema proposto é usar otimização não-linear para determinar o ponto que minimiza o erro quadrático entre a distância estimada e a real, calculada em cada ponto candidato. Essa técnica vem sendo empregada e apresenta resultados satisfatórios. Entretanto, neste trabalho gostaríamos de verificar se, caso as distâncias sejam estimadas corretamente, a posição do ponto poderá ser recuperada sem erro. Veremos aqui uma técnica alternativa que facilita essa análise, mostrando em que condições a recuperação exata da posição é possível.

Para isso, considere $\mathbf{X} \in \mathbb{R}^{N\times L}$, o conjunto de N pontos conhecidos em $\mathbb{R}^L$. Suponha ainda a existência de $\mathbf{x} \in \mathbb{R}^L$ desconhecido, mas cujas distancias para cada $\mathbf{x}_i \in \mathbf{X}$, dadas por $||\mathbf{x}-\mathbf{x}_i||^2 = d_i^2$, são conhecidas. Suponha ainda que tenhamos um outro ponto $\mathbf{r} \in \mathbb{R}^L$ (possivelmente um ponto extraído de $\mathbf{X}$), tal que $||\mathbf{x}-\mathbf{r}||^2 = d_r^2$ e $||\mathbf{x}_i-\mathbf{r}||^2 = d_{ir}^2$ também são conhecidas. Assim, temos:

$$
\begin{aligned}
||\mathbf{x}-\mathbf{x}_i||^2 &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\sum_{j=1}^{L}(x_j-x_{i,j})^2 &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\sum_{j=1}^{L}(x_j-r_j+r_j-x_{i,j})^2 &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\sum_{j=1}^{L}[(x_j-r_j)+(r_j-x_{i,j})]^2 &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\sum_{j=1}^{L}[(x_j-r_j)-(x_{i,j}-r_j)]^2 &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\sum_{j=1}^{L}[(x_j-r_j)^2+(x_{i,j}-r_j)^2-2(x_j-r_j)(x_{i,j}-r_j))] &= d(\mathbf{x}_i,\mathbf{x})^2 \\
\underbrace{\sum_{j=1}^{L}(x_j-r_j)^2}_{d(\mathbf{x},\mathbf{r})^2}+\underbrace{\sum_{j=1}^{L}(x_{i,j}-r_j)^2}_{d(\mathbf{r},\mathbf{x}_i)^2}-2\sum_{j=1}^{L}(x_j-r_j)(x_j^i-r_j)) &= d(\mathbf{x}_i,\mathbf{x})^2 \\
-2\sum_{j=1}^{L}(x_j-r_j)(x_{i,j}-r_j)) &= d(\mathbf{x}_i,\mathbf{x})^2-d(\mathbf{x},\mathbf{r})^2-d(\mathbf{r},\mathbf{x}_i)^2 \\
\sum_{j=1}^{L}\underbrace{(x_j-r_j)}_{\theta_j}\underbrace{(x_{i,j}-r_j))}_{A_{ij}} &= \underbrace{\frac{1}{2}[d(x,r)^2+d(r,\mathbf{x}_i)^2-d(\mathbf{x}_i,\mathbf{x})^2]}_{b_i}
\end{aligned}
$$

Assim, após resolver o sistema $\mathbf{A}\boldsymbol{\theta}=\mathbf{b}$, fazemos $\mathbf{x}=\boldsymbol{\theta}+\mathbf{r}$ para recuperar a posição de $\mathbf{x}$. Observe que o ponto $\mathbf{r}$ pode ser selecionado dentre os pontos de $\mathbf{X}$ e assim atender a todas as condições necessárias à aplicação da técnica.

Vale observar que resolver o sistema $\mathbf{A}\boldsymbol{\theta}=\mathbf{b}$ com exatidão só é possível quando $\mathbf{A}$ for não-singular. A priori isso não é necessariamente verdade para qualquer conjunto de pontos $\mathbf{X}$. Veremos na próxima seção em que condições isso é verdade.

3.2.1 Condições para não singularidade da matriz A

Normalmente temos a "oferta"de uma quantidade de pontos conhecidos bem maior que a dimensão dos dados. Considerando que tenhamos as distâncias estimadas perfeitamente, poderíamos então selecionar apenas $L+1$ amostras para formar a nossa matriz A e ainda assim chegar na solução exata. Porém, e se esses pontos formarem uma matriz singular? Existe alguma propriedade que devemos procurar enquanto selecionamos os $L+1$ pontos? Como os dados estão em $\mathbb{R}^L$, quaisquer $L+1$ pontos serão sempre linearmente dependentes, mas e se selecionarmos

L pontos LI e um outro, que condições esse outro ponto deve ter para que a matriz construída conforme visto na seção anterior seja não singular? Veremos a seguir que para construir uma matriz $\mathbf{A}$ inversível, devemos selecionar $L+1$ pontos **afim independentes**.

Definição: (Combinação Afim) Dado um conjunto $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \in \mathbb{R}^L$, dizemos que $\mathbf{v} \in \mathbb{R}^L$ é uma **combinação afim** dos vetores dados se existem $\alpha_1, \alpha_2, \ldots, \alpha_N$ tais que $\sum \alpha_i = 1$ e $\sum \alpha_i \mathbf{v}_i = \mathbf{v}$.

Teorema 3.2.1 (Condição suficiente para a inversibilidade de A) *Dados um conjunto* $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N \in \mathbb{R}^L$ *e* $\mathbf{v} \in \mathbb{R}^L$*, os vetores* $\mathbf{v}_1 - \mathbf{v}, \mathbf{v}_2 - \mathbf{v}, \ldots, \mathbf{v}_N - \mathbf{v}$ *são LI se e somente se* $\mathbf{v}$ *não é uma combinação afim de* $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.

Prova:

Suponha, por absurdo, que $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ é um conjunto LI, que $\mathbf{v}$ não é uma combinação afim de V e que $\mathbf{V}' = \{\mathbf{v}_1 - \mathbf{v}, \ldots, \mathbf{v}_n - \mathbf{v}\}$ é LD. Assim, existem $\mu_1, \ldots, \mu_n$, não todos nulos, tais que:

$$\begin{aligned}
\sum_{i=1}^{N} \mu_i (\mathbf{v}_i - \mathbf{v}) &= 0 \\
\sum_{i=1}^{N} \mu_i (\mathbf{v}_i - \sum_{j=1}^{N} \lambda_j \mathbf{v}_j) &= 0 \\
\sum_{i=1}^{N} \mu_i \mathbf{v}_i - \sum_{i=1}^{N} \mu_i \sum_{j=1}^{N} \lambda_j \mathbf{v}_j &= 0 \\
\sum_{i=1}^{N} \mu_i \mathbf{v}_i - \sum_{i=1}^{N} \lambda_i \mathbf{v}_i \sum_{j=1}^{N} \mu_j &= 0 \\
\sum_{i=1}^{N} (\mu_i \mathbf{v}_i - \lambda_i \mathbf{v}_i \sum_{j=1}^{N} \mu_j) &= 0 \\
\sum_{i=1}^{N} \underbrace{(\mu_i - \lambda_i \sum_{j=1}^{N} \mu_j)}_{\theta_i} \mathbf{v}_i &= 0 \\
\sum_{i=1}^{N} \theta_i \mathbf{v}_i &= 0 \qquad (3.8)
\end{aligned}$$

Como V é LI, a equação acima só pode ser satisfeita quando todos os θ_i forem iguais a zero. Isso resulta em $\mu_i = \lambda_i \sum \mu_i \ \forall i$. Se $\sum \mu_i = 0$ teríamos $\mu_i = \lambda_i 0 = 0 \ \forall i$, o que não pode ser verdade, pois supomos que μ_i não são todos nulos. Assumindo então que $\sum \mu_i \neq 0$ temos $\lambda_i = \frac{\mu_i}{\sum \mu_i}$, mas isso faz $\sum \lambda_i = 1$. Como supomos que $\mathbf{v}$ não é combinação afim de $\mathbf{V}$, chegamos a nova contradição e concluímos a prova.

Esse teorema nos permite concluir que o problema da trilateração, quando as distancias são estimadas com erro zero, levariam a uma solução exata sempre que usarmos $L+1$ pontos de referencia no espaço da saída formarem um conjunto afim independente.

3.3 Conclusão

Neste capítulo estudamos a capacidade de interpolação da MLM em relação um conjunto arbitrário de dados. Esse estudo foi feito em duas etapas:

1. demonstrar a inversibilidade da matriz de distâncias;
2. encontrar uma condição suficiente para a correta estimação da saída, a partir da estimação das distâncias exata.

Na primeira etapa vimos que um conjunto de dados formado por N pontos distintos em $\mathbb{R}^L$ sempre dará origem a uma matriz de distâncias não singular, permitindo que a estimação das distâncias no espaço da saída seja feita com exatidão. De posse dessa informação passamos para a segunda etapa, buscando uma condição suficiente para a correta estimação da saída, partindo da exata estimação da saída. Aqui vimos que isso será possível sempre que, no espaço da saída, a amostra que se deseja recuperar formar um conjunto afim independente com os demais pontos de referência. Isso conclui o nosso estudo sobre a capacidade de interpolação da MLM. A capacidade de interpolação é uma propriedade importante, pois mostra que o modelo não é excessivamente simples, além de ser um passo na direção de demonstrar uma outra garantia teórica importante, a capacidade de aproximação universal de funções.

4 AVALIAÇÃO EXPERIMENTAL DA MÁQUINA DE APRENDIZAGEM MÍNIMA

Como vimos no Capítulo 2, a máquina de aprendizagem mínima faz um mapeamento linear entre uma matriz de distâncias computada no espaço da entrada e uma matriz de distâncias no espaço da saída. Essas matrizes são formadas calculando a distância entre cada ponto de treinamento para um subconjunto chamado **pontos de referência**. Com isso, conseguimos recuperar uma estimação das distâncias de um novo ponto para os pontos de referência no espaço da saída. Em seguida, pode-se aplicar alguma técnica para recuperar uma estimativa para a saída desse novo ponto. Neste capítulo, realizaremos alguns experimentos com o objetivo de avaliar os principais pontos da MLM e de que forma eles podem influenciar o seu desempenho.

Na Seção 4.1 veremos algumas alternativas para a seleção dos pontos de referência e veremos o impacto da escolha do método no resultado da MLM e na quantidade de pontos necessários. A Seção 4.3 estuda o impacto do número de pontos de referências e do nível de ruído, que corrompe as estimativas da distância, na capacidade da estimação da saída de recuperar a posição do ponto no espaço de saída.

4.1 Seleção dos pontos de referência

Na proposta original do MLM, os pontos de referência eram selecionados aleatoriamente. Com isso, podemos dizer que a MLM possui sua capacidade de realizar uma aproximação não linear baseada em uma projeção aleatória dos dados de entrada. Essa característica, que também é presente em outros métodos, como a máquina de aprendizado extremo (*Extreme Learning Machine, ELM*) Huang *et al.* (2012), vem sendo estudada recentemente. Isso se deve, dentre outros fatores, ao fato de que tais métodos normalmente possuem poucos hiper-parâmetros e o tempo de treinamento costuma ser menor. Por outro lado, pouco se pode afirmar sobre a capacidade de generalização e estabilidade desses modelos.

Um dos objetivos deste trabalho é estudar outros métodos de seleção de pontos de referência e comparar seus resultados. A seguir veremos algumas possibilidades consideradas e em seguida os resultados de uma comparação experimental entre eles.

4.1.1 Successive Projection Algorithm - SPA

SPA é um método de seleção de atributos proposto em Araújo *et al.* (2001), que visa reduzir a redundância de informação das variáveis selecionadas. Essa redução se dá pela busca de

variáveis menos correlacionadas (a partir de uma variável inicial fornecida).

Considerando que o algoritmo foi descrito para seleção de atributos, a aplicação à seleção de pontos de referência não é imediata, mas é bem simples. Se tivermos um conjunto de dados representado por uma matriz onde cada linha representa uma amostra e cada coluna representa um atributo, usando o SPA, informa-se uma coluna base e o número k de colunas desejada e ele me retornaria uma nova matriz com a mesma quantidade de linhas e k colunas. Sendo essas colunas as menos correlacionadas (a partir da coluna inicial dada). Para aplicar essa metodologia à seleção de pontos de referência, construímos um conjunto de dados "alternativo"com n linhas e n colunas (onde n é a quantidade de dados). Esse conjunto de dados seria, na formulação do MLM, a matriz D_x com todos os pontos sendo pontos de referência. Note que aplicar o SPA a essa matriz é equivalente a selecionar pontos de referência. A coluna inicial fornecida neste trabalho foi escolhida aleatoriamente.

4.1.2 K-Centros

K-centros é um algoritmo de agrupamento onde dado um conjunto $\mathscr{X}$, com N pontos, busca-se selecionar um subconjunto $\mathscr{R}$ de $\mathscr{X}$, com K elementos (chamados de **centros**), de modo a minimizar a maior distancia entre um ponto e o seu centralizador (ou seja, o centralizador mais próximo a ele). Em outras palavras, podemos definir uma função $d_{\mathscr{R}}(\mathbf{x}) = min_{\mathbf{r}\in\mathscr{R}} d(\mathbf{x},\mathbf{r})$, que representa a menor distância entre um elemento $\mathbf{x} \in \mathscr{X}$ e algum $\mathbf{r} \in \mathscr{R}$. Assim, o objetivo do k-centros é encontrar o conjunto $\mathscr{R}$ que minimiza o valor de $max_{\mathbf{x}\in\mathscr{X}} d_{\mathscr{R}}(\mathbf{x})$.

Uma consequência potencialmente problemática dessa metodologia é que pontos muito isolados, que possivelmente fossem outliers, tendem a ser eles considerados centralizadores, pois caso contrário a distancia dele para um centralizador poderia ser enorme. Mais informações sobre o K-Centros pode ser encontrada em Malkomes *et al.* (2015) e Gonzalez (1985).

4.1.3 K-medoides

Proposto em Vinod (1969), o método k-medoides é semelhante ao k-centros, com a diferença que o objetivo é minimizar a soma das distâncias entre um ponto do conjunto de dados e o ponto centralizador mais próximo a ele. Isto é, dado um conjunto $\mathscr{X}$, com N pontos, busca-se selecionar um subconjunto $\mathscr{R}$ de $\mathscr{X}$, com K elementos (chamados de **objetos representativos**

ou **medoides**), de modo a minimizar a soma das distancias entre cada ponto e o seu centralizador (ou seja, o centralizador mais próximo a ele). Esse processo é feito através do problema de programação inteira formulado da seguinte forma:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} d(\mathbf{x}_i, \mathbf{x}_j) z_{ij}$$

Sujeito à:

$$\sum_{i=1}^{N} z_{ij} = 1 \forall j \in \{1, ..., N\} \tag{4.1}$$

$$z_{ij} \leq y_i \forall i, j \in \{1, ..., N\} \tag{4.2}$$

$$\sum_{i=1}^{N} y_i = K \tag{4.3}$$

$$y_i, z_{ij} \in \{0, 1\} \forall i, j \in \{1, ..., N\} \tag{4.4}$$

onde as restrições 4.1 e 4.4 garantem que cada amostra $\mathbf{x}_i \in \mathscr{X}$ deve estar associado a um único medoide $\mathbf{x}_j$. A restrição 4.2 assegura que um ponto $\mathbf{x}_j$ só pode ser associado a um medoide $\mathbf{x}_i$ se este último tiver sido marcado como um objeto representativo. A restrição 4.3 garante que o número de centralizadores escolhido será dado pelo hiperparâmetro K. Mais informações sobre o k-medoids pode ser encontrada em Cao e Yang (2010) e Vinod (1969).

4.2 Resultados Obtidos

Nesta seção descreveremos os experimentos desenvolvidos ao longo da nossa pesquisa. Com eles visamos comparar o efeito do método de seleção dos pontos de referência no desempenho da rede MLM, medida através da acurácia (para dados de classificação) e erro quadrático médio (em dados de regressão).

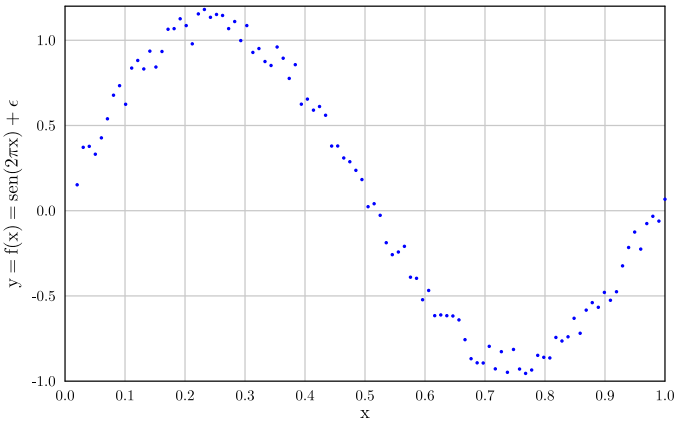
A Subseção a seguir descreve os experimentos feitos em conjuntos de dados artificiais. Os dados usados são bidimensionais, nos permitindo uma visualização geométrica dos resultados obtidos. Aqui foi trabalhada a tarefa de regressão

Na Subseção seguinte, trazemos os resultados de experimentos obtidos em bases de dados reais.

4.2.1 Dados artificiais

O primeiro experimento descrito tem o objetivo de avaliar o erro quadrático médio quando variamos o valor de K, além de comparar os resultados obtidos por diferentes métodos de seleção de pontos de referência. Para isso nós usamos um conjunto de dados artificial que pode ser visto na Figura 5. Trata-se da função $f(x) = sen(2\pi x)$ corrompida por um ruído gaussiano de amplitude 0.2, computados em 120 pontos amostrados uniformemente entre 0 e 1.

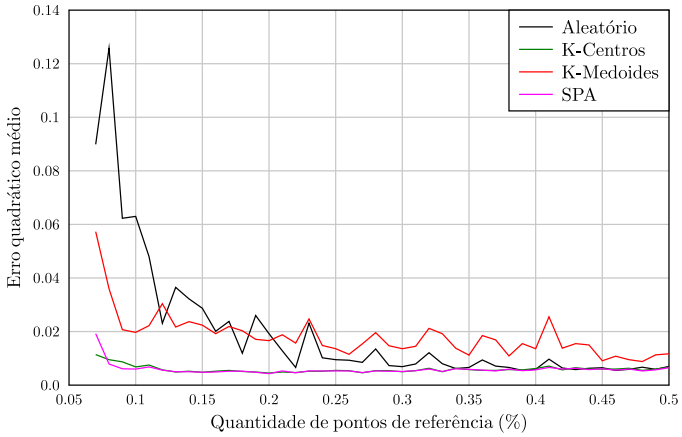
Figura 5 – Conjunto de dados artificial usado para ilustrar o efeito do método de seleção dos pontos de referência.



No experimento foram testados valores de K entre 5% e 50% do tamanho do conjunto de dados, com incremento de 1%. Os dados foram divididos em 80% para treinamento e 20% para testes e o experimento foi repetido 20 vezes. O erro que consideramos foi a média do erro obtido em cada execução. A Figura 6 apresenta o erro obtido. O gráfico nos permite notar que o método de seleção usado interfere menos no resultado a medida que o número de pontos de referência aumenta. Isso faz sentido, se considerarmos, por exemplo, o caso extremo de usar todo o conjunto de dados como ponto de referência, onde método de seleção seria completamente irrelevante. Podemos ver também que, usando menos pontos de referências, qualquer método de seleção testado apresenta resultados melhores que a seleção aleatória.

Usamos outras bases de dados bidimensionais para visualizar o resultado produzido pela MLM com cada um dos métodos de seleção estudados. Além da função $f(x) = seno(2\pi x)$ que já descrevemos, usamos também a função normal corrompida com um ruído gaussiano de amplitude 0.02 e a função logística com ruído de 0.1.

Figura 6 – Métodos de seleção de pontos de referência (Erro quadrático médio versus K)



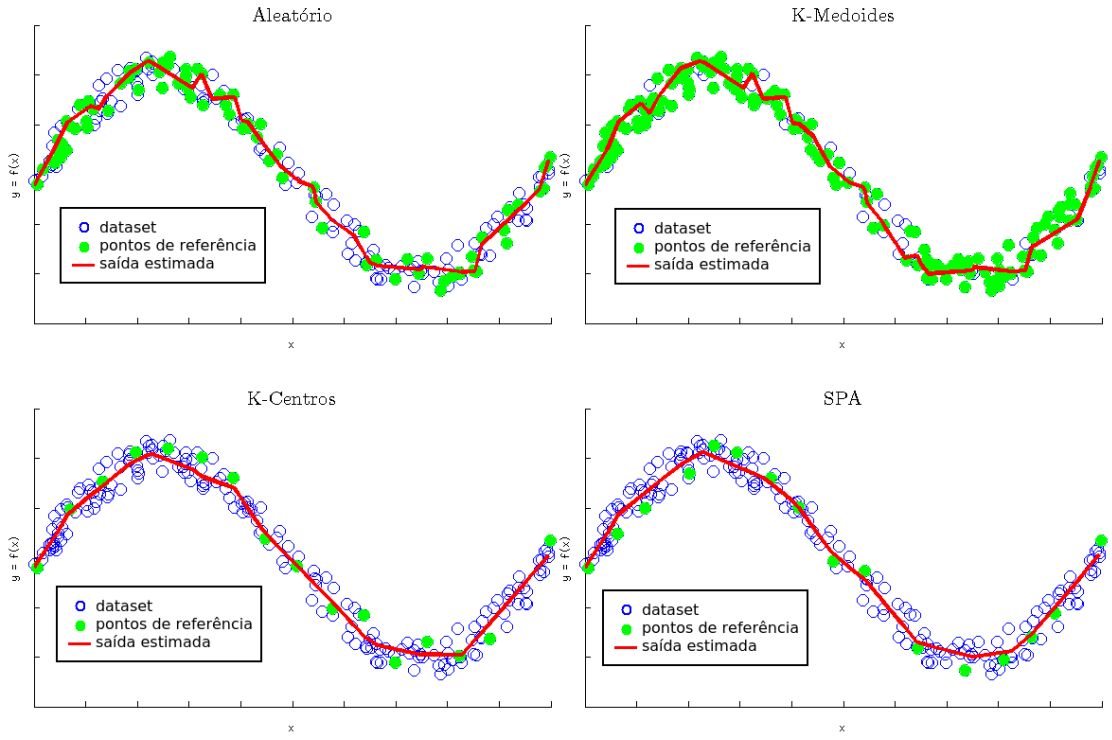
Neste experimento nós usamos validação cruzada com 10 *folds* para determinar o número de pontos de referência ótimo para cada método. Os valores de K testados nessa etapa variavam entre 10% e 100% dos dados, com incremento de 10%. Cada conjunto de dados possuía 200 amostras que foram divididas em 80% para treinamento e 20% para teste.

Os resultados obtidos aparecem na Figura 7, para a função seno, Figura 8 para a função normal, Figura 9 para a função logística. Em todos esses gráficos, os pontos do conjunto de dados aparecem em azul, os pontos de referência em verde e a curva gerada pela MLM nos dados de teste aparece em vermelho, conforme indicado nas legendas. A observação dessas figuras nos permitiu notar que as curvas obtidas com K-Centros e SPA, em geral, possuem menos variações bruscas que as geradas pelos métodos Aleatório e K-Medoides. Além disso, notamos também que o K-Medoides apresenta muitos pontos de referência. Essa observação se comprova através da Figura 10, que apresenta a quantidade percentual de pontos de referência retornada na validação cruzada em cada experimento.

4.2.2 Dados reais

A seguir discutiremos os resultados obtidos em algumas simulações feitas com bases de dados reais, extraídas de Lichman (2013). Para isso, os dados são divididos em um conjunto de treinamento, com 80% dos dados, e teste, com os 20% restantes. Tomaremos como métrica o desempenho médio obtido nos dados de teste de uma rede MLM após trinta execuções, alterando apenas o método de seleção e a quantidade de pontos de referência. A métrica de avaliação considerada será o erro quadrático médio em tarefas de regressão e a taxa de classificação correta

Figura 7 – Regressão aplicada a função seno.



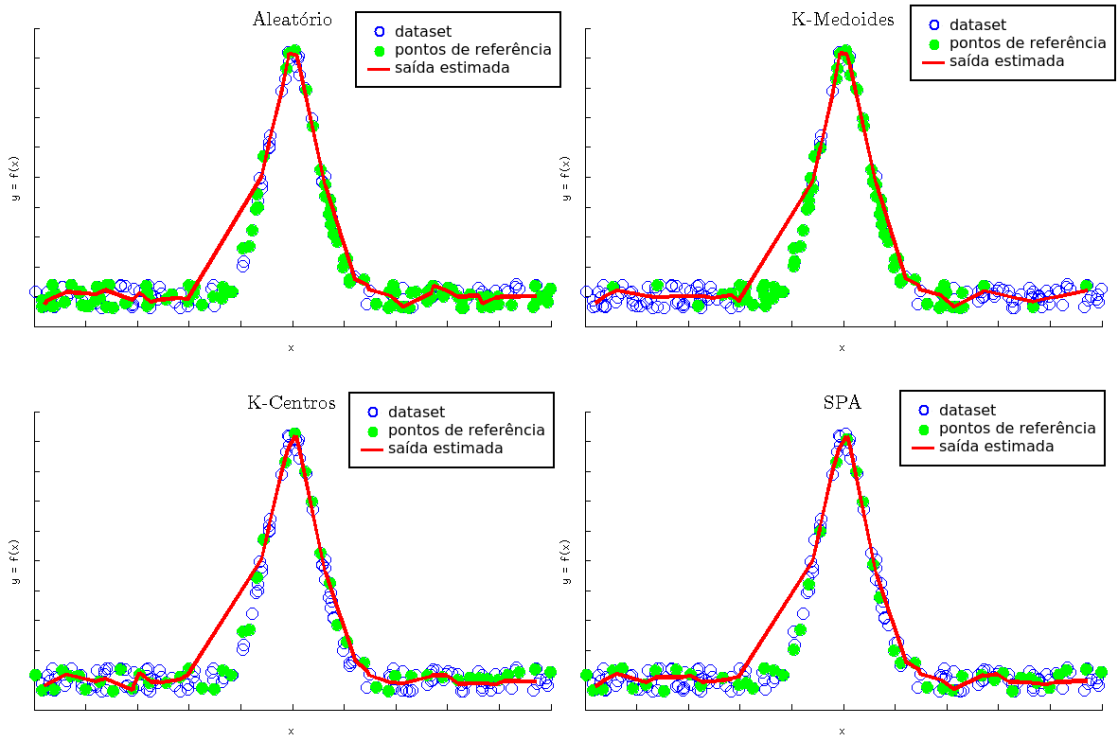
(ou acurácia) em tarefas de classificação. Os números de pontos de referência considerado foram 1%, 10% e 20% dos dados de treinamento.

Os dados usados na tarefa de regressão estão descritos na Tabela 4 e os resultados obtidos aparecem na Tabela 6. A descrição dos conjuntos de dados de classificação usados está na Tabela 3 e os resultados na Tabela 5. Tais informações também estão disponíveis em gráficos de barras das Figuras 11 a 27, que facilita a comparação dos resultados.

Analisando os resultados obtidos, pudemos perceber que:

1. a quantidade de pontos de referência é um fator mais determinante para o resultado que a forma de seleção, de modo que a variação da quantidade costuma provocar o mesmo efeito (melhorar ou piorar) em todos os métodos.
2. como observado nas bases de dados artificiais, uma grande quantidade de pontos de referência tende a reduzir a influencia do método de seleção. Isso fica mais claro observando o resultado dos dados (COM), que é o conjunto de dados com mais instâncias dentre os que testamos. Note que fixando o número de pontos de referência, quase não há variação nos diferentes métodos (ver Figura 23). Por outro lado, uma base de dados bem menor,

Figura 8 – Regressão aplicada a função normal.



como a CPU, apresenta resultados bem diferentes em cada método, especialmente quando usamos apenas 1% dos dados como pontos de referência, como fica evidente na figura 24.

Tabela 3 – Descrição dos conjuntos de dados de classificação

Dataset	**#Amostras**	**#Atributos**	**#Classes**
Balance (BAL)	625	4	3
Breast (BRE)	569	30	2
Diabetes (DBT)	768	8	2
Glass (GLA)	214	10	2
Hayes (HAY)	160	3	3
Heart (HEA)	270	13	2
Liver (LIV)	345	6	2
Monk1 (MK1)	556	6	2
Monk2 (MK2)	601	6	2
Monk3 (MK3)	554	6	2
Sonar (SON)	208	60	2
Wine (WNE)	178	13	3

Figura 9 – Regressão aplicada a função logística.

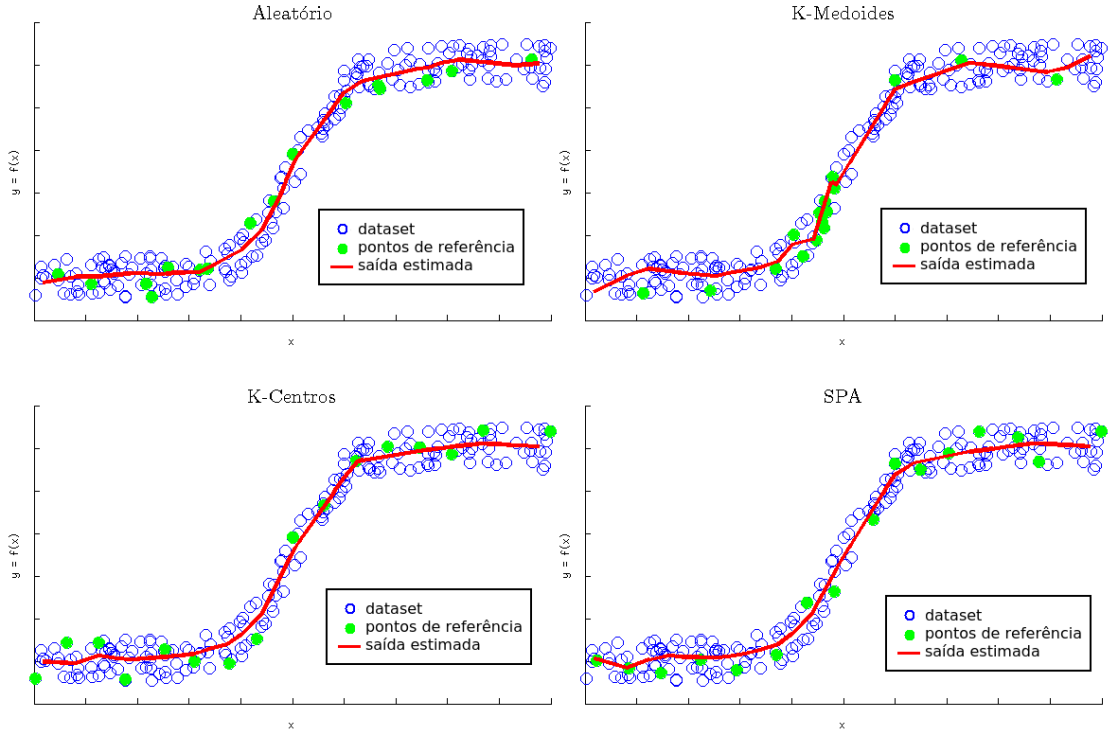


Figura 10 – Quantidade de pontos de referências retornada pelo processo de validação cruzada.

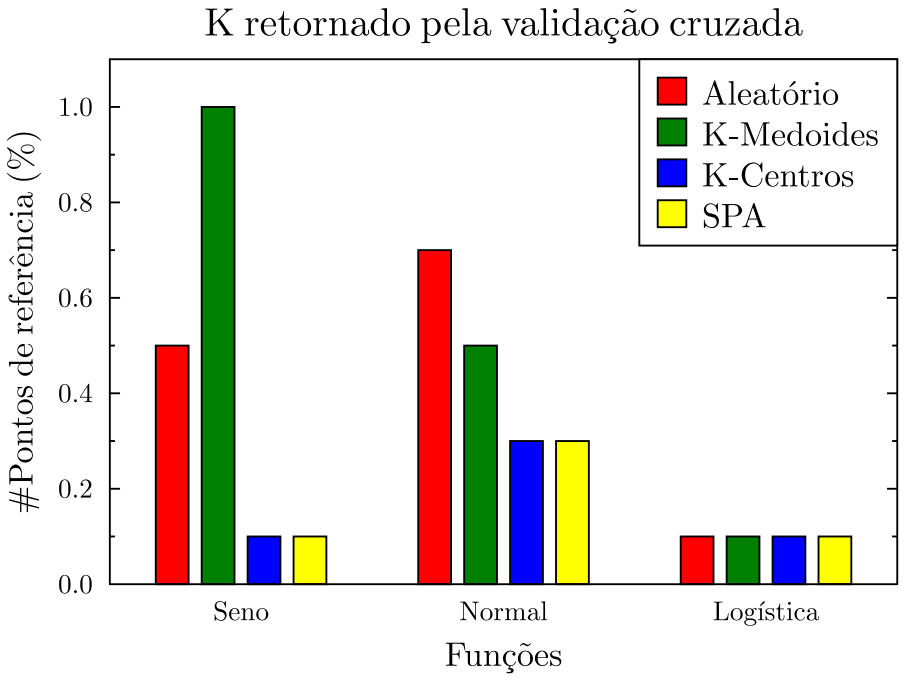


Tabela 4 – Descrição dos conjuntos de dados de regressão

Conj. de dados	**Amostras**	**Atributos**
computer hardware (COM)	1030	8
cpu (CPU)	209	9
facebook (FCB)	495	20
housing (HOU)	506	13
auto mpg (MPG)	392	7

Tabela 5 – Acurácia obtida nos conjuntos de dados de classificação

	Aleatório			**K-Medoides**			**K-Centros**			**SPA**		
	1%	**10%**	**20%**	**1%**	**10%**	**20%**	**1%**	**10%**	**20%**	**1%**	**10%**	**20%**
Balance	0.82	0.90	0.90	0.83	0.91	0.91	0.62	0.90	0.90	0.81	0.90	0.90
Breast	0.85	0.96	0.97	0.93	0.96	0.97	0.91	0.96	0.97	0.93	0.96	0.97
Diabetes	0.72	0.75	0.74	0.70	0.75	0.75	0.73	0.76	0.76	0.74	0.76	0.75
Glass	0.78	0.92	0.94	0.77	0.93	0.94	0.82	0.94	0.96	0.76	0.94	0.96
Hayes	0.16	0.67	0.80	0.46	0.79	0.82	0.29	0.75	0.86	0.06	0.66	0.83
Heart	0.58	0.82	0.82	0.73	0.83	0.83	0.71	0.82	0.82	0.68	0.82	0.82
Liver	0.56	0.69	0.69	0.56	0.69	0.69	0.58	0.68	0.68	0.57	0.67	0.68
Monk1	0.53	0.69	0.76	0.50	0.69	0.77	0.50	0.69	0.77	0.55	0.69	0.75
Monk2	0.65	0.67	0.73	0.67	0.67	0.74	0.67	0.62	0.73	0.67	0.67	0.73
Monk3	0.56	0.86	0.91	0.50	0.82	0.89	0.69	0.85	0.91	0.57	0.84	0.90
Sonar	0.54	0.73	0.77	0.56	0.74	0.77	0.55	0.70	0.77	0.52	0.74	0.77
Wine	0.54	0.96	0.98	0.60	0.98	0.98	0.66	0.97	0.98	0.59	0.98	0.98

Figura 11 – Acurácia obtida no conjunto de dados BAL

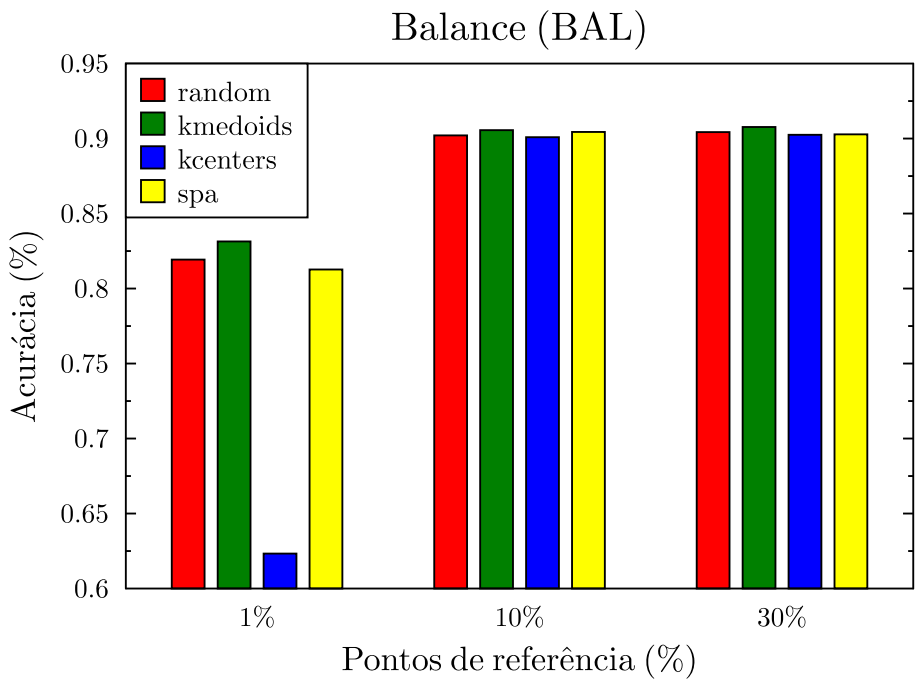


Figura 12 – Acurácia obtida no conjunto de dados BRE

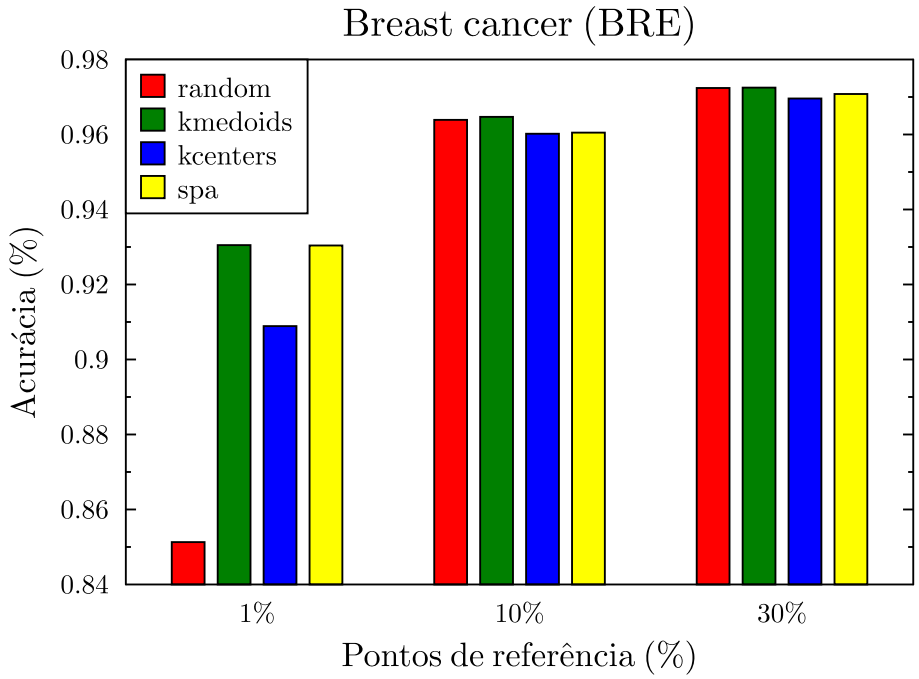
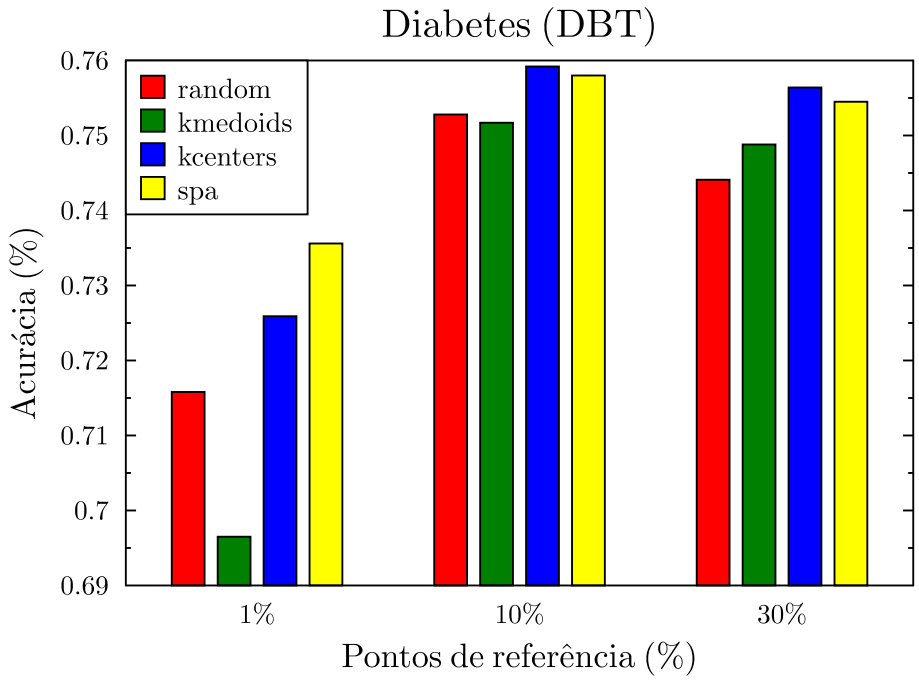


Figura 13 – Acurácia obtida no conjunto de dados DBT



4.3 Estimação da saída

Foram realizados testes computacionais para avaliar o comportamento do método de estimação da saída visto e do método tradicional baseado em otimização não linear. Nos testes, avaliamos

Figura 14 – Acurácia obtida no conjunto de dados GLA

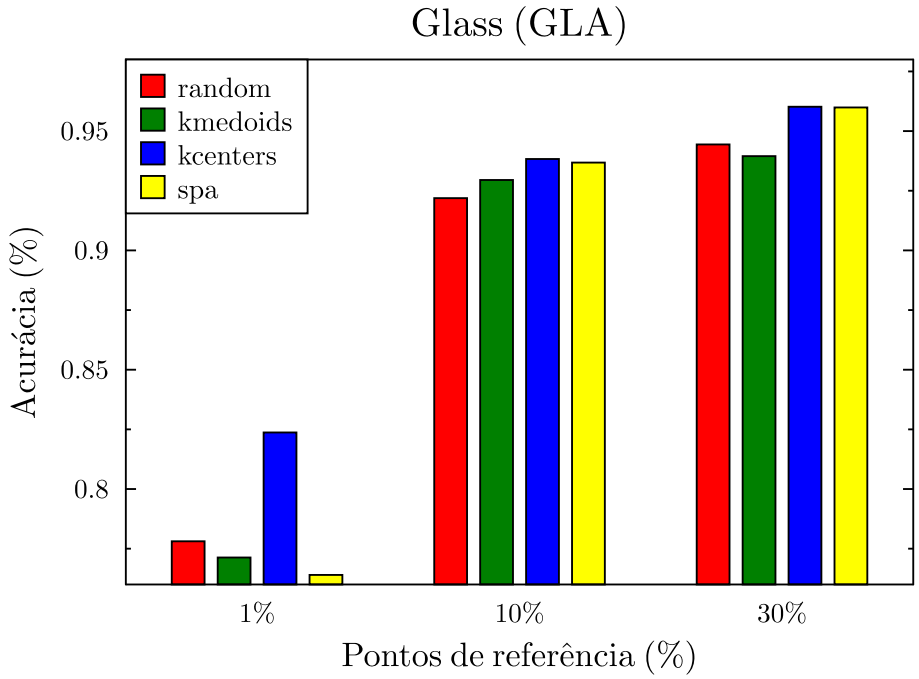
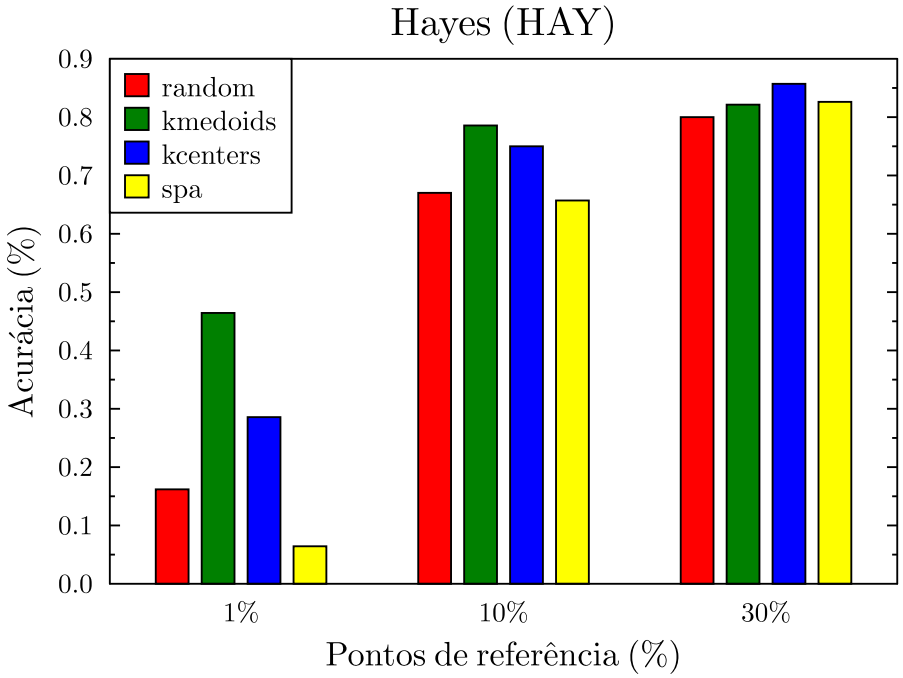


Figura 15 – Acurácia obtida no conjunto de dados HAY



a influencia do nível e tipo de ruido, da dimensão dos dados e da quantidade de pontos de referência, nas estimativas da distância.

Figura 16 – Acurácia obtida no conjunto de dados HEA

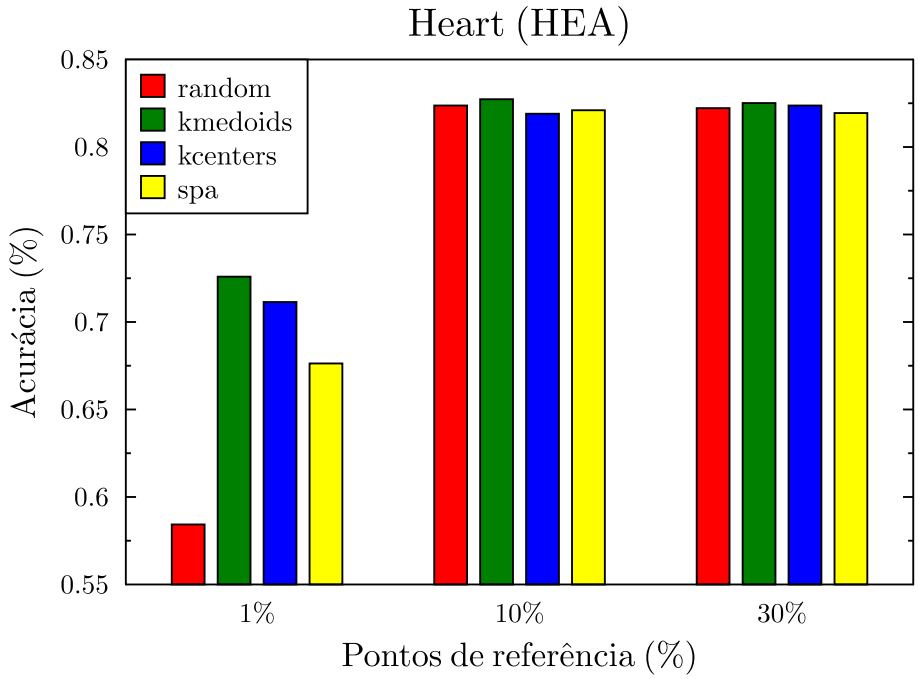
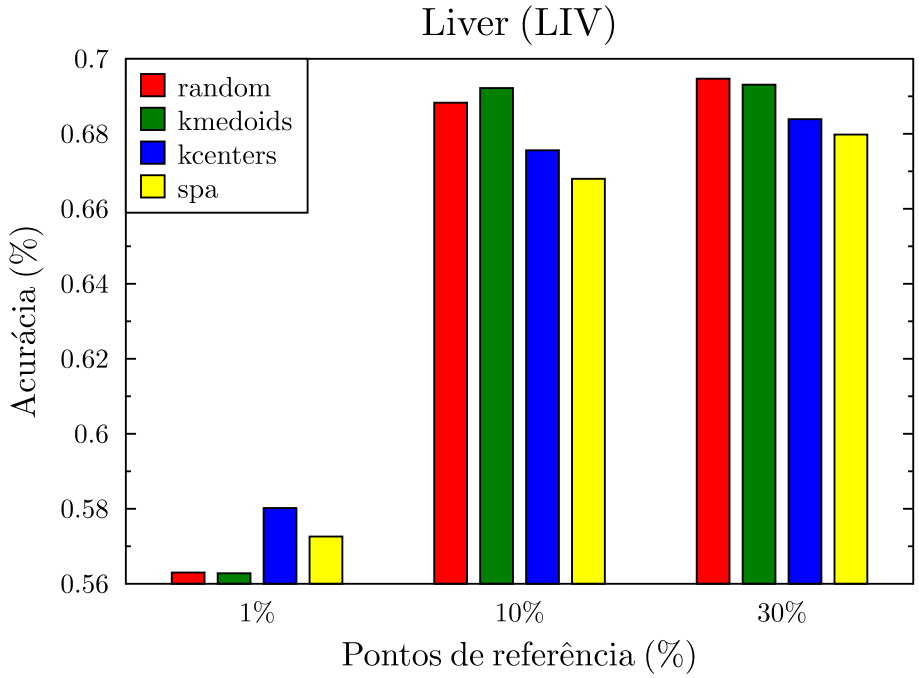


Figura 17 – Acurácia obtida no conjunto de dados LIV



O experimento consiste em sortear um ponto base P_0 e K pontos de referência $\{P_1, ..., P_k\}$, calcular a matriz (que nesse caso é um vetor) de distâncias entre P_0 e cada ponto de

Figura 18 – Acurácia obtida no conjunto de dados MK1

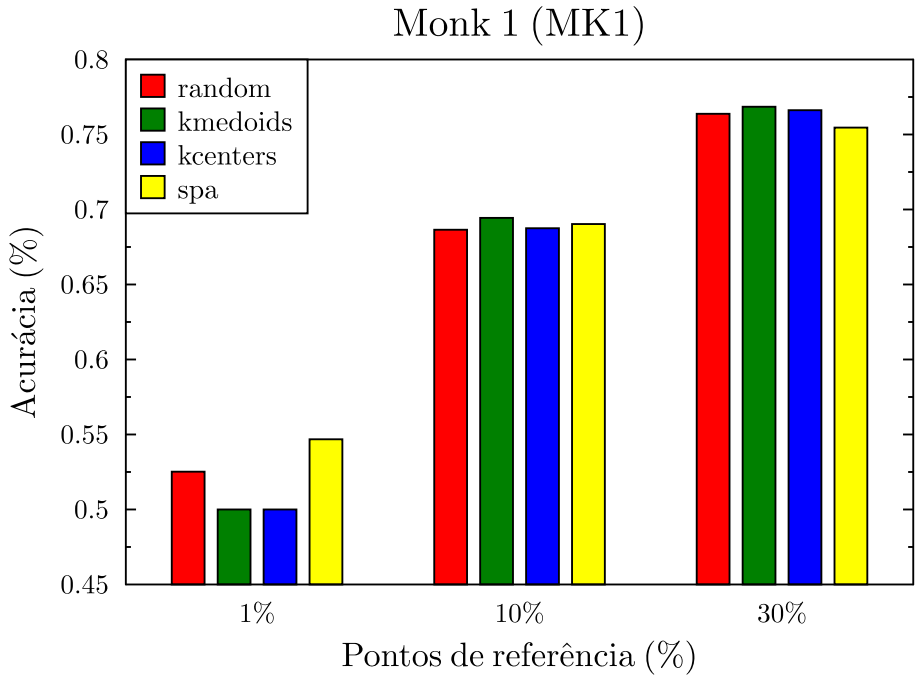
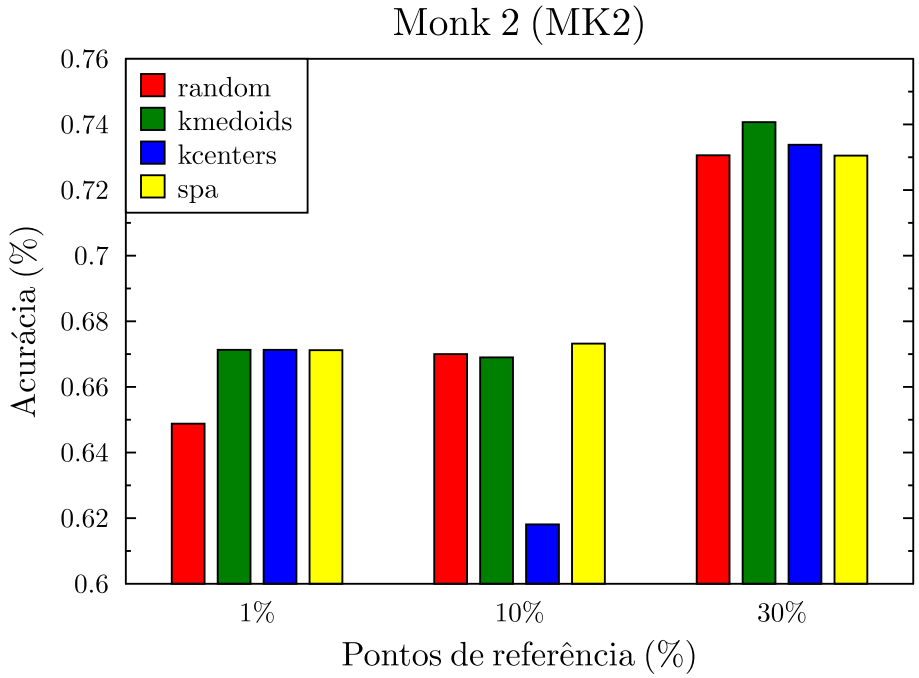


Figura 19 – Acurácia obtida no conjunto de dados MK2



referência P_i, corromper essas distâncias com um ruído, cujo tipo e magnitude variam em cada experimento, e usar esse vetor corrompido para encontrar uma estimativa $\hat{P}_0$ para a posição de P_0

Figura 20 – Acurácia obtida no conjunto de dados MK3

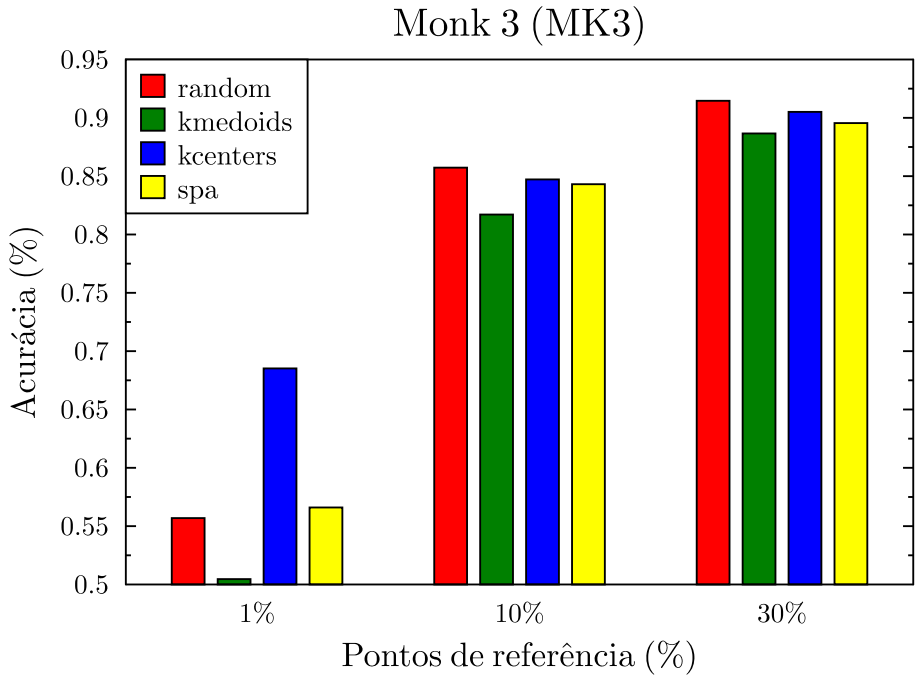
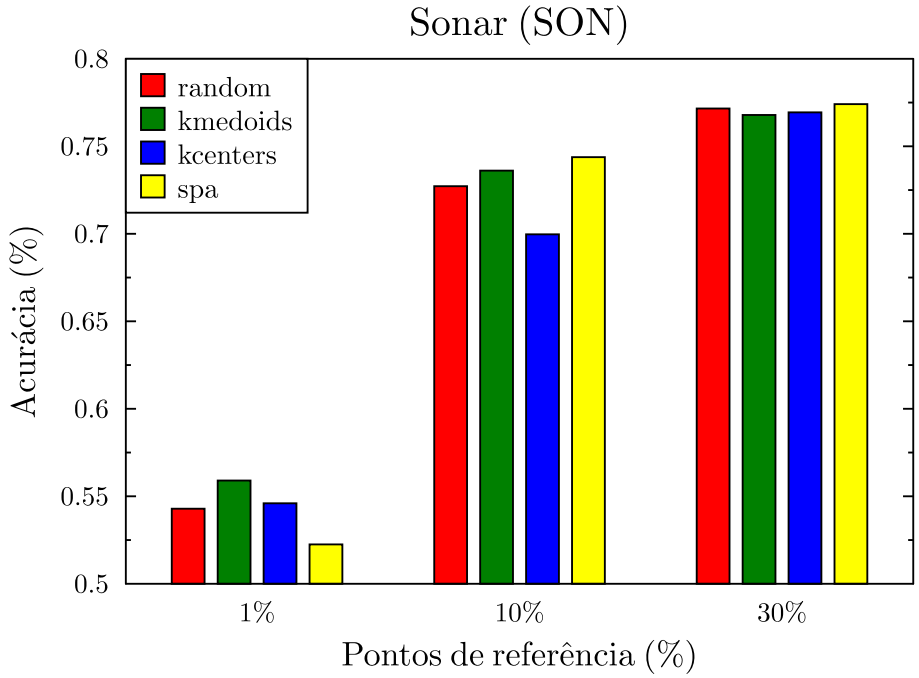


Figura 21 – Acurácia obtida no conjunto de dados SON



usando os métodos linear (visto neste trabalho) e o método tradicional fSolve, de otimização não linear. A métrica usada foi a distância euclidiana entre o ponto original P_0 e a estimativa

Figura 22 – Acurácia obtida no conjunto de dados WNE

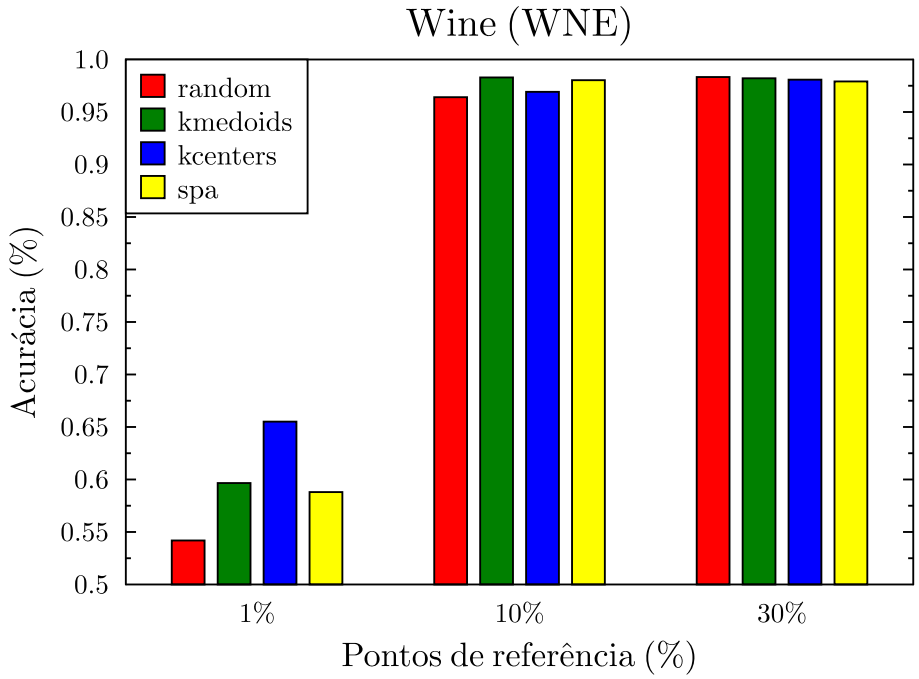


Tabela 6 – Erro quadrático médio obtido nos conjuntos de dados de regressão

	Aleatório			**K-Medoides**			**K-Centros**			**SPA**		
	1%	**10%**	**20%**	**1%**	**10%**	**20%**	**1%**	**10%**	**20%**	**1%**	**10%**	**20%**
COM	160.74	63.10	51.69	166.62	71.72	60.49	159.05	62.32	53.88	175.78	66. 30	54.18
CPU	17943.11	3520.83	2648.41	8988.52	3649.80	3115.37	6436.34	1121.05	1084.28	6417.95	1193.72	1057.66
FCB	1.03	0.70	0.67	1.13	0.72	0.71	0.75	0.53	0.54	0.80	0.53	0.54
HOU	0.48	0.20	0.17	0.40	0.23	0.21	0.41	0.17	0.15	0.47	0.18	0.14
MPG	18.81	8.05	7.85	16.25	8.78	8.25	20.74	7.67	7.13	20.55	7.77	7.19

recuperada $\hat{P}_0$. Cada experimento foi repetido 20 vezes e o erro considerado foi a média das distâncias obtidas.

Os ruídos usados foram gerados com base nas distribuição normal e uniforme, com níveis variando entre 0 e 0.2, com incremento de 0.01. A quantidade de pontos de referência inicia com valor igual a dimensão dos dados mais 1 e testa o total de 100 valores, com incremento de 2. Os dados considerados pertencem a $\mathbb{R}^5$, $\mathbb{R}^{10}$, $\mathbb{R}^{15}$ e $\mathbb{R}^{20}$.

Os resultados aparecem na Figura 28 com dados de dimensão 5, Figura 29 em dimensão 10, Figura 30 em dimensão 15 e Figura 31 em dimensão 20. Em todas as imagens o eixo x indica o nível de ruído, o eixo y o número de pontos de referência e a cor indica o erro de estimação médio obtido.

Alguns resultados esperados foram observados nos experimentos. Por exemplo, podemos observar, em qualquer linha, de qualquer tabela, que o erro tende a aumentar quando o

Figura 23 – Erro quadrático médio obtido no conjunto de dados COM

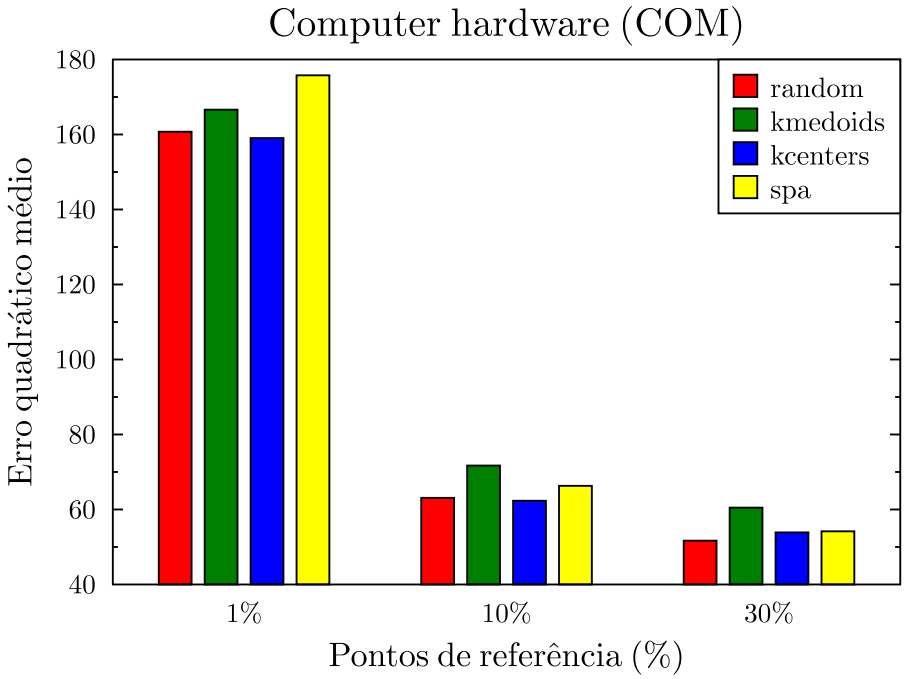
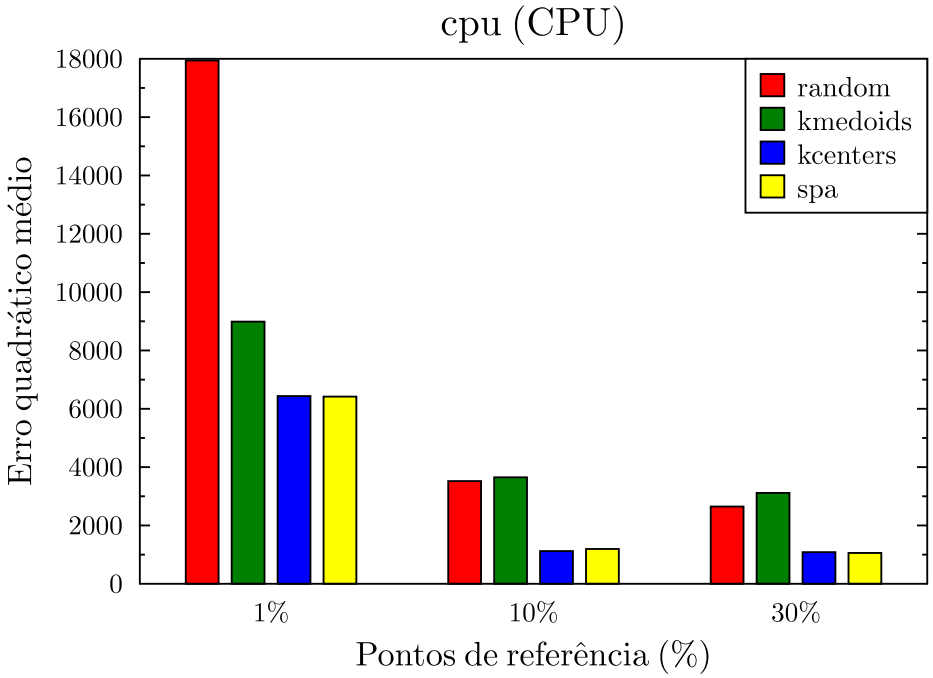


Figura 24 – Erro quadrático médio obtido no conjunto de dados CPU



nível de ruido aumenta. Além disso, se olharmos para as colunas (qualquer coluna de qualquer tabela), notamos que o erro reduz a medida que aumentamos o número de pontos de referência.

Figura 25 – Erro quadrático médio obtido no conjunto de dados FCB

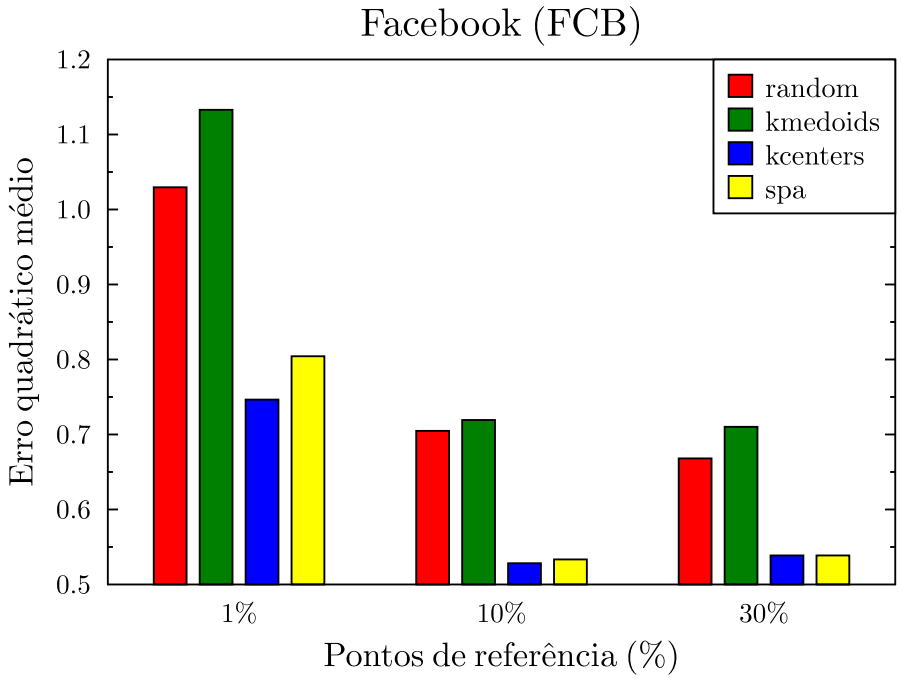
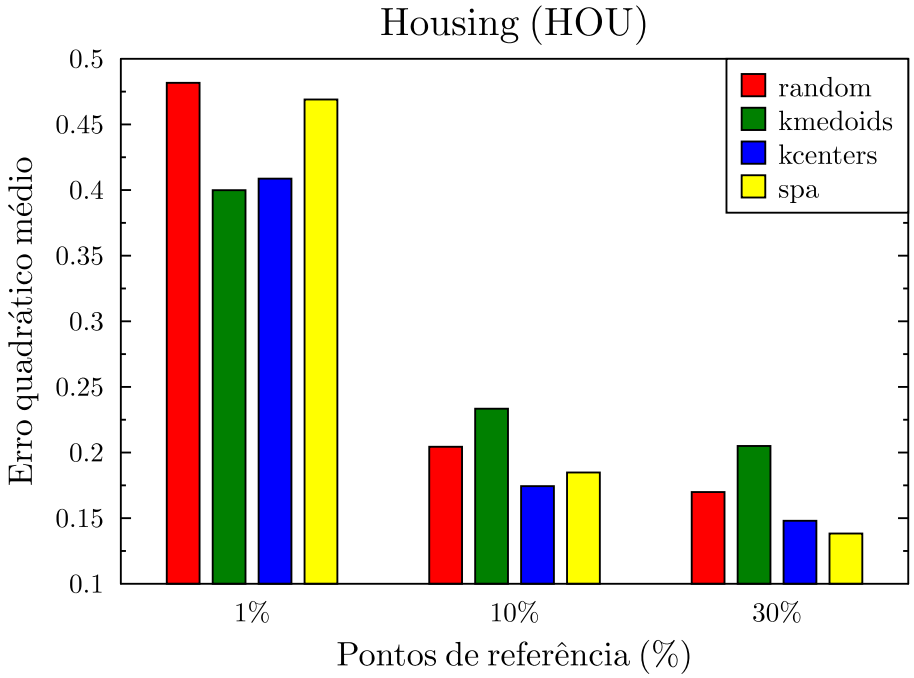


Figura 26 – Erro quadrático médio obtido no conjunto de dados HOU



Um outro resultado percebido foi que o método linear e o método baseado em otimização obtém erro médio zero quando os dados não estão corrompidos com ruído, e esse era

Figura 27 – Erro quadrático médio obtido no conjunto de dados MPG

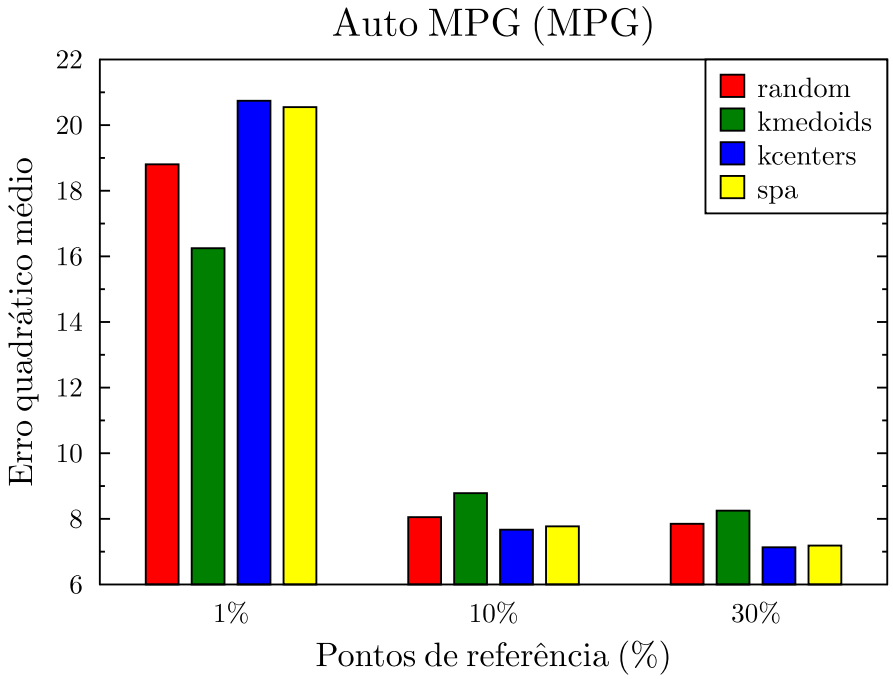


Figura 28 – Erro obtido em dados do $\mathbb{R}^5$

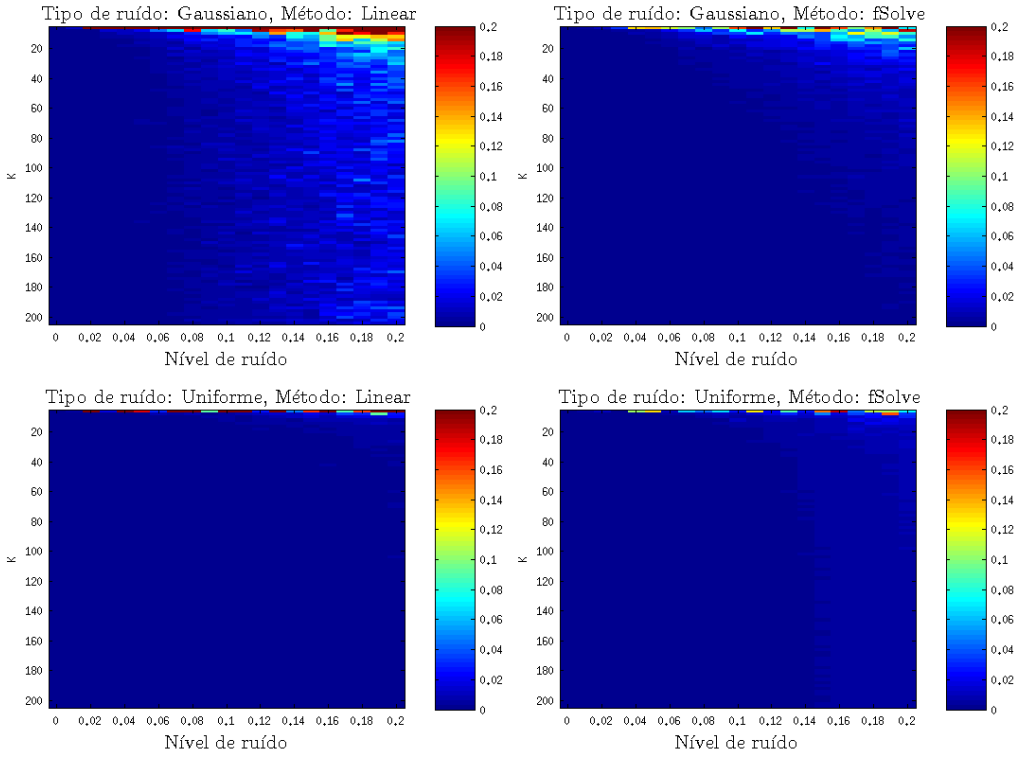
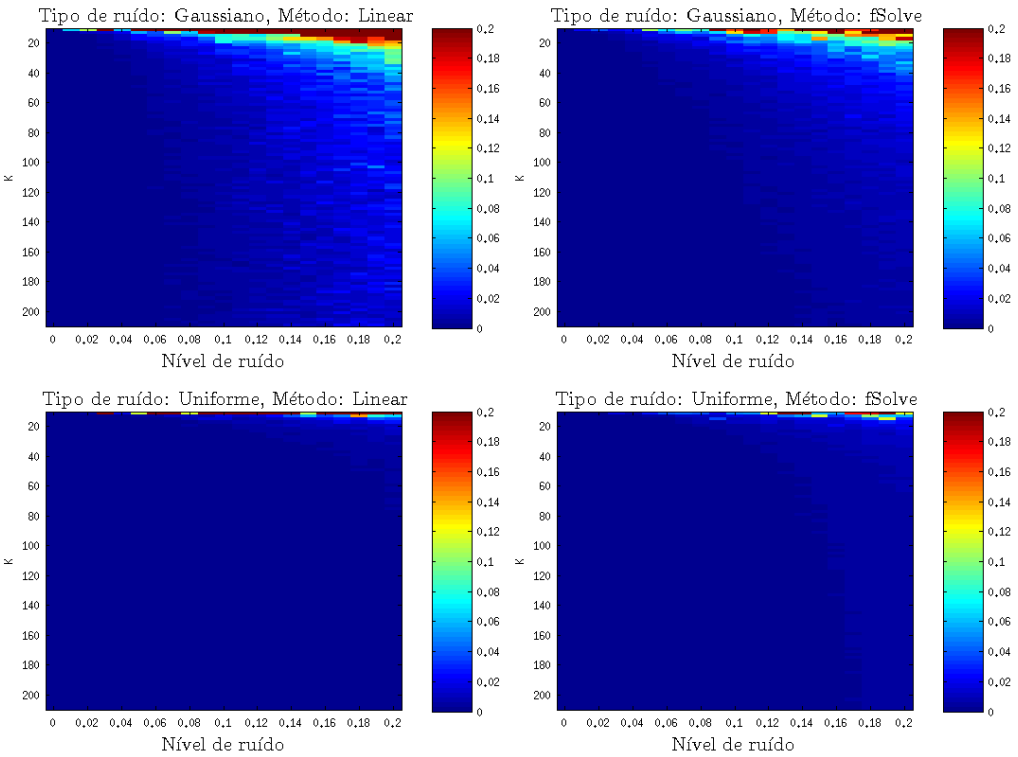


Figura 29 – Erro obtido em dados do $\mathbb{R}^{10}$



o resultado que procurávamos. Além disso, notamos também que o uso de otimização não linear torna o método mais robusto, pois o aumento no nível do ruído exerce uma influencia menor no resultado obtido.

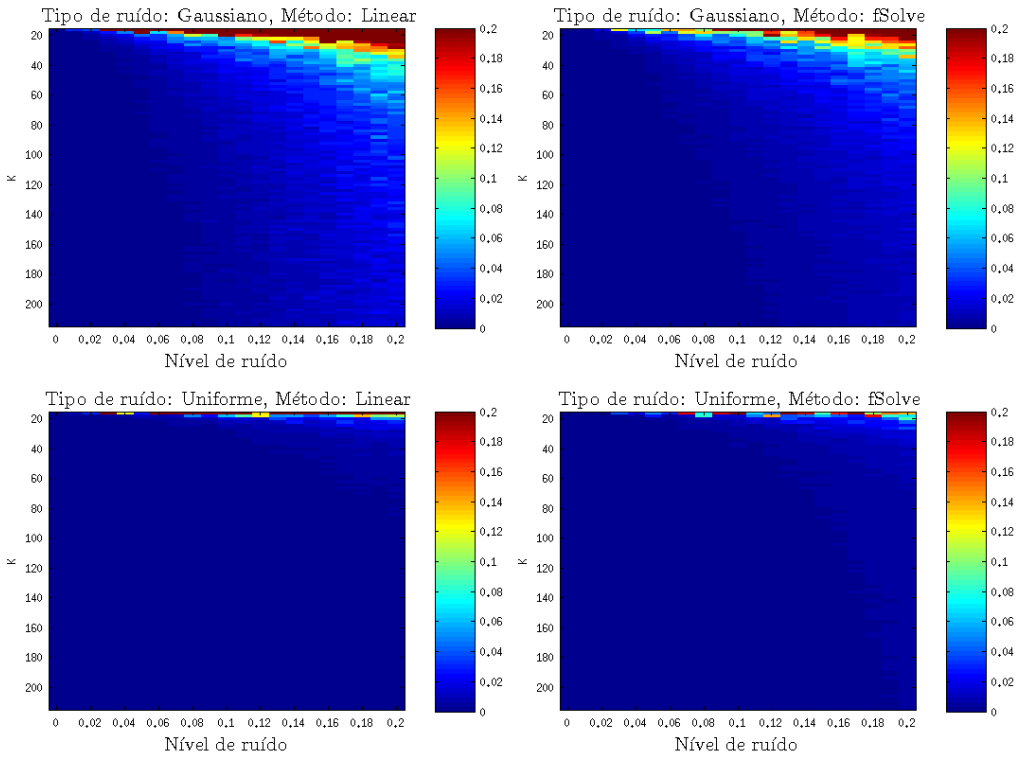
4.4 Conclusão

Nesta seção avaliamos experimentalmente alguns aspectos da MLM. Tais experimentos atacaram dois aspectos do algoritmo: a seleção dos pontos de referência, durante o treinamento, e a estimação da saída, na etapa de teste.

Em relação ao método de seleção dos pontos de referência, notamos que a quantidade de pontos de referência exerce maior influência no resultado que o método de seleção. Além disso, percebemos também que o método de seleção vai se tornando menos relevante a medida que a quantidade de pontos de referência aumenta. Já quando se tem poucos pontos de referência, como nos casos em que a base de dados não é muito numerosa, a influencia do método de seleção pode ser bem mais significativa.

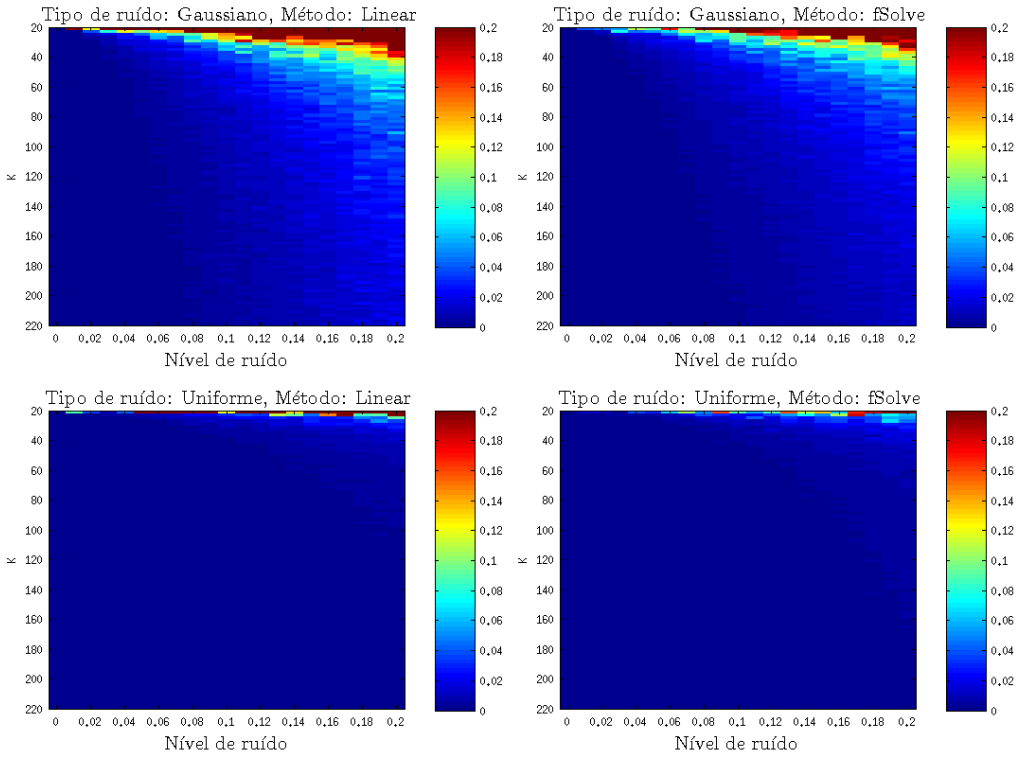
Quanto aos métodos de estimação da saída, buscamos avaliar o comportamento do

Figura 30 – Erro obtido em dados do $\mathbb{R}^{15}$



método usado na proposta original e do método linear que tratamos no Capítulo 3 quando os dados apresentam ruído. Observamos que com o aumento do número de pontos de referência, a influencia negativa do ruído tende a diminuir, nos permitindo concluir que a multilateração confere certa robustez ao modelo final.

Figura 31 – Erro obtido em dados do $\mathbb{R}^{20}$



5 MLM *TOOLBOX* PARA MATLAB

Como vimos nos capítulos anteriores, a máquina de aprendizado mínimo é uma técnica robusta e com desempenho compatível com a maioria dos algoritmos do estado da arte. Sua aplicação possui garantias teóricas sólidas e uma gama de variações documentadas na literatura.

A MLM possui algumas variações importantes, como as diferentes formas de estimar a saída, principalmente em relação aos diferentes tipos de tarefas, bem como no treinamento onde pode-se usar diferentes formas de selecionar os pontos de referência. Além disso, outra variação importante é em relação ao uso de comitês de MLMs, onde toda a estratégia de geração e integração possui variantes e uma implementação particular.

Nesse contexto, torna-se interessante o desenvolvimento de uma *toolbox* para Matlab que disponibilize essas funcionalidades de forma mais simples, abstraindo para o usuário todos os detalhes inerentes a implementação.

5.1 Implementação

A *toolbox* de MLM para Matlab encontra-se disponível em <goo.gl/Hth2j0> e pode ser usada e modificada livremente, nos termos da licença BSD. Ela foi desenvolvida buscando uma boa usabilidade, legibilidade e com código expansível.

Nossa *toolbox* foi feita para permitir o uso da MLM convencional ou do comitê de MLMs. Em ambos os casos, ela funciona tanto para classificação quanto para regressão. Ela também possui ferramentas para determinar o valor do hiper-parâmetro k através de validação cruzada e traz uma biblioteca de utilitários para manipulação de dados e medição de acurácia.

5.2 O código

Nossa *toolbox* foi desenvolvida buscando tornar o uso da MLM acessível à todos os usuários. A forma mais simples usa uma série de parâmetros pré-definidos, isentando o usuário da obrigação de selecionar qualquer hiper-parâmetro do método. Várias opções, porém, estão disponíveis para serem manipuladas pelo usuário de forma simples e organizada. Veremos nas seções seguintes uma descrição de cada um dos 9 arquivos presentes na *toolbox*.

5.2.1 *modelSelection.m*

Este arquivo serve para selecionar o hiper-parâmetro k através de validação cruzada. Seu cabeçalho é dado por `function [Kopt, error] = modelSelection(data, param, nFolds, method, lambda)`, onde os parâmetros de entrada e saída são descritos na tabela 7.

Tabela 7 – Parâmetros do método de seleção de modelo

Saída	
k_{opt}	Quantidade de pontos de referência que obteve menor valor de erro quadrático médio
error	Valor de erro quadrático médio obtido na validação cruzada com $k = k_{opt}$
Entrada	
data	Conjunto de dados de treinamento. Estrutura contendo data.x e data.y, respectivamente para input e output
param	Vetor contendo os valores de k que serão testados
nFolds (opcional)	Quantidade de pastas da validação cruzada. Valor padrão é 10
method (opcional)	Método de seleção dos pontos de referência. Valor padrão é "random", ou "aleatório"
lambda (opcional)	Valor do coeficiente de regularização. Valor padrão é zero.

Este é um arquivo que, a priori, o usuário não precisaria editar. É uma ferramenta que poderia ser usada ou ignorada, mas não deve haver manipulação alguma a ser feita em seu código.

5.2.2 *selectReferencePoints.m*

Este arquivo contém os métodos de seleção de pontos de referência implementados. Todos os métodos descritos nesta pesquisa, especialmente no Capítulo 4, já estão implementados. O usuário pode, entretanto, implementar novas estratégias se considerar pertinente a sua pesquisa. Um trecho da implementação pode ser vista a seguir:

```
1    function [refPoints] = selectReferencePoints(data, K, method)
2        if(strcmpi(method, 'random')),
3            ind = randperm(size(data.x,1));
4            refPoints.x = data.x(ind(1:K), :);
5            refPoints.y = data.y(ind(1:K), :);
6        end
```

Esta função, usada para selecionar os pontos de referência, normalmente não será chamada pelo usuário, mas sim pelos métodos da própria *toolbox*, cabendo ao usuário apenas editá-la para acrescentar novos métodos de seleção ou ignora-la.

Caso queira adicionar novos métodos, pode-se seguir o modelo dos métodos já implementados. Note-se que deve ser respeitado o formato da variável de saída `refPoints`, que deve ser uma estrutura onde `refPoints.x` deve conter as coordenadas dos pontos de referência no espaço da entrada e `refPoints.y` deve conter as coordenadas no espaço da saída. Os atributos de entrada são `data`, que contém o conjunto de dados, K, que indica a quantidade de pontos de referência e `method`, que deve ser usado no filtro onde se escolhe qual o método de seleção que deve ser empregado.

5.2.3 outputEstimation.m

Este arquivo contém os métodos de estimação da saída implementados. Todos os métodos descritos nesta pesquisa, especialmente nos Capítulos 2 e 3, já estão implementados. O usuário pode, entretanto, implementar novas estratégias se considerar pertinente a sua pesquisa. Um trecho da implementação pode ser vista a seguir:

```
1   function [yhat] = outputEstimation(Dyh, y, method)
2       N = size(Dyh, 1);
3       S = size(y, 2);
4       K = size(y, 1);
5       yhat = zeros(N, S);
6
7       if(strcmpi(method, 'fsolve'))
8           options_fsolve = optimset('Display', 'off', 'Algorithm','
                levenberg-marquardt', 'FunValCheck', 'on', 'TolFun', 10e-6
                );
9           yh0 = mean(y); % initial estimate for y
10          for i = 1: N,
11              yhat(i, :) = fsolve(@(x)(sum((y - repmat(x, K, 1)).^2, 2)
                    - Dyh(i,:)'.^2), yh0, options_fsolve);
12          end
13      end
```

Esta função, usada para estimar a saída, normalmente não será chamada pelo usuário, mas sim pelos métodos da própria *toolbox*, cabendo ao usuário apenas editá-la para acrescentar novos métodos ou ignora-la.

Caso queira adicionar novos métodos, pode-se seguir o modelo dos métodos já implementados. Note-se que deve ser respeitado o formato da variável de saída `yhat`, que deve ser uma matriz com N linhas e S colunas, onde $\hat{y}_{i,j}$ deve conter a estimativa do j-ésimo da i-ésima amostra no espaço da saída. Os atributos de entrada são `Dyh`, que contém a estimação das distâncias aos pontos de referência, y, que indica as posições dos pontos de referência no espaço da saída e `method`, que deve ser usado no filtro onde se escolhe qual o método de estimação da saída que deve ser empregado. Aqui cabe destacar que caso um novo método possua alguma restrição de uso, como o caso da equação do terceiro grau, que só pode ser aplicado a dados unidimensionais, é uma boa prática deixar isso explícito no código.

5.2.4 MLMUtil.m

Este arquivo contém uma classe contendo apenas alguns métodos estáticos úteis em experimentos de aprendizado de máquinas, podendo inclusive ser usados com outros algorítimos. O usuário normalmente não ira editá-lo, mas provavelmente fará uso de seus métodos. A seguir vemos a estrutura da classe, com os cabeçalhos dos métodos e em seguida entenderemos o que cada método faz.

```
1   classdef MLMUtil
2       methods(Static)
3           function [encoded_y, labels] = outputEncoding(y, labels)
4               %implementacao
5           end
6
7           function [decoded_y] = outputDecoding(y, labels)
8               %implementacao
9           end
10
11          function [accuracy] = getAccuracy(t, yhat)
12              %implementacao
13          end
14
```

```
15              function [mse] = getMSE(t, yhat)
16                  %implementacao
17              end
18          end
19      end
```

5.2.4.1 *outputEncoding*

Este método deve ser usado para converter o formato da saída para o padrão normalmente empregado por algorítimos de classificação. Um exemplo seria um conjunto de dados contendo três classes, onde o vetor y fosse preenchido com os valores 1, 2 ou 3, de acordo com a classe de cada amostra. O método `outputEncoding` iria converter esse vetor em uma matriz contendo N linhas, onde N é o tamanho do vetor, e 3 colunas, já que no nosso exemplo temos 3 classes. Na matriz resultante `encoded_y` teríamos o vetor $[1,0,0]$ onde antes havia 1, $[0,1,0]$ onde havia 2 e $[0,0,1]$ onde costumava haver 3. O output `labels` traria os rótulos originais, que poderiam ser usados, por exemplo, para reverter a codificação, gerando uma saída no mesmo formato dos dados originais.

5.2.4.2 *outputDecoding*

Função análoga a `outputEncoding`, porém aqui nós convertemos os vetores em valores indicados no atributo `labels`. Caso esse atributo não seja preenchido será considerado os números de 1 a S, onde S é o número de classes.

5.2.4.3 *getAccuracy*

Calcula a acurácia a partir dos valores alvo e predição, dados respectivamente pelos atributos `t` e `yhat`.

5.2.4.4 *getMSE*

Calcula o erro quadrático médio a partir dos valores alvo e predição, dados respectivamente pelos atributos `t` e `yhat`.

5.2.5 train.m

Aqui encontramos a função de treinamento da MLM. Esse método deverá ser bastante chamado pelo usuário que, a priori, não deverá editá-lo. O cabeçalho do método é dado por `function [model] = train(data, K, method, lambda)`. A descrição dos parâmetros de entrada e saída pode ser vista na Tabela 8.

Tabela 8 – Parâmetros de treinamento da MLM

	Saída
model	Modelo da rede MLM treinada.
	Entrada
data	Conjunto de dados de treinamento. Estrutura contendo data.x e data.y, respectivamente para input e output
K (opcional)	Números de pontos de referência. Valor padrão é $\frac{N}{2}$.
method (opcional)	Método de seleção dos pontos de referência. Valor padrão é "random", ou "aleatório"
lambda (opcional)	Valor do coeficiente de regularização. Valor padrão é zero.

5.2.6 predict.m

Este arquivo contém o método usado no teste de uma rede MLM previamente treinada. O cabeçalho do método é dado por `function [yhat, error] = predict(model, data, method)`. A descrição dos parâmetros de entrada e saída pode ser vista na tabela 9.

Tabela 9 – Parâmetros de teste da MLM

	Saída
yhat	Modelo da rede MLM treinada.
error	Erro da estimação (acurácia, para problemas de classificação, e mse, para problemas de regressão)
	Entrada
model	Modelo da rede MLM treinada.
data	Conjunto de dados de treinamento. Estrutura contendo data.x e data.y, respectivamente para input e output
method (opcional)	Método de estimação da saída. Valor padrão: fsolve.

5.2.7 ensembleGeneration.m

Este método é usado para o treinamento de um comitê de MLMs. Esse método deverá ser bastante chamado pelo usuário que, a priori, não deverá editá-lo. O cabeçalho do método é dado por `function [ ensemble ] = ensembleGeneration( data, M, bagging, r, nFolds )`. A descrição dos parâmetros de entrada e saída pode ser vista na tabela 10.

Tabela 10 – Parâmetros de treinamento do comitê de MLMs

	Saída
ensemble	Comitê de MLMs treinado de acordo com o parâmetros informados
	Entrada
data	Conjunto de dados de treinamento. Estrutura contendo data.x e data.y, respectivamente para input e output
M	Quantidade de MLMs ultilizadas
bagging	Flag que indica se as MLMs serão treinadas com conjuntos de dados diferentes
r	Fração dos dados que será dada para cada conjunto de treinamento
nFolds	Quantidade de folds utilizada na validação cruzada

5.2.8 ensembleIntegration.m

Este arquivo contém o método usado no teste de um comitê de MLMs previamente treinado. O cabeçalho do método é dado por `function [ yhat, error ] = ensambleIntegration ( ensembleModels, data, task, method)`. A descrição dos parâmetros de entrada e saída pode ser vista na tabela 11.

Tabela 11 – Parâmetros de teste de um comitê de MLMs

	Saída
yhat	Modelo da rede MLM treinada.
error	Erro da estimação (acurácia, para problemas de classificação, e mse, para problemas de regressão)
	Entrada
ensembleModel	Comitê de MLMs treinado.
data	Conjunto de dados de treinamento. Estrutura contendo data.x e data.y, respectivamente para input e output
task	Tipo de tarefa (classificação ou regressão)
method (opcional)	Método de estimação da saída. Valor padrão: fsolve.

5.3 Conclusão

A implementação que disponibilizamos, trás as principais funcionalidades da MLM de forma legível e expansível. Assim cobrindo tanto usuários que pretendem aplicá-la na solução de problemas quanto aos que pretendem se dedicar ao estudo da própria MLM, implementando novas alternativas para alguma das etapas do algoritmo. Além disso, trouxemos também uma biblioteca de funções úteis aos trabalhos de aprendizado de máquina de maneira geral. Essa biblioteca, na forma de uma classe composta por métodos estáticos, pode facilmente ser usada, não só pela MLM, mas também por outros algoritmos, facilitando o pré-processamento e a análise dos resultados em tarefas de reconhecimento de padrões e aproximação de funções.

6 CONCLUSÃO

Neste trabalho abordamos alguns aspectos importantes da máquina de aprendizagem mínima. Nele buscamos explorar tópicos ainda carentes de uma análise mais completa em relação ao algoritmo.

Nossa primeira contribuição foi estudar a capacidade de interpolação da MLM, mostrando em que condições o modelo pode ajustar-se perfeitamente aos dados de treinamento, produzindo, portanto, erro zero nos dados apresentados. Este comportamento, embora não seja exatamente o que se busca em um algoritmo de aprendizado de máquina, é um resultado importante, primeiro por mostrar que o modelo não é excessivamente simples, o que o tornaria incapaz de capturar a dinâmica de dados mais complexos, e segundo por ser um importante passo na direção de estudar a capacidade de aproximação universal da MLM. Essa tarefa, aliás, é a principal recomendação de trabalho futuro decorrente desta dissertação.

Além disso, também realizamos um estudo com alguns métodos alternativos para selecionar os pontos de referência, realizando um estudo comparativo dos resultados de cada um e o impacto da variação do número de pontos de referência K. Aqui conseguimos notar, experimentalmente, que o método de seleção influencia significativamente o resultado para K relativamente pequeno, mas essa influencia diminui com o aumento da quantidade de pontos de referência. Um trabalho futuro em relação a isso é o estudo de novas técnicas de seleção de pontos de referência, especialmente com o uso de métodos supervisionados. Outra técnica que pode ser avaliada com mais cuidado é o uso de pontos de referência diferentes no espaço da entrada e da saída. Neste ponto, pode-se, por exemplo, usar menos pontos de referência na saída ou tentar selecionar pontos afim-independentes, o que seria útil a tarefa de estimação da saída, como vimos no Capítulo 3.

Em relação a estimação da saída, verificamos que a multilateração apresenta uma certa tolerância ao ruído, de modo que a qualidade da estimativa da posição do ponto deixa de ser influenciada significativamente pelo ruído da estimação das distâncias a medida que se aumenta a quantidade de pontos de referência. Esse comportamento se repete tanto para o método da proposta original (otimização não linear) quanto a que discutimos neste trabalho, e tanto para o ruído gaussiano quanto uniforme.

Por fim, produzimos uma *toolbox* para Matlab com as principais funcionalidades da MLM. Nela trazemos as principais funções propostas na literatura, dispostos de forma legível e expansível, tornando seu uso indicado tanto aos que desejam apenas usar o modelo na solução

de um problema específico, quanto aos que pretendem modificar o algoritmo ou estudar seu comportamento. Além disso, trazemos também uma biblioteca de funções para facilitar o pré-processamento dos dados e a análise dos resultados que pode ser usada com qualquer algoritmo de aprendizado de máquina.

REFERÊNCIAS

ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. **Learning From Data**. [S.l.]: AMLBook, 2012. ISBN 1600490069, 9781600490064.

ALENCAR, A. S. C.; CALDAS, W. L.; GOMES, J. P. P.; SOUZA, A. H. d.; AGUILAR, P. A. C.; RODRIGUES, C.; FRANCO, W.; CASTRO, M. F. d.; ANDRADE, R. M. C. Mlm-rank: A ranking algorithm based on the minimal learning machine. In: **2015 Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2015. p. 305–309.

ARAúJO, M. C. U.; SALDANHA, T. C. B.; GALVãO, R. K. H.; YONEYAMA, T.; CHAME, H. C.; VISANI, V. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. **Chemometrics and Intelligent Laboratory Systems**, v. 57, n. 2, p. 65 – 73, 2001. ISSN 0169-7439. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169743901001198>.

AUER, J. W. An elementary proof of the invertibility of Distance matrices. **Linear and Multilinear Algebra**, v. 40, n. 2, p. 119–124, 1995. Disponível em: <http://dx.doi.org/10.1080/03081089508818427>.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006.

BRAGA, A. de P. **Redes neurais artificiais: teoria e aplicações**. LTC Editora, 2007. ISBN 9788521615644. Disponível em: <https://books.google.com.br/books?id=R-p1GwAACAAJ>.

CALDAS, W. L.; GOMES, J. P. P.; CACAIS, M. G.; MESQUITA, D. P. P. Co-mlm: A ssl algorithm based on the minimal learning machine. In: **2016 5th Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2016. p. 97–102.

CAO, D.; YANG, B. An improved k-medoids clustering algorithm. In: **Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on**. [S.l.: s.n.], 2010. v. 3, p. 132–135.

de Souza Junior, A.; CORONA, F.; MICHé, Y.; LENDASSE, A.; BARRETO, G. A. Extending the minimal learning machine for pattern classification. In: **2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence**. [S.l.: s.n.], 2013. p. 236–241. ISSN 2377-0589.

de Souza Junior, A. H.; CORONA, F.; MICHE, Y.; LENDASSE, A.; BARRETO, G. A.; SIMULA, O. Minimal Learning Machine: A New Distance-Based Method for Supervised Learning. In: ____. **Advances in Computational Intelligence: 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 408–416. ISBN 978-3-642-38679-4. Disponível em: <http://dx.doi.org/10.1007/978-3-642-38679-4{_}.>

GARCIA-PEDRAJAS, N.; HERVAS-MARTINEZ, C.; ORTIZ-BOYER, D. Cooperative coevolution of artificial neural network ensembles for pattern classification. **Evolutionary Computation, IEEE Transactions on**, v. 9, n. 3, p. 271–302, 2005. ISSN 1089-778X.

GOMES, J.; MESQUITA, D.; FREIRE, A.; JUNIOR, A. S.; KARKKAINEN, T. A robust minimal learning machine based on the m-estimator. **European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN**, 2017.

GOMES, J. P. P.; SOUZA, A. H.; CORONA, F.; NETO, A. R. R. A cost sensitive minimal learning machine for pattern classification. In: ____. **Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I**. Cham: Springer International Publishing, 2015. p. 557–564. ISBN 978-3-319-26532-2. Disponível em: <http://dx.doi.org/10.1007/978-3-319-26532-2_61>.

GONZALEZ, F. Clustering to minimize intercluster distance*. v. 38, p. 293–306, 1985.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HUANG, G.-B.; ZHOU, H.; DING, X.; ZHANG, R. Extreme learning machine for regression and multiclass classification. **IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics**, v. 42, n. 2, p. 513–29, 2012. ISSN 1941-0492. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/21984515>.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

MALKOMES, G.; KUSNER, M. J.; CHEN, W.; WEINBERGER, K. Q.; MOSELEY, B. Fast distributed k-center clustering with outliers on massive data. Curran Associates, Inc., p. 1063–1071, 2015. Disponível em: <http://papers.nips.cc/paper/5997-fast-distributed-k-center-clustering-with-outliers-on-massive-data.pdf>.

MESQUITA, D. P. P.; GOMES, J. P. P.; Souza Junior, A. H. Ensemble of Efficient Minimal Learning Machines for Classification and Regression. **Neural Processing Letters**, Springer US, 2017. ISSN 1370-4621. Disponível em: <http://link.springer.com/10.1007/s11063-017-9587-5>.

MICCHELLI, C. A. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. **Constructive Approximation**, v. 2, n. 1, p. 11–22, 1986. ISSN 01764276.

OLIVEIRA, A. C. d.; GOMES, J. P. P.; NETO, A. R. R.; SOUZA, A. H. d. Efficient minimal learning machines with reject option. In: **2016 5th Brazilian Conference on Intelligent Systems (BRACIS)**. [S.l.: s.n.], 2016. p. 397–402.

OLIVEIRA, S. A. F.; SOUZA, A. H. d. Reference Points Selection based on Corner Detection for Minimal Learning Machines. 2017.

VINOD, H. D. Integer programming and the theory of grouping. **Journal of the American Statistical Association**, [American Statistical Association, Taylor Francis, Ltd.], v. 64, n. 326, p. 506–519, 1969. ISSN 01621459. Disponível em: <http://www.jstor.org/stable/2283635>.