Renata Gomes Cordeiro

Um método de classificação de dados utilizando programação linear

Campos dos Goytacazes/RJ

2013

Renata Gomes Cordeiro

Um método de classificação de dados utilizando programação linear

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para obtenção do título de Bacharel em Ciência da Computação, sob orientação do Profº. Fermín Alfredo Tang Montané.

Tutor: Fermín Alfredo Tang Montané.

Universidade Estadual do Norte Fluminense Darcy Ribeiro

Campos dos Goytacazes/RJ

2013

"Para tudo há um tempo, para cada coisa há um momento debaixo do céu."

Ecl 3,1

Agradecimentos

À Deus por guiar os meus passos e permitir o aconteciemnto desse momento.

À minha família, pais, irmãs, cunhados, sobrinhos, avós, a todos, por todo apoio e felicidade.

Aos meus pais, Nerivaldo e Selma, por entenderem a importância da educaçãoe serem a base para as minhas conquistas.

Às minhas irmãs Cintia e Aline por serem presentes e fundamentais em mais essa conquista. Em especial à Aline, por acompanhar de perto e contribuir em cada etapa dessa caminhada e por ser minha inspiração.

Ao meu cunhado Adelson, pelos conselhos e por ser também fonte de inspiração para mim.

À Munir, por ser tão companheiro e me transmitir calma nos momentos de tensão.

Ao meu orientador, o professor Fermín Alfredo Tang Montané, pelo incentivo para a conclusão desse trabalho. E pela oportunidade se der sua bolsista de iniciação científica, etapa que deu início a essa trabalho.

A todos os professores, pelos ensinamentos passados durante o curso.

Aos meus amigos, por torcerem por mim.

Aos CNPq, pelo apoio financeiro para a realização deste trabalho.

Resumo

A programação linear possui aplicações em diversas áreas, o objetivo deste trabalho é explorar uma aplicação na computação. Um modelo de programação linear é utilizado no processo de classificação de dados, que consiste em: para um conjunto de dados com p padrões conhecidos, dado um vetor desse conjunto é possível classificá-lo em um dos p padrões. O método de classificação possui a etapa de treinamento e teste. Na etapa de treinamento, para um par de padrões representados por vetores, que representam pontos no espaço n-dimensional, um modelo de programação linear gera um hiperplano que separa os conjuntos de pontos dos dois padrões. Na etapa de teste, utilizando a estrutura de uma árvore binária de torneio e com base nos hiperplanos separadores um vetor é classificado em dos padrões separados pelos hiperplanos. Foram realizados experimentos com quatro conjuntos de dados: dígitos escritos manualmente, gestos de língua brasileira de sinais, espécies da planta Iris e expressões faciais. Obtendo taxas de acerto de classificação variadas para os quatro conjuntos de dados. Através da classificação de dados é possível realizar atividades de reconhecimento como por exemplo, reconhecer uma expressão facial através de uma imagem.

Palavras-chave: Programação linear, classificação de dados, método de classificação.

Lista de Figuras

Lista de Tabelas

Sumário

Lista de siglas

SVM	*Support Vector Machines*
MPLHS	Programação Linear de Hiperplano Separador
LIBRAS	Língua Brasileira de Sinais

1 Introdução

A programação linear é uma das disciplinas que compõem a programação matemática e constitui um dos pilares da pesquisa operacional. As aplicações da programação linear estão presentes em diversos setores, tais como nas indústrias, nos transportes, na saúde, na educação, na computação, etc. Mas é mais comumente aplicada na engenharia de produção, em problemas que buscam a distribuição eficiente de recursos, minimização de gastos e maximização do lucro. O presente trabalho está focado na utilização da programação linear na computação, mais especificamente no processo de classificação de dados em padrões.

Um dos focos dos estudos das aplicações da programação linear é na sua utilização na separação de pontos no espaço n-dimensional. Na computação esse tipo de aplicação pode ser utilizada em atividades com o objetivo de classificar dados, como por exemplo:

- Dígitos escritos manualmente
- Expressões faciais
- Espécies de plantas
- Gestos manuais

De forma geral, esse tipo de aplicação da programação linear pode ser utilizado em abordagens que permitam a representação dos dados em forma de vetor numérico. Esse vetor de tamanho n representa um ponto no espaço n-dimensional

e cada valor do vetor representa uma característica, por isso é denominado vetor de características.

No processo de classificação de dados, os padrões são as possíveis classificações que um dado pode receber.

No presente trabalho é utilizada a metodologia para classificação de dados apresentada em Hadid (2005) e Dyer (2005). Primeiramente, o modelo de programação linear (Programação Linear de Hiperplano Separador (MPLHS)) proposto por Bennett e Mangasarian (1992) é utilizado na geração de hiperplanos, cada hiperplano separa dois conjuntos de pontos, onde cada conjunto representa um padrão. Na etapa de classificação de um dado, com padrão inicialmente desconhecido, uma árvore binária de torneio é utilizada na determinação de qual padrão o dado pertence com base nos hiperplanos gerados.

O MPLHS dois conjuntos de pontos, porém também pode ser utilizado em problemas onde busca-se a separação de múltiplos padrões. O presente trabalho foca nessa última abordagem, em todos os testes realizados o número de conjuntos de pontos (padrões) é igual ou maior que três.

O método simplex proposto por Danzitg (1963) é utilizado para a resolução do MPLHS. É um dos métodos mais conhecidos e eficientes para resolver problemas de programação linear. Foi implantado comercialmente há mais de 40 anos e atualmente está presente em softwares comerciais tais como CPLEX e LINGO.

1.1 Objetivos e justificativas

O objetivo do presente trabalho é o estudo da utilização da programação linear e do método simplex, na classificação de dados em padrões. O MPLHS gera hiperplanos, cada hiperplano separa dois padrões representados por dois conjuntos de pontos e uma árvore binária de torneio realiza a classificação de um dado com padrão inicialmente desconhecido.

Através desse estudo deve ser possível verificar a eficiência da programação linear nesse tipo de aplicação. São realizados testes com quatro conjuntos de dados para verificar a eficácia da metodologia utilizada que é composta pelo MPLHS na separação de padrões e uma árvore de torneio para classificação. Três desses conjuntos foram obtidos já na forma de vetores de características, o quarto conjunto foi obtido em forma de imagens e um método de extração de características foi utilizado para obter os vetores.

A programação linear possui aplicações em diversas áreas, como: indústria, produção, saúde e computação gráfica. Porém é um método mais comumente utilizado na engenharia de produção. O presente trabalho justifica-se pelo fato de abordar uma aplicação prática dentro da computação, onde a aplicação da programação linear não é tão explorada quanto na engenharia de produção.

1.2 Metodologia

Para o cumprimento do objetivo final, o trabalho é composto pelas seguintes etapas:

- O estudo do método simplex revisado, suas características e vantagens do ponto de vista computacional;
- Obtenção de dados para testes;
- A implementação do MPLHS utilizando a linguagem de programação JAVA juntamente com o software CPLEX;
- A implementação da etapa de classificação através da árvore binária de torneio;
- Realização de testes e análise dos resultados.

1.3 Estrutura do trabalho

Este trabalho está estruturado em seis capítulos. O primeiro capítulo apresenta a introdução ao tema abordado, os objetivos e justificativas e a metodologia descrevendo as etapas para o cumprimento do trabalho.

O segundo capítulo é composto pela descrição geral do problema de programação linear, algumas aplicações práticas, uma apresentação de alguns métodos de classificação e métodos de validação da metodologia, além de alguns trabalhos relacionados à classificação de dados.

O capítulo 3 apresenta de forma detalhada o Método Simplex e outro importante método de solução de problemas de programação linear.

No quarto capítulo é apresentado o MPLHS utilizado na geração dos hiperplanos separadores e o processo de classificação de dados.

O capítulo 5 apresentada os experimentos realizados, suas análises e os dados utilizados

O sexto e último capítulo trata das considerações finais, conclusões e propostas de continuação deste trabalho.

2 Referencial teórico

Este capítulo apresenta um revisão teórica sobre as principais metodologias utilizadas nesse trabalho: programação linear, métodos de classificação e métodos de validação. Também é apresentada uma revisão de trabalhos sobre classificação de dados.

2.1 A programação linear

Na pesquisa operacional, a programação linear é uma das técnicas mais utilizadas para resolver problemas de otimização. Os problemas de programação linear geralmente buscam a distribuição eficiente de recursos limitados para atender um determinado objetivo, por isso suas aplicações estão presentes em diversas áreas como computação, administração, indústria e transporte (PAMPLONA, 2005).

Um problema de programação linear é expresso através de um modelo composto por equações e inequações lineares. Esse tipo de problema busca a distribuição eficiente de recursos com restrições para alcançar um objetivo, em geral, maximizar lucros ou minimizar custos. Em um problema de programação linear esse objetivo é expresso através de uma equação linear denominada função objetivo. Para a formulação do problema, é necessário também definir os recursos necessários e em que proporção são requeridos. Essas informações são expressas em equações ou inequações lineares, uma para cada recurso. Esse conjunto de equações ou inequações é denominado restrições do modelo (PAMPLONA, 2005).

2.2 Descrição do problema de programação linear

O modelo de um problema de programação linear é apresentado em uma das formas a seguir: **Modelo 1**

$$Max\ z = c^T x \tag{2.1}$$

$$s.a. \begin{cases} Ax \leq b \\ x \geq 0 \end{cases} \tag{2.2}$$

Modelo 2

$$Min\ z = c^T x \tag{2.3}$$

$$s.a. \begin{cases} Ax \geq b \\ x \geq 0 \end{cases} \tag{2.4}$$

Nos Modelos 1 e 2 as equações 2.1 e 2.3 representam as funções objetivo e as inequações 2.2 e 2.4 representam as restrições. Onde,

- $\boldsymbol{x}$ é o vetor com as variáveis do modelo
- $\boldsymbol{c}$ é o vetor de coeficientes da função objetivo
- $\boldsymbol{z}$ é o valor da função objetivo
- $\boldsymbol{A}$ é a matriz com as constantes das restrições
- $\boldsymbol{b}$ é o vetor com os valores limites das restrições

Sendo que c, A e b são dados conhecidos.

A seguir é apresentado um exemplo de aplicação da programação linear no contexto da otimização da produção. Uma empresa, que fabrica vários produtos, deseja maximizar o lucro na venda de dois desses produtos. A produção é feita

em três setores diferentes e está sujeita a restrições de tempo disponível para a produção em cada setor. O problema pode ser representado pelo seguinte modelo de programação linear com duas variáveis(HILLIER; LIEBERMAN, 2006).

$$Maximize\ z = 3x_1 + 5x_2 \tag{2.5}$$

$$Sujeito\ a$$

$$1x_1 \leq 4 \tag{2.6}$$

$$2x_2 \leq 12 \tag{2.7}$$

$$3x_1 + 2x_2 \leq 18 \tag{2.8}$$

$$x_1 \geq 0, x_2 \geq 0 \tag{2.9}$$

Onde,

- $\boldsymbol{x_1}$ representa a quantidade do produto 1 produzido em uma semana.
- $\boldsymbol{x_2}$ representa a quantidade do produto 2 produzido em uma semana.
- $\boldsymbol{z}$ representa o lucro total por semana de produção desses dois produtos (em milhões de dólares), sendo o lucro do produto 1 de 3 milhões e o do produto 2 de 5 milhões.

E as restrições representam as restrições de tempo de cada setor que compõe o processo de produção.

- A equação 2.6 garante que, durante o processo de produção, cada produto 1 necessita de 1 hora no setor 1, e o setor só tem disponível 4 horas por semana.
- A equação 2.7 garante que, durante o processo de produção, cada produto 2 necessita de 2 horas no setor 2, e o setor só tem disponível 12 horas por semana.

- A equação 2.8 garante que, durante o processo de produção, cada produto 1 necessita de 3 horas no setor 3, e cada produto 2 necessita de 2 horas no setor 3, e o setor só tem disponível 18 horas por semana.
- A equação 2.9 garante que, os dois produtos devem ser produzidos.

Um problema de programação linear com até três variáveis pode ser representado graficamente utilizando três eixos cartesianos. Os problemas com duas variáveis podem ainda ser facilmente resolvidos por meio da representação gráfica (PASSOS, 2009).

A seguir, o problema exemplo é apresentado graficamente. Apesar de, na prática os modelos de programação linear possuírem um número de variáveis muito maior que dois ou três, a visualização gráfica do modelo, mesmo que simples, contribui para o entendimento dos métodos de resolução apresentados no capítulo a seguir.

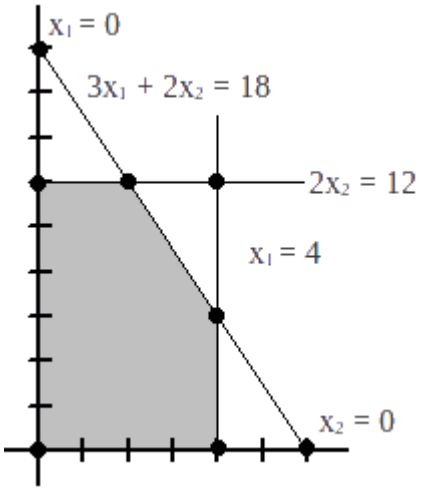


Figura 1: Representação gráfica do modelo de programação linear de duas variáveis

Onde cada reta representa uma restrição do modelo, e a área cinza representa a região viável, ou seja, nessa área estão contidas os valores viáveis de x_1 e x_2 para a maximização do lucro.

Os métodos para resolução de problemas de programação linear buscam esses valores de x_1 e x_2 para a determinação da solução ótima.

2.3 Aplicações utilizando programação linear

Um problema de programação linear, como já dito anteriormente, busca um resultado ótimo sujeito a restrições, com utilização em diversas áreas.

A programação linear se aplica na área da saúde, como demonstrado por Goldberg (2006) em seu trabalho. Em um determinado tipo de tratamento de câncer são inseridos cateteres na área afetada para introduzir o medicamento necessário, porém o medicamento acaba afetando células saudáveis além das cancerígenas. Como os problemas de programação linear podem ser resolvidos como problemas determinísticos e com solução exata, Goldberg (2006) propõe a formulação de um problema de otimização para determinar o tempo de permanência dos cateteres, minimizando os desvios em relação a quantidade da dose necessitada pelo paciente através da minimização de custos atribuídos. Nos testes realizados, foi obtida uma melhoria nos desvios em relação a quantidade da dose necessitada pelo paciente mas clinicamente os resultados obtidos não mostraram uma vantagem significativa em relação ao método atualmente utilizado.

Em Moreira (2003) modelos de programação linear foram desenvolvidos para duas aplicações relacionadas a área da saúde. Em um dos modelos busca-se a formulação de uma dieta com custo mínimo, considerando as restrições alimentares e os nutrientes essenciais em uma dieta. Um segundo modelo visa analisar intervenções médicas que buscam maximizar os anos de vida de pacientes considerando uma população, como restrições são utilizados os custos e o número de visitas médicas. Para a resolução dos modelos foi utilizado o método simplex. O autor considerou a programação linear como um instrumento útil na tomada de decisões na área da saúde, considerando a importância da substituição de métodos tradicionais baseados no bom senso e tentativa de acerto por métodos com soluções

otimizadas.

Nos trabalhos de Goldberg (2006) e Moreira (2003) foi exemplificado como a programação pode ser útil e importante na área da saúde. No primeiro trabalho apesar de não demonstrar uma melhoria no resultado geral, foi comprovada a equivalência dos resultados com o método probabilístico atualmente utilizado. Em Moreira (2003) apesar dos dados utilizados serem reduzidos e uma dieta real possuir mais restrições que as propostas no trabalho, o autor demonstrou a aplicabilidade da programação linear em duas situações da área da saúde.

Em Possamai e Pescador (2011) a programação linear foi utilizada na resolução de um problema abrangendo o transporte, o processamento e a estocagem de fumo, buscando como objetivo a minimização dos custos. O modelo proposto foi composto por 650 variáveis e 150 restrições. A partir do resultado obtido foi possível determinar parâmetros desde o transporte da matéria bruta, estocagem até a venda do produto.

Dentre as aplicações mais conhecidas da programação linear, encontram-se os problemas de planejamento de produção e controle de estoque, assuntos relacionados a engenharia de produção, como no trabalho de Possamai e Pescador (2011). A partir desse trabalho tem-se o exemplo de como os resultados extraídos de um modelo de programação linear podem ser determinantes no sucesso de um processo de produção.

Em seu trabalho Krukoski (2010) propõe a utilização da programação linear na economia. Considerando que os investimentos em ações estão cada vez mais acessíveis para pequenos investidores, o autor propõe um modelo de programação linear na determinação de uma operação de compra ou venda que maximize o lucro e baseando-se também nos riscos. Os resultados obtidos em sua maioria foram positivos e lucrativos, apesar de, de acordo com o autor, algumas operações obtidas nos resultados serem pouco aplicáveis se comparadas às atitudes rotineiras dos investidores.

No trabalho proposto por (KRUKOSKI, 2010) foi apresentada uma aplicação que pode auxiliar na tomada de decisões na área da economia. Os bons resultados mostram que a programação linear, como método determinístico, pode ser empregada mesmo em problemas que estão mais relacionados a métodos probabilísticos, como é o caso dos investimentos em ações.

A programação linear além de estar presente, é fundamental em diversas áreas, tornando-se uma ferramenta de apoio a decisão e contribuindo para o sucesso de projetos nas áreas em que se aplica.

2.4 Métodos de classificação

Em problemas de classificação, dado um conjunto de dados dividido entre conjunto de treinamento e conjunto de teste, onde, no conjunto de treinamento cada subconjunto pertence a um entre n padrões, o objetivo é que a partir de um dado do conjunto de teste o método de classificação retorne a qual dos n padrões esse dado pertence. A seguir são apresentados alguns dos métodos de classificação mais citados na literatura.

2.4.1 *Support Vector Machines*

Support Vector Machines (SVM) ou Máquinas de vetores de suporte têm a capacidade de gerar classificadores na etapa de treinamento e de classificar os dados na etapa de teste de acordo com os classificadores gerados. Considerando um problema de classificação com dois padrões, uma SVM irá determinar um plano que separe os pontos desses dois padrões, de forma que a distância entre o hiperplano e os pontos seja a máxima possível. Os pontos de cada padrão utilizados como referência são denominados vetores de suporte. No caso de conjuntos linearmente inseparáveis, é utilizada um função denominada Kernel. Essa função é responsável por elevar a dimensão espacial a fim de que os pontos se tornem linearmente separáveis. Portanto para a obtenção de um classificador utilizando SVM

é necessária a escolha de uma função kernel e os parâmetros dessa função. Essa escolha influencia diretamente no desempenho do classificador(GUNN, 1998)(LIMA, 2002).

No caso em que o número de padrões é maior que dois, duas abordagens podem ser utilizadas com as SVM (LORENA, 2003):

- **Um contra todos:** Nessa abordagem, para k padrões são geradas k SVM. Na criação das SVM, é gerado um hiperplano que separa 1 padrão dos k-1 padrões. Na determinação do padrão de um dado x, o padrão é aquele que, entre as k SVM obteve x do lado do hiperplano onde se encontrava o padrão separado dos demais.

- **Todos contra um:** Nessa abordagem os padrões são agrupados em pares e uma SVM é gerada para cada par. Um esquema de votação deve ser utilizado para determinar o padrão de uma instância, que deve ser analisada a partir de cada SVM e cada SVM retorna um padrão possível. O padrão que obtiver mais pontos da instância é a classe atribuída.

2.4.2 *Naive Bayes*

Na metodologia *Naive Bayes* as características do dado a ser classificado são analisadas de forma independente. O vetor de características é formado pela número de vezes que cada característica ocorre no dado caracterizado. A probabilidade de uma instância pertencer a um determinado padrão é dada por uma probabilidade inicial, baseada na quantidade total de dados, e pelas probabilidades das ocorrências das características (MCCALLUM; NIGAM, 1998)
(LANGLEY; THOMPSON, 1992).

2.4.3 *k*- nearest neighbor

Nesse método a classificação é feita pela similaridade da instância a ser classificada com um dado ou vetor utilizado no treinamento, a classe do dado utilizado no treinamento é então atribuída a instância com padrão inicialmente desconhecido. As etapas desse método consistem em utilizar um parâmetro para medir a semelhança ou a distância do dado a ser classificado em relação a cada um dos dados utilizados no treinamento. Os k vetores mais similares ou mais próximos são selecionados e entre o padrão mais frequente é atribuído ao dado inicialmente desconhecido. Apesar de ser considerado um método simples, sua dificuldade está em definir os parâmetros ou a métrica que definirá a semelhança ou distância entre os dados(SANTOS, 2009).

2.4.4 Redes Neurais Artificias

Em seu trabalho Morais (2010) define Redes Neurais Artificiais (RNA) como sistemas paralelos e distribuídos compostos por unidades de processamento simples (neurônios) interligadas por conexões, esses neurônios calculam determinadas funções matemáticas. Na fase de treinamento ou aprendizagem "um conjunto de exemplos é apresentado a rede que extrai automaticamente características necessárias para representar a informação fornecida"(MORAIS, 2010). Uma rede neural pode receber como entrada, na etapa de treinamento, um conjunto de vetores com seus respectivos padrões. Ao submeter como entrada um vetor com padrão desconhecido a rede neural deve classificar esse vetor. A vantagem desse método é que uma rede pode construir fronteiras não lineares entre padrões, o que pode ser muito vantajoso em problemas complexos de classificação. Existem vários modelos de redes neurais que possibilitam o reconhecimento de padrões, entre eles: Perceptron de Camada Simples, Perceptron de Múltiplas e Redes de Kohonen (ZUBEN; CASTRO, 2003).

2.5 Aplicações em classificação de dados

Em seu trabalho Lima (2002) busca o classificação de impressões digitais em 5 padrões. Esse tipo de classificação tem o propósito de gerenciar grandes bancos de dados de impressões digitais e acelerar o processo de identificação. Foi utilizado um banco contendo 4000 imagens igualmente distribuídas entre os cinco padrões. O autor dividiu o banco em dois grupos, utilizando um para treinamento e outro para teste. Após a extração das características das imagens, Máquinas de Vetores de Suporte foram utilizadas na classificação. Na abordagem todos contra um foi obtida uma média de precisão de 91,18%, já na abordagem um contra todos a precisão na classificação das impressões foi de 90,43%.

Santos (2009) apresentou uma técnica para a classificação de textos, uma atividade importante em aplicações como bibliotecas digitais e em outras que em geral buscam a organização de documentos disponíveis no formato digital. O autor propôs variações no método *k near neighbor* a fim de aprimorar a escolha dos textos, utilizados no treinamento, mais próximos ao texto que precisa ser classificado. Nos testes feitos com três coleções de textos, as médias de acertos na classificação obtidas foram acima de 90% em duas coleções, e próxima a 70% na terceira.

No trabalho de Simões e Costa (2003) uma rede neural artificial foi utilizada na classificação de laranjas através de imagens. Os autores consideraram que apesar da automação de muitos setores industriais, em um sistema de produção, as frutas consideradas boas, são selecionadas através de inspeção visual humana. A principal característica utilizada foi a cor, que pode classificar a fruta em cinco padrões. No trabalho ficou comprovada a aplicabilidade do método para o domínio proposto que era classificar as frutas de acordo com a cor.

2.6 Métodos de validação da metodologia

Uma metodologia de classificação pode ser verificado em relação a sua acurácia a partir de um conjunto de dados ou a sua performance em relação a outros métodos classificadores. Algumas técnicas utilizadas com esse intuito serão apresentadas a seguir. No presente trabalho a técnica Cross Validation é utilizada com o objetivo de verificar a acurácia, nesse caso, da metodologia utilizada na classificação de dados.

A seguir são apresentadas as três técnicas mais comumente encontradas na bibliografia com suas ventagens e desvantagens:

2.6.1 *Handout*

Esse é um método simples, onde os dados são divididos em 2 grupos mutuamente exclusivos, sendo um grupo utilizado para treino e o outro para teste. Em geral 2/3 dos dados são utilizados no treinamento e 1/3 na etapa de testes. Os dois grupos devem conter dados suficientes de todos os padrões, nesse caso, suficientes para que o resultado final não seja comprometido pela falta de dados, de um determinado padrão, no grupo da etapa de treino e nem no grupo de testes. Apesar de ser um método de fácil implementação, a divisão entre conjunto de treinamento e teste deve ser bem estudada de forma que todos os padrões estejam bem representados nos dois grupos, por isso é um método recomendável quando o conjunto de dados possui um grande número de representações de cada padrão (KOHAVI, 1995) (BALDISSEROTTO, 2005).

2.6.2 *Cross Validation*

Nesse método, o conjunto de dados também é dividido em conjunto de treinamento e de teste, porém esses dois grupos variam a cada rodada de teste. O tipo de *cross validation* mais comumente utilizado é o *k-fold cross validation*, onde os dados são

divididos em k conjuntos mutuamente exclusivos e as etapas de treinamento e teste são realizadas k vezes, de forma que a cada rodada um conjunto diferente é utilizado como teste e os outros k-1 conjuntos são utilizados para treinamento. Esse método é recomendável quando o conjunto de dados não é tão grande, pois não existe uma preocupação de quais dados separar para teste e quais separar para treinamento, já que ao final da execução do método todos o dados terão participado dos conjuntos de treinamento e teste (KOHAVI, 1995) (BALDISSEROTTO, 2005).

2.6.3 *Leave-one-out*

Nessa técnica, do conjunto com o número total de dados n, são realizadas n rodadas de treinamento e teste, sendo que a cada rodada n-1 dados são utilizados para teste e o dados restantes são utilizados para treinamento. Pelas suas características, esse é um método recomendável para conjuntos de dados pequenos já que a quantidade de rodadas de treinamento e teste é igual a quantidade de dados o que o torna um método com alto custo computacional (KOHAVI, 1995) (BALDISSEROTTO, 2005).

3 Métodos de Solução

Entre os métodos mais utilizados para a resolução de problemas de programação linear destaca-se o método simplex e o método de pontos interiores. Depois da apresentação do método simplex,outros métodos com diferentes abordagens foram propostos (TODD, 2002). Porém, dentre os métodos existentes apenas o método de pontos interiores é atualmente competitivo em relação ao método simplex (BIXBY, 1992). A principal diferença entre esses dois métodos é que o método simplex busca a solução ótima em um dos vértices da região viável, enquanto o método de pontos interiores percorre o interior da região viável definindo um caminho até a solução ótima (MACULAN; FAMPA, 2006). Além disso, uma outra diferença é que o simplex exige muitas iterações com cálculos simples, enquanto no método de pontos interiores poucas iterações são exigidas, porém com cálculos mais elaborados. Apesar das vantagens do método de pontos interiores em relação ao método estudado neste trabalho, o método simplex possui melhor desempenho na resolução de problemas de pequeno e médio porte em relação ao método de pontos interiores.

3.1 Método Simplex

O método simplex é um dos algoritmos mais conhecidos para a resolução de problemas de programação linear. Surgiu há mais de 60 anos atrás e foi proposto por George Dantzig.

É um método iterativo, e sua ideia principal consiste no fato de que a cada iteração uma nova solução é encontrada, sempre melhor que a anterior até o ponto em que a solução ótima é obtida. Outra característica do método é o fato de ser matricial, ou seja, os dados a serem calculados são armazenados em matrizes.

Com a utilização do método, foi percebido que a cada iteração eram requeridos muitos cálculos sobre valores que nem sempre importavam para a iteração seguinte, fato que do ponto de vista computacional tornaria o método ineficiente. Esse método é chamado de método simplex padrão ou tabular. A partir desse fato foi desenvolvido o método simplex revisado visando a resolução de problemas de programação linear computacionalmente.

3.1.1 Princípio básico do método

A idéia do método simplex consiste em resolver repetidas vezes um sistema de equações lineares, e assim obter uma sucessão de soluções até encontrar a solução ótima. Ou seja, é um processo onde nos movemos de uma solução viável para outra sempre melhor ou pelo menos não pior.

Um problema de programação linear é sempre constituído de uma função objetivo e várias restrições. Geometricamente, essas restrições resultam em uma forma geométrica, no espaço n-dimensional sendo n o número de variáveis no modelo. E cada vértice dessa forma geométrica é considerado como uma possível solução, retomando o exemplo apresentado na seção 2.2.

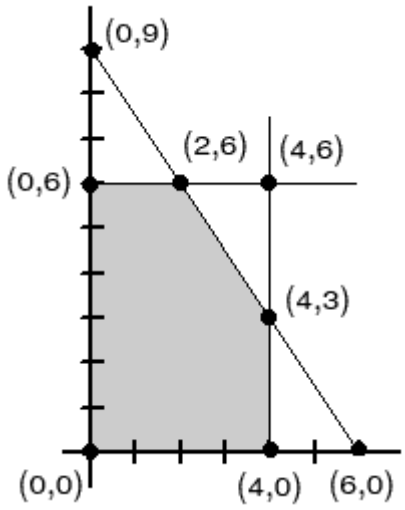


Figura 2: Representação gráfica de um modelo de programação linear

Nesse exemplo, de acordo com a representação geométrica do modelo, existem cinco possíveis soluções: ($x_1 = 0$ e $x_2 = 0$), ($x_1 = 0$ e $x_2 = 6$), ($x_1 = 2$ e $x_2 = 6$), ($x_1 = 4$ e $x_2 = 3$), ($x_1 = 4$ e $x_2 = 0$). Como o objetivo é maximizar, concluímos que a melhor solução é $x_1 = 2$ e $x_2 = 6$, em que $z = 36$. No método Simplex a cada iteração, antes da solução ótima ser obtida, é encontrada uma dessas possíveis soluções que se localizam nos vértices da forma geométrica.

3.1.2 Descrição do Método Simplex Revisado

O método simplex revisado surgiu como uma solução para evitar cálculos desnecessários. Esse método foi projetado para problemas a serem solucionados computacionalmente.

No simplex revisado, são armazenados na memória volátil apenas os dados realmente necessários. Além disso, os cálculos são realizados apenas sobre a coluna que é utilizada na iteração, o que evita cálculos com matrizes que poderiam acarretar uma imprecisão nos resultados.

Esse método mantém a característica do simplex, que é a troca entre a variável

que entra na base e a que sai (as variáveis da base são as que contribuem para o valor da solução ótima), além de também exigir certo esforço computacional. Porém sua grande vantagem é a economia de tempo e espaço, que é garantida pelo modo como é desenvolvida a solução.

Nesta seção são descritos os passos do algoritmo simplex revisado. O problema considerado é de maximização. A distinção entre matrizes, vetores e escalares foi feita da seguinte forma: letra maiúscula e negrito para matriz (**MATRIZ**); letra minúscula e negrito para vetor (**vetor**); letra em itálico para escalar (*escalar*).

Para a resolução do problema através do método simplex revisado, é necessário que o problema de programação linear esteja na forma padrão. Dizemos que um problema de programação linear está na sua forma padrão quando estiver no seguinte formato:

$Maximizar$ $\mathbf{cx}$
$Sujeito\ a$
$\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \geq 0$

ou

$Minimizar$ $\mathbf{cx}$
$Sujeito\ a$
$\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \geq 0$

com $\mathbf{b} \geq 0$.

Um problema de programação linear pode ser tranformado para a forma padrão acrescentando variáveis de folga, como no exemplo abaixo:

$$
\begin{aligned}
Maximize\ z = 3x_1 + 5x_2 \\
Sujeito\ a \\
1x_1 \leq 4 \\
2x_2 \leq 12 \\
3x_1 + 2x_2 \leq 18 \\
x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

Adicionando as variáveis de folga:

$$
\begin{aligned}
Maximize\ z = 3x_1 + 5x_2 \\
Sujeito\ a \\
1x_1 + x_3 = 4 \\
2x_2 + x_4 = 12 \\
3x_1 + 2x_2 + x_5 = 18 \\
x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

A inserção das variáveis de folga formam uma matriz base na matriz de coeficientes das restrições, as variáveis correspondentes a essa matriz base são denomindas variáveis básicas. Inicialmente as variáveis básicas são as variáveis de folga.

O vetor $\mathbf{c}$, de coeficientes na função objetivo, é dividido em duas componentes: $\mathbf{c_B}$ e $\mathbf{c_N}$, coeficientes das variáveis básicas e não-básicas, respectivamente. Por analogia, o vetor $\mathbf{x}$, de incógnitas do problema, é subdividido em $\mathbf{x_B}$ e $\mathbf{x_N}$. A matriz $\mathbf{A}$, de coeficientes das restrições, é dividida em duas submatrizes: $\mathbf{B}$ e $\mathbf{N}$, coeficientes das variáveis básicas e não-básicas, respectivamente. O vetor $\mathbf{b}$ são os valores das restrições. Os passos do algoritmo são os seguintes.

Passo 1: Calcular o vetor multiplicador: $\lambda = \mathbf{c_B^t B^{-1}}$

Passo 2: Escolher a variável que entra na base: $\mathbf{p} = \mathbf{c_N} - \lambda\mathbf{N}$
Se $\mathbf{p} = (\mathbf{c_N} - \lambda\mathbf{N}) \leq 0$ PARAR.
A solução $\mathbf{x_B}$ já é a solução ótima.
Se não, escolher uma coluna de $\mathbf{N}$, coluna $\mathbf{a_k}$, tal que $p_k = c_n k - \lambda\mathbf{a_k} > 0$
Um critério frequente é escolher a coluna $\mathbf{a_k}$ que resulte no maior valor de p_k.
Então, assumindo $e = k$, k representa o índice da coluna, $\mathbf{a_k}$ representa a coluna de $\mathbf{A}$ candidata a entrar na base.

Passo 3: Determinar a variável que sai da base: $\mathbf{y = B^{-1}a_k}$
Se $\forall i \frac{b_i}{y_i} \leq 0$ PARAR, a solução é não-limitada.
Senão, calcular: $\forall i \underset{\substack{1\leq i\leq m \\ y_i>0}}{Min} \left\{ \frac{(x_b)_i}{y_i} \right\}$ e guardar o índice correspondente a $s = i$. A variável $(x_b)_s$ sai da base.

Passo 4: Criar a matriz $\mathbf{E}$. $\mathbf{E} = (\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_s - 1}, \gamma, \mathbf{e_s + 1}, ..., \mathbf{e_m})$
Onde, cada coluna da matriz é um vetor unitário com exceção da coluna s, que corresponde ao vetor γ, calculado da seguinte maneira:
$\gamma^{\mathbf{t}} = \left(\frac{-\mathbf{A}_{1,e}}{\mathbf{A}_{s,e}} \ \frac{-\mathbf{A}_{2,e}}{\mathbf{A}_{s,e}} \ ... \ \frac{-\mathbf{A}_{s-1,e}}{\mathbf{A}_{s,e}} \ \frac{1}{\mathbf{A}_{s,e}} \ \frac{-\mathbf{A}_{s+1,e}}{\mathbf{A}_{s,e}} \ ... \ \frac{-\mathbf{A}_{m,e}}{\mathbf{A}_{s,e}} \right)$

Passo 5: Calcular nova matriz $\mathbf{B^{-1}}$: $\mathbf{B^{-1} = EB^{-1}}$

Passo 6: Atualizar os vetores $\mathbf{c_B}$ e $\mathbf{x_B}$.

A seguir, o modelo de programação linear apresentado na seção 2.2 é resolvido através do método Simplex Revisado.

$$
\begin{aligned}
Maximize\ z = 3x_1 + 5x_2 \\
Sujeito\ a \\
1x_1 \leq 4 \\
2x_2 \leq 12 \\
3x_1 + 2x_2 \leq 18 \\
x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

Adicionando as variáveis de folga:

$$
\begin{aligned}
Maximize\ z = 3x_1 + 5x_2 \\
Sujeito\ a \\
1x_1 + x_3 = 4 \\
2x_2 + x_4 = 12 \\
3x_1 + 2x_2 + x_5 = 18 \\
x_1 \geq 0, x_2 = 0
\end{aligned}
$$

Inicialização:

$$\mathbf{x_N} = [x_1 \ \ x_2] \qquad \mathbf{x_B} = [x_3 \ \ x_4 \ \ x_5] \qquad \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{B^{-1}}$$

$$\mathbf{x_B B} = \mathbf{b} \Rightarrow \mathbf{x_B} = \mathbf{B^{-1}b} \Rightarrow \mathbf{x_B} = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$$\mathbf{c_B} = [0 \ \ 0 \ \ 0] \qquad z = \mathbf{c_B x_B} = [0 \ \ 0 \ \ 0] \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = 0 \qquad \mathbf{c_N} = [3 \ \ 5] \qquad \mathbf{c_B} = [0 \ \ 0 \ \ 0]$$

$$\mathbf{A} = [\ \mathbf{N} \mid \mathbf{B}\ ] = \left[\begin{array}{cc|ccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{array}\right]$$

Iteração 1

Passo 1: Calcular o vetor multiplicador

$\lambda = \mathbf{c_B^t B^{-1}}$

$$\lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} [0 \;\; 0 \;\; 0] = [0 \;\; 0 \;\; 0]$$

Passo 2: Escolher a variável que entra na base $\mathbf{p} = \mathbf{c_N} - \lambda \mathbf{N}$

$$\mathbf{p} = [3 \;\; 5] - [0 \;\; 0 \;\; 0] \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} = [3 \;\; 5]$$

$k = 2$ a variável $(x_N)_k$ entra na base, $(x_N)_k = x_2$.

Passo 3: Determinar a variável que sai da base $\mathbf{y} = \mathbf{B^{-1}a_k}$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

$Min \left\{ \frac{4}{0}, \frac{12}{2}, \frac{18}{2} \right\} = \frac{12}{2}$

$s = 2$ a variável $(x_B)_s$ sai da base, $(x_B)_s = x_4$

Passo 4: Criar a matriz **E**.

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & \frac{1}{2} & 1 \\ 1 & -1 & 1 \end{bmatrix}$$

Passo 5: Calcular nova matriz $\mathbf{B^{-1}}$

$\mathbf{B^{-1}} = \mathbf{E B^{-1}}$

$$\mathbf{B^{-1}} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & \frac{1}{2} & 1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix}$$

Passo 6: Atualizar os vetores $\mathbf{c_B}$ e $\mathbf{x_B}$.

$$\mathbf{c_B} = [0 \;\; 5 \;\; 0] \quad \mathbf{x_B} = \begin{bmatrix} x_3 \\ x_2 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix} \quad z = [0 \;\; 5 \;\; 0] \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix} = 30$$

Iteração 2

Passo 1: Calcular o vetor multiplicador

$\lambda = \mathbf{c_B^t B^{-1}}$

$$\lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} [0 \;\; 5 \;\; 0] = [0 \;\; \frac{5}{2} \;\; 0]$$

Passo 2: Escolher a variável que entra na base $\mathbf{p} = \mathbf{c_N} - \lambda \mathbf{N}$

$$\mathbf{p} = [3 \;\; 0] - [0 \;\; \frac{5}{2} \;\; 0] \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 3 & 0 \end{bmatrix} = [3 \;\; -\frac{5}{2}]$$ $k = 1$ a variável $(x_N)_k$ entra na base, $(x_N)_k = x_1$.

Passo 3: Determinar a variável que sai da base $\mathbf{y} = \mathbf{B^{-1}a_k}$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$Min \left\{ \frac{4}{1}, \frac{6}{0}, \frac{6}{3} \right\} = \frac{6}{3} = 2$

$s = 3$ a variável $(x_B)_s$ sai da base, $x_B)_s = x_5$

Passo 4: Criar a matriz **E**.

$$\mathbf{E} = \begin{bmatrix} 1 & 1 & -\frac{1}{3} \\ 1 & 1 & 0 \\ 1 & 1 & \frac{1}{3} \end{bmatrix}$$

Passo 5: Calcular nova matriz $\mathbf{B^{-1}}$

$\mathbf{B^{-1}} = \mathbf{E B^{-1}}$

$$\mathbf{B^{-1}} = \begin{bmatrix} 1 & 1 & -\frac{1}{3} \\ 1 & 1 & 0 \\ 1 & 1 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

Passo 6: Atualizar os vetores $\mathbf{c_B}$ e $\mathbf{x_B}$.

$\mathbf{c_B} = [0 \;\; 5 \;\; 3] \qquad \mathbf{x_B} = [x_3 \;\; x_2 \;\; x_1]$

$$\begin{bmatrix} x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -\frac{1}{3} \\ 1 & 1 & 0 \\ 1 & 1 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix} \qquad z = [0 \;\; 5 \;\; 3] \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix} = 36$$

Iteração 3

Passo 1: Calcular o vetor multiplicador

$\lambda = \mathbf{c_B^t B^{-1}}$

$$\lambda = \begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix} [0 \quad 5 \quad 3] = [0 \quad \frac{3}{2} \quad 1]$$

Passo 2: Escolher a variável que entra na base $\mathbf{p} = \mathbf{c_N} - \lambda \mathbf{N}$

$$\mathbf{p} = [0 \quad 0] - [0 \quad \frac{3}{2} \quad 1] \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = [-1 \quad -\frac{3}{2}]$$

Não há valor maior que 0, então chegamos a solução ótima. $z = 36$

3.2 Método de Pontos Interiores

Em 1984, Karmarkar revolucionou a área de programação linear com a publicação de um algoritmo de complexidade polinomial que apresentou bom desempenho quando aplicado a problemas práticos (MACULAN; FAMPA, 2006). Essa publicação deu origem a um novo campo de pesquisa, chamado de método dos pontos interiores.

O método de pontos interiores tem como principal característica realizar a busca por soluções no interior da região viável do problema, até encontrar a solução ótima (MENEZES, 2008). Em teoria, o método de pontos interiores é melhor que o método simplex, principalmente quando se leva em conta o critério de complexidade de pior caso. O método de pontos interiores possui complexidade polinomial, enquanto o método simplex possui complexidade exponencial No entanto, na prática ambos os métodos concorrem até hoje. Já que o sucesso do

método depende da estrutura dos problemas, da esparsidade [1] e da arquitetura dos computadores (MACULAN; FAMPA, 2006).

[1]Quando uma matriz possui uma grande proporção de elementos nulos diz-se que é uma matriz esparsa (MUNARI, 2009).

4 Classificação de dados

Para a realização da classificação de dados em padrões são necessárias as etapas de obtenção e organização dos dados, treinamento, que na metologia utilizada nesse trabalho corresponde a geração dos hiperplanos separadores através do MPLHS, e teste, onde é feita a classificação de um dado com padrão inicialmente desconhecido. Todo o processo, desde a organização dos dados até a etapa de classificação, foi implementado utilizando a linguagem de programação Java. A implementação foi feita em módulos, organizados da seguinte forma: Módulo organizador de dados, Módulo de geração de classificadores, Módulo de classificação, Módulo validador. No módulo organizador de dados os conjuntos de dados são organizados em subgrupos, cada subgrupo contendo vetores de todos os padrões, essa organização é importante para a etapa de validação do método. No Módulo de geração de classificadores está implementado o MPLHS e são gerados os hiperplanos separadores. Os Módulos de classificação e validador são utilizados na etapa de teste, o primeiro realiza a classificação de um vetor em um dos padrões separados pelo hiperplano e o segundo realiza a validação do método de classificação de dados utilizando o método cross validation.

Na figura 3 estão representadas as etapas da metodologia utilizada para a classificação de dados em padrões.

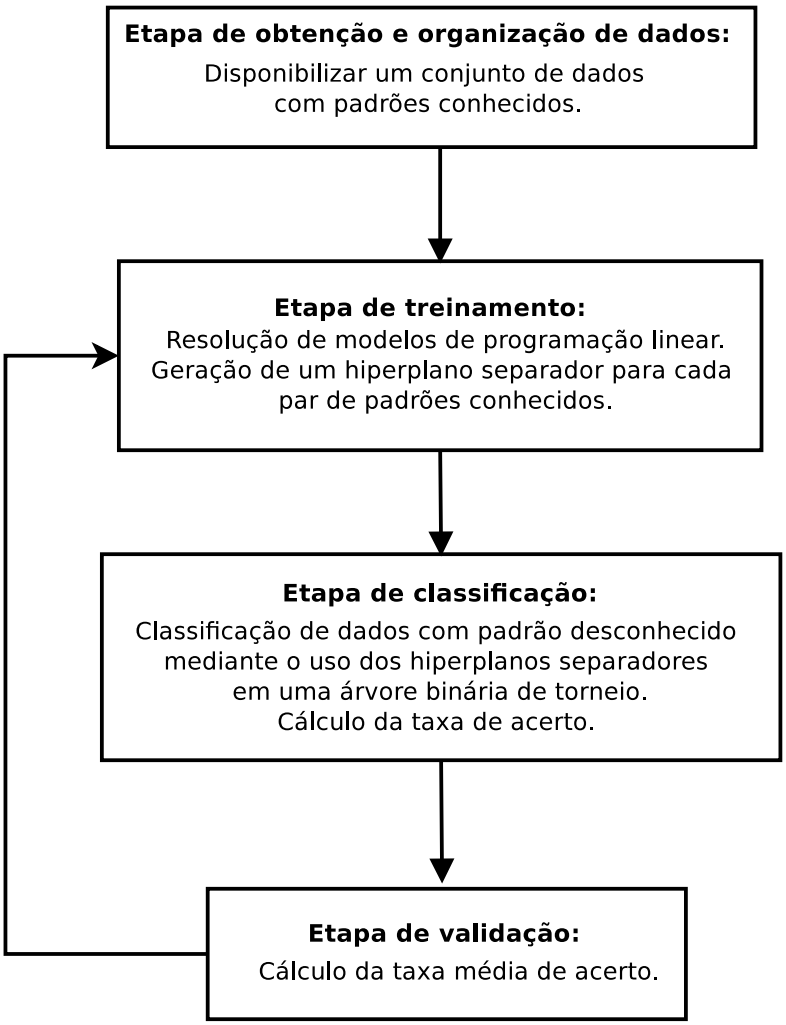


Figura 3: Diagrama das etapas da metodologia

Primeiramente os dados foram obtidos já no formato de vetores de características ou extraídos de imagens, e depois foram divididos em subgrupos. Em seguida, foram submetidos ao MPLHS, responsável por gerar os classificadores e por último vetores com padrão desconhecido foram classificados. As duas últimas etapas são repetidas a cada ciclo de teste, essas repetições são controladas pelo módulo validador. Nas seções seguintes essas etapas serão apresentadas de forma mais detalhada.

4.1 Etapa de obtenção e organização de dados

Nessa etapa inicial é obtido um conjunto de dados com um número de padrões p, esse conjunto é composto por vetores contendo n características. Para a classificação dos dados, o número p de padrões deve ser previamente conhecido, assim como o número de vetores de características para cada padrão. A organização consiste em dividir o conjunto de dados em subconjuntos. A cada ciclo de treinamento e teste, alguns desses subconjuntos de dados são utilizados para treinamento enquanto um subconjunto será utilizados para teste. Cada vetor utilizado para teste será classificado em um dos p padrões previamente conhecidos.

4.2 Etapa de treinamento

Na tarefa onde o objetivo é a separação de dois ou mais conjuntos de pontos, sendo que cada conjunto representa um padrão, busca-se um método para que essa separação seja obtida com resultado ótimo. Cada padrão é representado por de um conjunto de pontos e cada ponto é representado por um vetor de características. Em (BENNETT; MANGASARIAN, 1992) é proposto um MPLHS capaz de gerar um hiperplano, que separa dois conjuntos de pontos, minimizando a soma das médias das violações dos pontos que permaneceram do lado "errado"do hiperplano. Dessa forma o MPLHS é capaz de gerar um hiperplano separador para conjuntos de pontos linearmente separáveis e para conjuntos linearmente inseparáveis, o MPLHS gera o melhor hiperplano de forma que essa soma das médias calculada seja mínima. Na figura abaixo estão ilustrados os casos de dois conjuntos linearmente separáveis e dois conjuntos linearmente inseparáveis.

Em 4(a) são apresentados dois conjuntos linearmente separáveis e em 4(b) são apresentados dois conjuntos linearmente inseparáveis É possível observar que na figura 4(a) podem ser traçadas inúmeras retas separando os dois conjuntos de pontos. Já na figura 4(b) o MPLHS de programação linear poderá definir a melhor

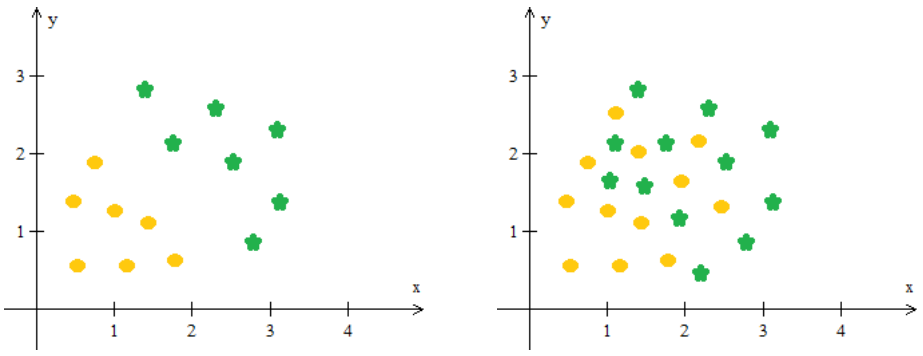


Figura 4: Conjuntos linearmente separáveis e linearmente inseparáveis

reta possível para separar os pontos.

A proposta do MPLHS é separar dois conjuntos de pontos, porém também pode ser utilizado em problemas onde busca-se a separação de múltiplos padrões. O presente trabalho foca nessa última abordagem, em todos os testes realizados o número de conjuntos de pontos é igual ou maior que três.

4.2.1 O modelo (MPLHS)

Considerando dois conjuntos, no espaço n-dimensional R^n, sendo o conjunto A representado pela matriz m x n, e o conjunto B representado pela matriz k x n. Contextualizando, k e m são as quantidades de pontos em cada conjunto, sendo que cada vetor de características é um ponto e n é a quantidade de características. O MPLHS calcula um hiperplano que procura separar os pontos dos dois conjuntos. De forma que os m pontos do conjunto A fiquem de um lado do hiperplano e os k pontos do conjunto B fiquem do outro lado do hiperplano. Nos casos em que os conjuntos não são separáveis, alguns pontos ficam localizados do lado "errado"do hiperplano. Na figura abaixo o MPLHS é apresentado. E a seguir é apresentado de forma mais detalhada.

$$\min_{\omega,\gamma,y,z} \frac{e^T y}{m} + \frac{e^T z}{k}$$

$$s.a. \begin{cases} -A\omega + e\gamma + e \leq y \\ B\omega - e\gamma + e \leq z \\ y \geq 0, y \geq 0 \end{cases}$$

O MPLHS determina o hiperplano:

$$x\omega = \gamma$$

Onde, é ω o vetor normal ao hiperplano e γ é um escalar. De forma que:

$$A\omega \geq e\gamma$$

e

$$B\omega \leq e\gamma$$

Onde e é um vetor de 1's com dimensão m para o conjunto A e k para o conjunto B. Para conjuntos linearmente separáveis torna-se fácil traçar um hiperplano separador, porém para conjuntos linearmente inseparáveis é necessária uma estratégia. Na busca pelo melhor hiperplano, o MPLHS gera dois hiperplanos limites somando ou subtraindo 1 unidade a equação do hiperplano separador:

$$A\omega \geq e\gamma + e$$

e

$$B\omega \leq e\gamma - e$$

De forma que o hiperplano separador fica localizado exatamente entre esses dois hiperplanos que formam uma faixa limite. Nesse caso a estratégia utilizada é a minimização da soma das médias das violações dos pontos localizados do lado "errado"do hiperplano e também é considerada um violação o ponto que se localiza na faixa limite. Detalhando o MPLHS de acordo com Trevisan (2010),

considerando x como um vetor em R^n, definimos:

1. $(x_i)_+ = \max_{i=1,2,\ldots,n} \{x_i, 0\}$
2. $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Podemos escrever o problema de minimização com norma da seguinte forma:

$$\min_{\omega,\gamma} \frac{1}{m} \left\|(-A\omega + e\gamma + e)_+\right\|_1 + \frac{1}{k} \left\|(B\omega - e\gamma + e)_+\right\|_1$$

Seja $g : R^n \mapsto R^m, h : R^n \mapsto R^k$ e S um subconjunto de R^n . Os problemas abaixo possuem soluções idênticas:

$$\min_{x \in S} \|g(x)_+\|_1 + \|h(x)_+\|_1$$

$$\min_{x \in S} e^t y + e^t z$$

$$s.a. \begin{cases} y \geq g(x) \\ z \geq h(x) \\ y \geq 0, z \geq 0 \end{cases}$$

Como $g(x)_+ \geq g(x)$ e $h(x)_+ \geq h(x)$ podemos observar que os valores ótimos de y e z podem ser dados pelas igualdades $y = g(x)_+$ e $z = h(x)_+$.

A partir do problema de minimização temos o problema de programação linear equivalente:

$$\min_{\omega,\gamma,y,z} \frac{e^T y}{m} + \frac{e^T z}{k} \tag{4.1}$$

$$Sujeito\ a$$

$$-A\omega + e\gamma + e \leq y \tag{4.2}$$

$$B\omega - e\gamma + e \leq z \tag{4.3}$$

$$y \geq 0, y \geq 0 \tag{4.4}$$

Dados do MPLHS:

- m: quantidade de vetores de características correspondente ao primeiro padrão;
- k: quantidade de vetores de características correspondente ao segundo padrão;
- A: Matriz contendo o conjunto de vetores de caraterísticas do primeiro padrão;
- B: Matriz contendo o conjunto de vetores de caraterísticas do segundo padrão;
- e: vetor coluna unitário;

Variáveis do MPLHS:

- ω: coeficientes do hiperplano que separa os conjuntos de pontos dos dois padrões;
- γ: constante do hiperplano que separa os conjuntos de pontos dos dois padrões;
- y: vetor contendo o quanto um ponto do conjunto A violou o hiperplano limite;
- z: vetor contendo o quanto um ponto do conjunto B violou o hiperplano limite;.

Formulação Matemática: A função objetivo 4.1 procura minimizar a soma das médias das violações dos pontos de ambos conjuntos. A restrição 4.2 exige que os pontos do conjunto A permaneçam do mesmo lado da reta fora da faixa limite definida pelos hiperplanos limite. A restrição 4.3 é análoga a restrição 4.2 aplicada ao conjunto B. As equações em 4.4 definem a natureza das variáveis do MPLHS, como sendo não negativas.

4.2.2 Exemplo Ilustrativo do MPLHS

Para ilustrar o MPLHS vamos considerar dois exemplos em R^2 (BENNETT; MANGASARIAN, 1992) de forma que a ilustração gráfica fique mais facilmente compreensível.

- Exemplo1:

Considerando as matrizes a seguir:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 2 & 1 \\ 2 & 2 \\ 2 & 3 \\ 2 & 4 \\ 3 & 3 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 1 \\ 4 & 2 \\ 4 & 3 \\ 5 & 2 \\ 5 & 3 \\ 5 & 4 \\ 6 & 2 \end{bmatrix}$$

Nesse caso, temos: m = 9 (quantidade de pontos da matriz A) e k = 7 (quantidade de pontos da matriz B). Resolvendo o MPLHS temos:

* Valor da função objetivo = 0

* ω = [-2 0]

* γ = -7

* y = [0 0 0 0 0 0 0 0 0]

* z = [0 0 0 0 0 0 0]

Como o valor da função objetivo depende dos vetores y e z, podemos notar que o valor 0 indica que os conjuntos A e B são linearmente separáveis. A seguir é apresentada a representação gráfica dos pontos e das retas geradas pelo MPLHS.

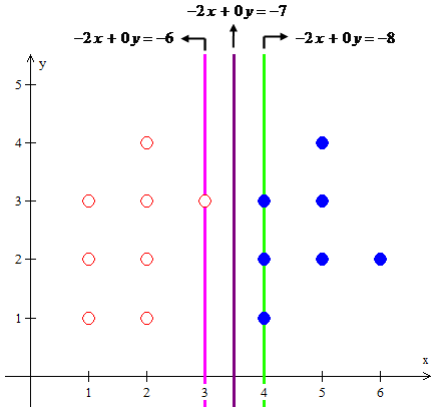


Figura 5: Representação ilustrativa do exemplo 1

A partir da representação gráfica podemos perceber que a reta separadora $-2x + 0y = -7$ se localizou exatamente entre os dois conjuntos com o auxílio das duas retas: $-2x + 0y = -6$ e $-2x + 0y = -8$.

- Exemplo2:

Vamos considerar agora as matrizes:

$$A = \begin{bmatrix} 3 & 4 \\ 4.3 & 4.5 \\ 4.5 & 2 \\ 3 & 5.5 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 2 \\ 4.5 & 3.5 \\ 5 & 3 \end{bmatrix}$$

Neste exemplo temos: m = 4 e k = 3. De acordo com o MPLHS, temos os seguintes valores:

∗ Valor da função objetivo = 0.92

* ω = [-0.91 0.91]

* γ = -0.82

* y = [0 0 2.46 0]

* z = [0 0.91 0]

Com o valor da função objetivo diferente de zero, temos dois conjuntos linearmente inseparáveis, como ilustrado na figura a seguir. Nesse caso, a reta separa os pontos de ambos os conjuntos minimizando a soma das médias das violações.

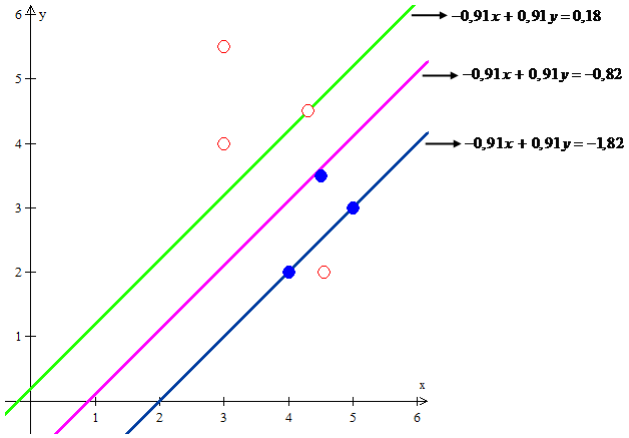


Figura 6: Representação ilustrativa do exemplo 2

Nesse exemplo notamos que o valor 2.46 no vetor y é referente ao ponto do conjunto A que está localizado do lado "errado"da reta. E o valor 0.91 é referente ao ponto do conjunto B que está localizado mais próximo à reta separadora, apesar do ponto estar localizado do ponto correto da reta ele está localizado acima da reta limite $-0.91x + 0.91y = -1.82$. Concluindo que, quando um ponto está localizado do lado "correto"do hiperplano ou em cima de um dos hiperplanos limites, seu valor correspondente no vetor y ou z é nulo. Mas se um ponto está localizado do lado "errado"do hiperplano ou dentro da faixa formado pelos hiperplanos limites, seu valor correspondente no vetor y ou z é positivo.

4.2.3 Geração de hiperplanos separadores

Na etapa de treinamento, os padrões são agrupados em pares e dessa forma são submetidos ao MPLHS, já que o modelo gera um hiperplano que separa dois padrões. Dessa forma para um conjunto de dados com N padrões, a quantidade de pares formados é dada pela fórmula:

$$C_N^2 = \frac{N!}{2 \times (N-2)!)}$$

A etapa de treinamento consiste em gerar um hiperplano para cada par formado. No caso de um conjunto de dados constituído por 3 padrões, são gerados 3 hiperplanos separadores, da seguinte forma:

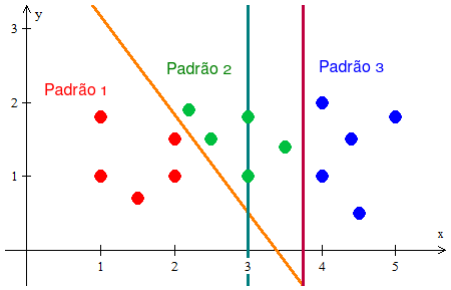
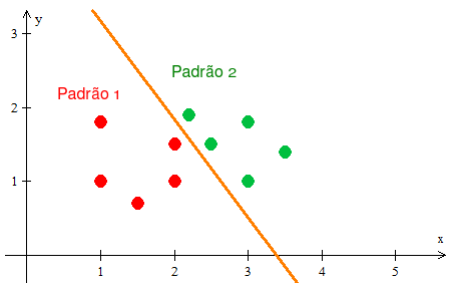
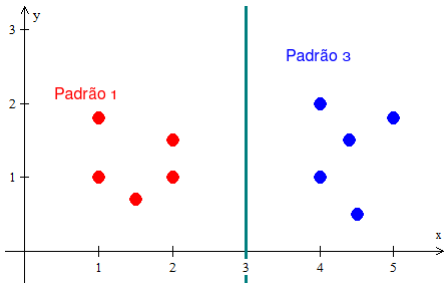


Figura 7: Hiperplanos para um conjunto de dados com 3 padrões

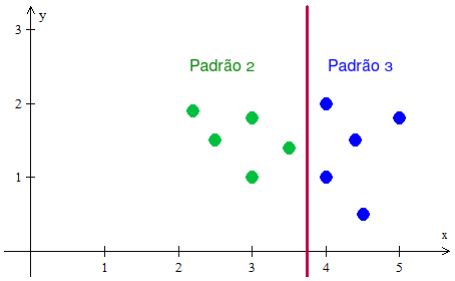
Na figura 7 é apresentada uma visão geral dos 3 conjuntos de pontos (padrões) e dos 3 hiperplanos separadores. A seguir os hiperplanos são apresentados separadamente, o MPLHS é resolvido 3 vezes, uma para cada par de padrões e cada resolução o modelo calcula um hiperplano separador para o par.



(a) Classificador dos padrões 1 e 2



(b) Classificador dos padrões 1 e 3



(c) Classificador dos padrões 2 e 3

Figura 8: Hiperplanos para um conjunto de dados com os 3 padrões organizados em pares

Na figura 8(a), os padrões 1 e 2 foram submetidos ao MPLHS, e o hiperplano separador foi gerado para esses dois padrões. O mesmo ocorre nas figuras 8(b) e 8(c), para cada para de padrões submetidos ao MPLHS, um hiperplano separador foi gerado.

O processo de treinamento corresponde ao módulo de geração de classificadores. Esse módulo foi implementado na linguagem de programação JAVA, utilizada como interface com o software CPLEX da IBM. O MPLHS é resolvido utilizando o método simplex revisado e toda etapa de resolução é feita pelo software CPLEX. A implementação desse módulo encontra-se no Apêndice A.

4.3 Etapa de classificação

Após a geração dos classificadores utilizando o MPLHS, é necessário um segundo procedimento onde ocorra de fato a classificação de um vetor com padrão inicial-

mente desconhecido. Nessa etapa de classificação foi implementada uma estrutura de árvore binária de torneio. Em seus trabalhos, Hadid (2005) e Dyer (2005) utilizaram essa mesma estrutura na etapa de classificação após a utilização do MPLHS para gerar os classificadores. No processo de teste, um vetor de características é submetido a estrutura de árvore binária de torneio e classificado em um dos p padrões do conjunto de dados. A cada teste uma árvore binária de torneio é construída a partir dos nós folhas e um vetor de características é classificado. Na estrutura de árvore binária de torneio os padrões são confrontados aos pares, o nó pai é obtido com base no hiperplano que separa os padrões que estão no nós filhos.A seguir é apresentada uma figura para ilustrar o mecanismo da árvore de torneio.

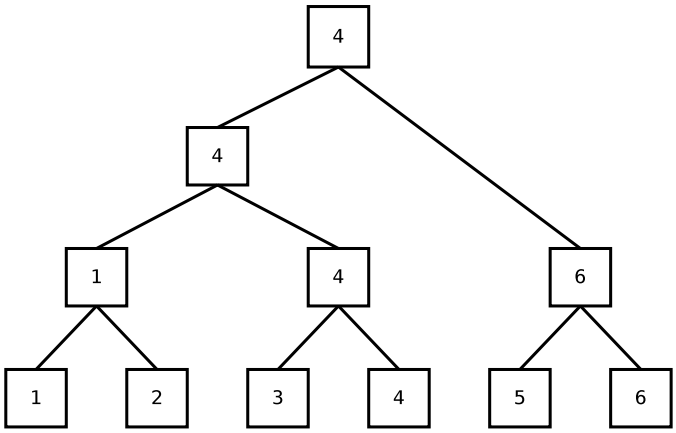


Figura 9: Ilustração de uma árvore de torneio com 6 padrões

Por analogia podemos pensar na árvore como a estrutura utilizada em um torneio esportivo e nos padrões como os times do torneio, os times devem se confrontar até que ao final saia o vencedor. Da mesma forma, analisamos os padrões até que ao final se obtenha o padrão ao qual o vetor pertence. Na árvore da figura 9, consideramos 6 padrões e o conjunto desconhecido pertencendo ao padrão 4. A quantidade de nós folhas deve ser a quantidade de padrões e a formação dos pares é arbitrária. Analisando o primeiro par, deve ser analisado em qual dos dois padrões o conjunto desconhecido mais se encaixa, ou seja, deve ser analisado de qual lado

do hiperplano (gerado pelo MPLHS quando foram submetidos os padrões 1 e 2) o conjunto desconhecido se encontra. Na figura exemplo, o ponto representado pelo vetor de características ficou localizado do mesmo lado do hiperplano que os pontos do padrão 1, por isso esse padrão continuou no próximo nível da árvore e o padrão 2 foi excluído. Esse procedimento ocorre por analogia em todos os pares, até que ao final, quando analisados os padrões 4 e 6, foi definido que o vetor pertence ao padrão 4.

4.3.1 Classificação com base em um hiperplano

Dado um vetor x de características, para determinar a qual padrão esse vetor pertence deve ser analisado de qual lado do hiperplano o ponto dado pelo vetor, se localiza, considerando que

Se $\begin{bmatrix}\omega_1 & ... & \omega_n\end{bmatrix} \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \geq \gamma$, x pertence ao padrão A;

Se $\begin{bmatrix}\omega_1 & ... & \omega_n\end{bmatrix} \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} \leq \gamma$, x pertence ao padrão B.

Considerando o seguinte hiperplano em R^4: ω = [1.21 8.0 -6.4 -19.2] γ = -33.6

E o vetor x = [6.7 3.0 5.2 2.3], temos:

$$\begin{bmatrix}1.21 & 8.0 & -6.4 & -19.2\end{bmatrix} \begin{bmatrix} 6.7 \\ 3.0 \\ 5.2 \\ 2.3 \end{bmatrix} \leq -33.6$$

Nesse caso o ponto representado pelo vetor x permaneceu do mesmo lado do hiperplano que os pontos do padrão B, sendo assim o vetor de características x é reconhecido como pertencente ao padrão B.

4.3.2 Classificação utilizando a estrutura da árvore binária de torneio

Para o caso apresentado anteriormente foram considerados apenas dois padrões, porém em um problema que considera três ou mais padrões, é necessária uma lógica para que seja realizada classificação do dado em um padrão entre os p padrões existentes. Nesse caso é utilizada a árvore binária de torneio. Considerando um exemplo com três padrões (A, B e C), e os seguintes hiperplanos em R^4:

- Hiperplano separador dos padrões A e B

  ω = [1.14 0.0 0.0 -7.86]

  γ = 0.0

- Hiperplano separador dos padrões A e C

  ω = [0.62 0.0 -1.03 0.0]

  γ = 0.0

- Hiperplano separador dos padrões B e C

  ω = [1.21 8.0 -6.4 -19.2]

  γ = -33.6

Para determinar a qual dos três padrões pertence o vetor $x = [4.8\ \ 3.0\ \ 1.4\ \ 0.3]$, a árvore binária de torneio é montada dinamicamente iniciando pelo nível de maior profundidade, nesse exemplo iniciamos com os padrões A e B.

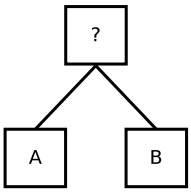


Figura 10: Nós folhas no nível de maior profundidade

Iniciando a classificação do vetor x analisando os padrões A e B como ilustrado pela figura 10, deve ser verificado de qual lado do hiperplano separador dos padrões A e B o ponto representado pelo vetor x permanecerá $\begin{bmatrix} 4.8 & 3.0 & 1.4 & 0.3 \end{bmatrix} \begin{bmatrix} 1.14 \\ 0.0 \\ 0.0 \\ -7.86 \end{bmatrix} \geq 0.0$

Nessa primeira análise, o ponto representado pelo vetor x ficou localizado do mesmo lado do hiperplano que os pontos do padrão A. Então o padrão A é elevado ao próximo nível da árvore, como ilustrado na figura a seguir.

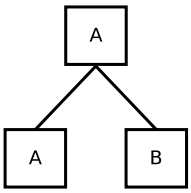


Figura 11: Padrão A é elevado ao próximo nível da árvore

Resta ainda analisar o padrão C.

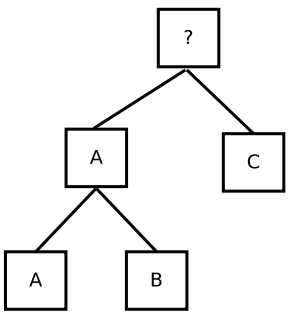


Figura 12: Segundo nível da árvore

Como mostrado na figura 12, é feita a análise utilizando o hiperplano separador dos padrões A e C. $\begin{bmatrix} 4.8 & 3.0 & 1.4 & 0.3 \end{bmatrix} \begin{bmatrix} 0.62 \\ 0.0 \\ -1.03 \\ 0.0 \end{bmatrix} \geq -33.6$

Analisando o hiperplano separador dos padrões A e C é determinado que o ponto representado pelo vetor x ficou localizado do mesmo lado que os pontos do padrão A. Como todos os padrões (A, B e C) já foram analisados e a árvore foi completada, então, finalmente, o vetor x é reconhecido como pertencente ao padrão A.

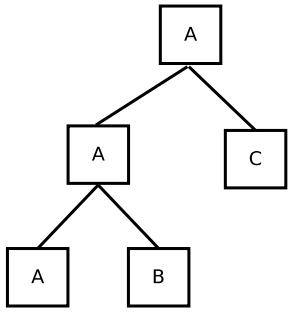


Figura 13: Árvore completa

A árvore completa é mostrada na figura 13, após analisar todos os padrões, e o nó raiz da árvore ser preenchido, a etapa de classificação é completada, e o vetor com padrão inicialmente desconhecido é classificado como pertencente ao padrão que está no nó raiz da árvore.

4.4 Etapa de validação

Para testar a metodologia adotada neste trabalho para a classificação de dados, foi utilizado o método *k-fold cross validation* (KOHAVI, 1995) (BALDISSEROTTO,

2005). Através dos resultados desse método é possível analisar a acurácia da metodologia de classificação, ou seja, a capacidade de classificar um dado com o seu padrão correto. Entre os três métodos de validação apresentados no referencial teórico: *Handout*, *cross validation* e *Leave One Out*. O método *cross validation* foi escolhido como melhor opção já que os teste são feitos com conjuntos de dados com tamanhos variados. Para conjuntos de dados muito grandes o método *leave one out* se tornaria computacionalmente custoso e o método *handout* poderia ter seu resultado comprometido dependendo dos parâmetros escolhidos na separação de dados para teste e de dados para validação. Além disso, o método *cross validation* é de fácil implementação e permite a utilização de todos os dados na etapa de teste e treinamento. Na figura abaixo o método *k-fold cross validation* é ilustrado com o parâmetro 5 *fold*, apesar desse parâmetro não ter sido utilizado em nenhum teste, foi utilizado para um facilitar a ilustração.

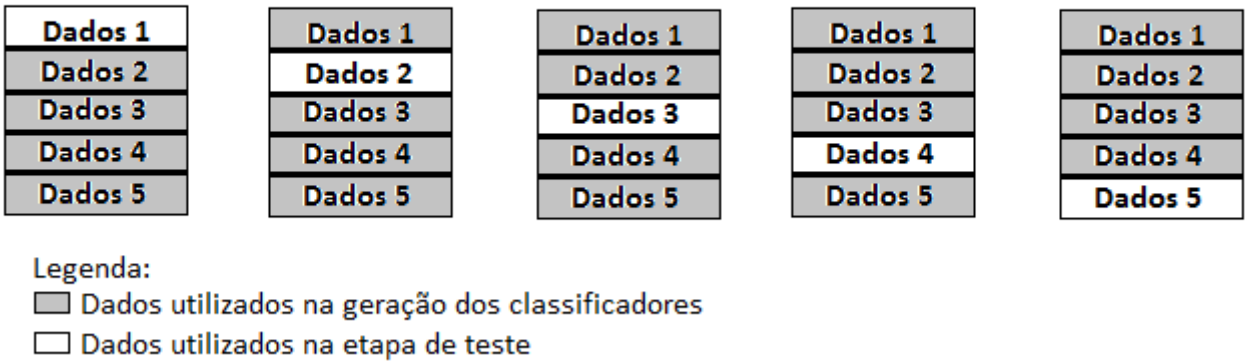


Figura 14: Mecanismo do método de validação 5- *fold cross validation*

Na ilustração o conjunto de dados é dividido em 5 subconjuntos e consequentemente são realizados 5 ciclos de treinamento e testes. A cada etapa 4 subconjuntos são utilizados para treinamento e um subconjunto diferente é utilizado para teste.

A metodologia *k-fold cross validation* foi implementada na linguagem de programação Java. Os dados são organizados em k arquivos (subconjuntos) e são realizados k ciclos de treinamento e teste. A implementação seleciona os arquivos para treinamento e o arquivo para teste, na etapa de teste de cada ciclo, todos os vetores do arquivo de teste são classificados. A cada ciclo de treinamento e teste a porcentagem de acerto de classificação é calculada da seguinte forma:

$$A = \frac{acertos}{total} \times 100$$

Onde total é a quantidade de vetores para classificação submetidos a cada ciclo de teste. E acertos é a quantidade de vetores que foram classificados corretamente. Ao final dos k ciclos de treinamento e teste, é calculada a média das porcentagens de acerto obtidas a cada ciclo

5 *Experimentos Computacionais*

Para a realização do experimentos computacionais foi necessário, primeiramente, obter e organizar os dados. Posteriormente os dados selecionados para treinamento são submetidos ao modelo para geração dos hiperplanos separadores e os dados para teste são classificados utilizando a árvore binária de torneio. Todos os experimento foram feitos utilizando um notebook com processador Intel Core i3, 2.27 GHz, 3 GB de RAM, sistema operacional Ubuntu 12.04, NetBeans IDE 7.1.2 e o software CPLEX da IBM.

5.1 Conjuntos de Dados

Foram realizados testes com quatro conjuntos de dados: dígitos de 0 a 9 escritos manualmente, gestos da Língua Brasileira de Sinais, espécies da planta Iris e expressões faciais . Desses conjuntos, os três primeiros foram obtidos do repositório de dados Bache e Lichman (2013) e já estavam representados na forma de vetores de características. O último conjunto foi gerado a partir do conjunto de imagens JAFFE (LYONS; GYOB, 1997). Os p padrões de cada conjunto de dados são numerados de 1 a p. Nos quatro conjuntos de dados todos os vetores de características possuem um atributo identificando a qual padrão o vetor pertence, o vetor a seguir, por exemplo, pertence ao conjunto de dados com espécies da planta Iris:

[5.1 3.5 1.4 0.2 1]

O valor 1 no final do vetor significa que esse vetor de características é de uma planta da espécie 1, ou seja, esse vetor pertence ao padrão 1 desse conjunto de dados. Na etapa de treinamento esse atributo é retirado do vetor. A seguir são apresentadas as caraterísticas de cada um dos quatro conjuntos de dados:

- **Dígitos de 0 a 9 escritos manualmente (ALPAYDIN; ALIMOGLU, 2013)** Na formação desse conjunto de dados 250 dígitos entre 0 e 9 foram escritos de forma aleatória por 44 pessoas, totalizando 11000 vetores de características, porém estão disponíveis 10992 vetores de características. Durante a coleta dos dados foram recolhidas, em intervalos fixos de 100 milissegundos, as coordenadas e a pressão da caneta sobre a superfície enquanto o dígito era escrito. No conjunto de dados utilizado foram considerados apenas os valores das coordenadas. Os dados foram reorganizados utilizando interpolação linear, foram utilizados 8 pontos, e obtidos vetores com 16 características. Cada vetor é composto por 16 atributos que variam entre 0 e 100 e mais um valor variando entre 0 e 9 que representa o classe a qual o vetor pertence. Os dados estão distribuídos como na tabela a seguir:

Classe	Quantidade de vetores
0	1143
1	1143
2	1144
3	1055
4	1144
5	1055
6	1056
7	1142
8	1055
9	1055

Tabela 1: Quantidade de vetores para cada dígito

- **Gestos da Língua Brasileira de Sinais (LIBRAS) (DIAS; BíSCARO,**) Esse conjunto de dados é composta por 15 padrões, ou seja, nela estão presentes 15 sinais da LIBRAS, sendo 24 vetores de características para cada padrão, totalizando 360 vetores de características. Os dados foram extraídos de vídeos com tempo médio de 7 segundos, em cada vídeo um movimento é executado e depois é representado como uma curva bidimensional. No pre processamento foram selecionados 45 frames de cada vídeo, em cada frame a coordenada do ponto central da mão é encontrado, compondo uma curva com 45 pontos. As coordenadas dos 45 pontos formam o vetor de características com 90 valores, os 45 primeiros valores representam os valores de x e o restante os valores de y. O vetor de características tem no total 91 valores, sendo que 90 deles caracterizam o movimento e o último valor representa o padrão ao qual o vetor pertence.

- **Espécies da planta Iris (FISHER, 1936)** Nesse conjunto de dados são representados três espécies da planta Iris, é composto por 50 vetores de características de cada espécie, totalizando 150 vetores. Cada vetor é composto por 4 características: comprimento da sépala (parte da flor que dá sustentação às pétalas), largura da sépala, comprimento da pétala, largura da pétala; e mais um atributo representando a espécie que o vetor representa.

- **Expressões faciais** No caso das expressões faciais, as imagens foram obtidas do conjunto de dados JAFFE (LYONS; GYOB, 1997) e o pré processamento e a extração e características foram implementados na linguagem de programação Python. Inicialmente a proposta do presente trabalho era focar apenas na classificação de expressões faciais utilizando a programação linear. Durante a pesquisa não foi encontrada nenhum conjunto de dados que disponibilizasse os vetores de características da imagens, portanto foi necessária a implementação para a obtenção dos dados. Posteriormente foi verificado

que seria necessário um aprofundamento do estudo na área da visão computacional, para que os vetores de características não comprometessem o método classificador. Portanto esses dados foram obtidos através de uma implementação superficial de extração de características. Após a obtenção do banco de imagens JAFFE. As imagens foram recortadas a fim de isolar a região da face, utilizando a elinguagem de programação python, como mostrado na figura

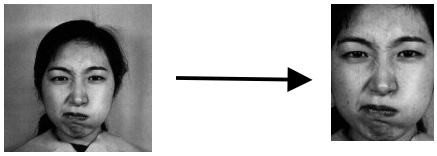


Figura 15: Pre processamento das imagens do banco JAFFE

Na extração dos vetores de características foi utilizado o método Local Binary Patterns (LBP) que é um classificador de texturas. A cada pixel da imagem é atribuído um código, que é gerado a partir dos pixels ao redor. Tomando como referência o pixel central, a cada pixel vizinho é atribuído o valor 0 ou 1: se o valor do pixel vizinho for menor que o valor do pixel central, o valor 0 é atribuído, se for maior, o valor atribuído é 1. A partir daí, é gerado um código binário que é transformado em um valor decimal, esse valor decimal é o código LBP do pixel central (SHAN; GONG; MCOWAN, 2009). A figura abaixo demonstra como é calculado o código LBP.

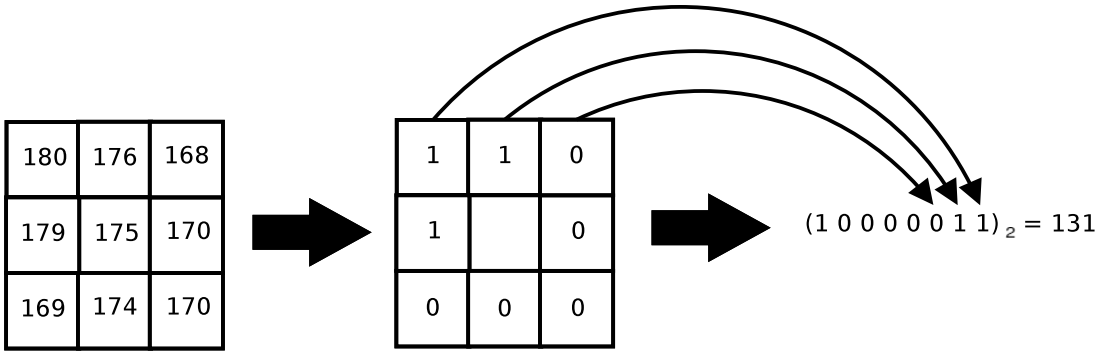


Figura 16: Cálculo do código LBP

Na implementação, primeiramente, a imagem é dividida em blocos, é gerado o histograma dos códigos LBP de cada bloco, por fim, os histogramas são concatenados. Esse resultado final é o descritor de texturas da imagem. A figura 17 representa esse processo.

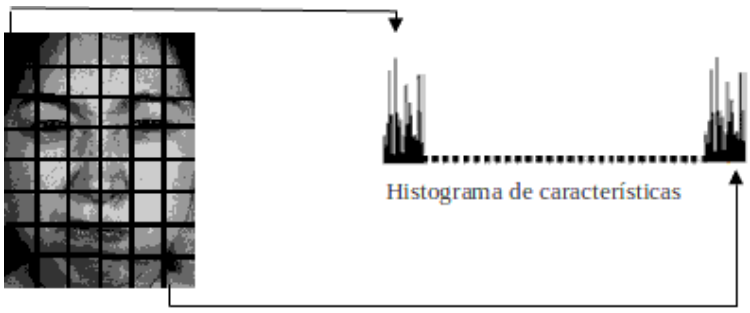


Figura 17: Imagem dividida em blocos e concatenação dos histogramas de cada bloco

Para reduzir o tamanho do vetor de características, os códigos LBP de cada pixel são somados e divididos pelo número de imagens do conjunto de dados, quando o resultado é menor do que um valor limiar os códigos desse pixel é excluído em todos os vetores (HADID, 2005). O limiar adotado nesse caso foi 5 (SHAN; GONG; MCOWAN, 2009). Resultando em vetores de tamanho 843, sendo composto por 842 características mais um dígito identificador do padrão (expressão).

O banco de imagens JAFFE é composto por 213 imagens, sendo divididas em 7 expressões: tristeza, alegria, desgosto, surpresa, raiva, medo e neutro. Na tabela 2 é apresentada a quantidade de imagens para cada expressão.

Expressão	Quantidade de imagens
Tristeza	31
Alegria	31
Desgosto	29
Surpresa	30
Raiva	30
Medo	32
Neutro	30

Tabela 2: Quantidade de imagens para cada expressão

Nos vetores de características de todos os conjuntos de dados utilizados, é acrescido um valor informando o padrão representada pelo vetor, para as etapas de treinamento e teste esse valor é omitido, ele só é utilizado na etapa de validação para verificar se o vetor foi corretamente classificado.

5.2 Organização dos dados

Em todos os testes foi o utilizado o método *k-fold cross validation* com k = 10, em seus trabalhos Baldisserotto (2005), Kohavi (1995), Dyer (2005) adotaram esse mesmo parâmetro. Os dados foram divididos em 10 subgrupos com mesmo tamanho (DYER, 2005) e com a mesma quantidade de vetores para cada classe, por isso não foram em utilizados todos os dados de todos os conjuntos de dados.

Conjuntos de dados			
Nome	Total de vetores	Vetores utilizados	Quantidade de padrões
Dígitos	10992	10500	10
LIBRAS	360	300	15
Íris	150	150	3
Expressões	213	210	7

Tabela 3: Organização dos dados

Na tabela 3 constam informações dos quatro conjuntos de dados utilizados nos experimentos nomeados da seguinte forma: Dígitos, LIBRAS, Íris, Expressões. Na segunda coluna são apresentados quantos vetores compõem o conjunto de dados no total sem distinguir o número de vetores por padrão. E na última coluna são apresentados o número de vetores uitlizados nos testes, em todos os casos o número de vetores utilizados é divisível por 10 por ser utilizado o parâmetro k = 10 no método *cross validation*.

No conjunto de dados Dígitos, os padrões que possuíam menos vetores eram as dos dígitos 3, 5, 8 e 9 com 1055 vetores, e as que possuíam mais vetores eram as dos dígitos 2 e 4 com 1144 vetores como mostrado na tabela 5.1. Nessa caso, foram utilizados 1050 vetores de cada padrão, totalizando 10500 vetores.

No conjunto de dados LIBRAS, cada padrão possui 24 vetores, para a distribuição dos vetores entre os 10 subgrupos foram utilizados 20 vetores de cada um dos 15 padrões, totalizando 300 vetores.

O conjunto de dados Iris possui 50 vetores para cada padrão, possibilitando a utilização dos 150 vetores na distribuição entre os 10 subgrupos.

No quarto conjunto de dados, Expressões, o padrão desgosto possui 29 vetores enquanto as classe restantes possuem 30 ou mais, como mostrado na tabela 2. Nesse caso, uma imagem do padrão desgosto foi repetida e utilizou-se 30 vetores

para cada padrão.

Para esse conjunto de dados Expressões, se vetores fossem excluídos em todas as classes, o número de vetores divisível por 10, que é a quantidade de subgrupos, mais próximo é 20, o que acarretaria a exclusão de 9 ou mais vetores por padrão, em contrapartida, na solução utilizada apenas 1 ou 2 vetores foram excluídas de cada padrão, com exceção do padrão desgosto que teve apenas uma imagem repetida. Já no conjunto de dados Dígitos optou-se pela exclusão, pois seria necessário repetir muitos vetores nos padrões com o número mais baixo de vetores, o mesmo ocorre com o conjunto de dados LIBRAS.

5.3 Resultados

Nesta seção são apresentados os resultados encontrados nos testes realizados com os quatro conjuntos de dados anteriormente descritos.

Na tabela a seguir são apresentados os resultados dos quatro conjuntos de dados após a validação *10-fold cross validation*, a taxa de acerto é uma média das taxas obtidas em cada um dos 10 ciclos de teste.

Conjunto de dados			Vetores por padrão			
Nome	Quantidade de características	Total de vetores	Total	No treinamento	Na classificação	Taxa de acerto (%)
Dígitos	16	10500	1050	945	105	98
LIBRAS	90	300	20	18	2	72
Íris	4	150	50	45	5	87
Expressões	842	210	30	27	3	34

Tabela 4: Resultados após a validação 10-fold cross validation

Na tabela 4 é apresentada a quantidade de características presente em cada vetor de cada conjunto da dados,a quantidade total de vetores, a quantidade de

vetores de cada classe utilizados para treinamento e a quantidade de vetores de cada classe utilizados para teste a cada ciclo da validação cruzada. Na última coluna é apresentada a taxa média de classificações corretas ao final da validação.

Nessa tabela é possível observar que as classe com menos características obtiveram as maiores taxas de acertos. Comparando os dois conjuntos com maiores taxas, observamos que o conjunto Dígitos com um número de vetores muito superior a do conjunto Íris obteve a maior taxa de acerto. O conjunto de dados Expressões, apesar de possuir uma quantidade de vetores superior a da classe LIBRAS, possui vetores com a quantidade de características muito superior aos vetores do conjunto LIBRAS e obteve uma taxa de acerto mais baixa.

Na tabela a seguir são apresentadas as taxas de acerto em cada ciclo de teste.

Conjunto de dados	Taxas de acertos por ciclo de teste (%)									
	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10
Dígitos	0,98	0,98	0,98	0,98	0,98	0,98	0,98	0,97	0,96	0,97
LIBRAS	0,4	0,77	0,74	0,7	0,63	0,7	0,63	0,83	0,93	0,9
Íris	0,74	0,8	0,93	0,87	0,93	0,74	0,87	0,87	0,93	0,87
Expressões	0,34	0,38	0,38	0,33	0,29	0,29	0,24	0,43	0,33	0,43

Tabela 5: Taxas de acerto em cada ciclo de teste

Na tabela 5 pode ser observado que nos dados Dígitos houve menos variação na taxa. Nos dados Íris houve certa variação entre 74% e 93%. Os dados LIBRAS e Expressões, que são os que possuem os maiores vetores apresentaram uma maior variação.

5.3.1 Quantidade de vetores X Taxa de acerto

Os conjuntos de dados são compostos por mais de um vetor de características de cada padrão, e a quantidade de vetores pode influenciar na taxa de acertos final.

Para fazer essa análise foi utilizado o banco de dados Dígitos por ser, entre os conjuntos utilizados, o que contém o maior número de vetores por padrão.

Conjunto de dados Dígitos			
Vetores por padrão			
Total	No treinamento	Na classificação	Taxa de acerto (%)
20	18	2	0,795
30	27	3	0,833
40	36	4	0,865
50	45	5	0.898
100	90	10	0.926
200	180	20	0.947
300	270	30	0.964
500	450	50	0.968
600	540	60	0.972
900	810	90	0.974
1000	900	100	0.975
1050	945	105	0.975

Tabela 6: Quantidade de Vetores X Taxa de acerto

Na primeira coluna da tabela 6 estão especificadas as quantidades de vetores de cada padrão e nas duas colunas seguintes as quantidades de vetores de cada padrão utilizados na etapa de treinamento e na etapa de testes respectivamente. Na última coluna estão especificadas as taxas de acertos em cada teste.

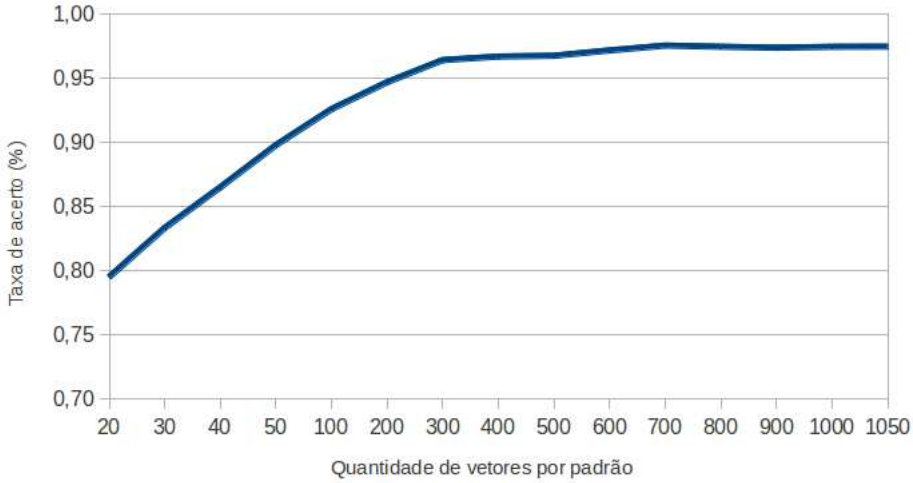


Figura 18: Gráfico quantidade de vetores X taxa de acerto

É possível observar um aumento na taxa de acerto a medida que a quantidade de vetores cresce. Porém esse aumento não é proporcional ao aumento do número de vetores. A variação na taxa de acerto diminui a medida que a quantidade de vetores aumenta, nos teste com 20, 30, 40 e 50 vetores por padrão houve uma variação média de 3,4% na taxa de acerto, do teste com 50 vetores por padrão para o teste com 100, houve uma variação de 2,8% na taxa de acerto. Já no testes com 200 e 500 a variação na taxa foi de 2%. E entre o teste com 600 vetores e o teste com quantidade máxima de vetores que é 1050 é observada uma variação bem pequena de 0,03% na taxa de acerto. Como pode ser observado no gráfico da figura 18 inicialmente há um aumento na taxa de acerto proporcional ao aumento na quantidade de vetores por padrão, mas há uma estabilização desse aumento na taxa de acerto a partir dos testes com 300 vetores por padrão.

5.3.2 Tempo computacional

A partir dos testes realizados também foram realizadas análises do ponto de vista computacional. Os resultados mostrados na tabela a seguir são as médias dos tempos de 5 execuções incluindo a etapa de validação compĺeta, com os dez ciclos

.

Conjuntos de dados				Tempo de execução (em segundos)		
Nome	Quantidade de padrões	Quantidade de hiperplanos	Vetores por padrão	Gerador de hiperplanos por ciclo	Gerador de hiperplanos total	Treinamento e Classificação
Dígitos	10	45	1050	2,18	21,88	30,53
Libras	15	105	20	0,65	6,47	7,87
Iris	3	3	50	0,04	0,41	0,65
Expressões	7	21	30	0,78	7,82	8,93

Tabela 7: Tempos computacionais

Nas colunas 5 e 6 da tabela 7 estão os tempos de execução apenas do módulo de geração de classificadores, que é o módulo onde o MPLHS é resolvido. Na sexta coluna é apresentado o tempo para geração dos classificadores considerando os 10 ciclos de treino e teste, é possível constatar um tempo computacional que corresponde a mais de 50% do tempo computacional total. A quantidade de classificadores gerados deve ser considerada. Na quita coluna é apresentado o tempo médio para geração dos classificadores a cada ciclo de teste e na terceira coluna da tabela são apresentadas as quantidades de classificadores gerados nesse tempo, para três conjuntos de dados esse tempo foi inferior a 1 segundo.

Na última coluna da tabela 7 são apresentados os tempos totais de execução de cada experimento. Pode-se constatar que esse tempo é proporcional ao tamanho dos vetores do conjunto de dados e a quantidade de classes. O conjunto de dados Íris obteve um tempo computacional menor que 1 segundo sendo um conjunto de dados com 3 classe e 50 vetores para cada padrão. Enquanto o conjunto de dados Dígitos obteve o maior tempo computacional apesar de possuir vetores com 16 características é o conjunto de dados com maior número de vetores por classe.

Comparando os tempos computacionais dos conjuntos Dígitos e Expressões é possível verificar que a quantidade de vetores por classe teve maior influência no tempo de execução do que o tamanho do vetor, já que a diferença dos tempos

de execução das duas classes é bastante alto, proporcional a as quantidades de vetores por classe, e inversamente proporcional ao tamanho dos vetores. O conjunto Dígitos possui vetores com 16 atributos, enquanto os vetores do conjunto Expressões possuem 842 atributos.

5.4 Análise dos resultados

Juntamente com o conjunto de dados Dígitos (ALPAYDIN; ALIMOGLU, 2013), estão disponíveis resultados de testes utilizando como método de classificação k nearest neighbor, a métrica utilizada para medir as distâncias entro os k vizinhos mais próximos foi a distância Euclidiana. O autor testou variando o parâmetro k de 1 à 11 com a taxa de acerto variando entre 97,34% e 97,80%. Em relação ao experimentos realizados neste trabalho é possível verificar que utilizando o método proposto foram obtidas taxas de acertos próximas as encontradas em Alpaydin e Alimoglu (2013).

Em seu trabalho Swain Sanjit Kumar Dash e Mohapatra (2012) abordou a classificação de padrões utilizando Redes Neurais e o conjunto de dados Íris. O autor dividiu 75 vetores para treinamento e 75 para teste. Foram realizados 3 testes, variando um parâmetro na configuração da rede neural. As taxas de acertos obtidas variaram entre 83,33% e 96,66%. No experimento utilizando o MPLHS a média de acertos obtida foi de 87%, um taxa que supera a obtida utilizando redes neurais, dependendo da configuração.

Em Hadid (2005) foi utilizado o conjunto de dados JAFFE (LYONS; GYOB, 1997). O MPLHS proposto em Bennett e Mangasarian (1992) é utilizado para gerar os classificadores e na classificação de imagens com expressão desconhecida foi utilizada uma árvore binária de torneio, assim como nos experimentos deste trabalho. A taxa de acerto foi de 93.8%.

No trabalho de Dyer (2005) também foi utilizado o banco JAFFE (LYONS; GYOB, 1997). Foram feitas comparações entre aluns métodos de classificação,

entre eles: Bayes com taxa de acertos 71,0%, Support Vector Machine linear e não linear, com taxa de acertos 92,4% e 91,9%, respectivamente e uma variação do modelo de programação linear utilizado neste trabalho, com taxa 91,0%. Em todos os casos foram utilizados mecanismos para seleção das características que seriam utilizadas

Tanto no trabalho de Hadid (2005) quanto no trabalho de Dyer (2005) foram obtidas taxas consideravelmente mais altas do que a obtida nos experimentos (34,0%). Essa diferença pode ser atribuída pela falta de pre processamento das imagens, como já comentado anteriormente.

6 Conclusão

A programação linear, apesar de possuir aplicações em diversas áreas, não é tão explorada na área da computação. O presente trabalho aprsentou a utilização de um modelo de programação linear no processo de classificação de dados. Como resultado, esse modelo gera hiperplanos separadores que são utlizados na tarefa de classificar um dado com padrão inicialmente desconhecido.

Inicialmente foi realizado um estudo sobre aplicações da programação linear e sobre métodos de resolução, princiaplmente sobre o método simplex revisado, um método desenvolvido para problemas de programação linear resolvidos computacionalmente. Também foram aprsentados alguns metódos de classificação de padrões e métodos de validação da metodologia.

Foram realizados experimentos utilizando quatro conjuntos de dados, três desses conjuntos foram obtidos já no formato de vetores de carcaterísticas. No caso do conjunto de dados Expressões, foram obtidas imagens e os vetores de características foram extraídos utilizando o método *Local Binary Patterns*. Nos experimentos, o MPLHS proposto por Bennett e Mangasarian (1992) foi utlizado, os padrões foram combinados em pares e os vetores de cada padrão submetidos ao modelo, gerando um hiperplano separador para cada combinação. Já na etapa de classificação foi utlizada uma estrtura de árvore binária de torneio, o vetor a ser classificado era subemtido a árvore e um padrão era retornado classificando esse vetor. Para a validação da metodologia o método *k -fold cross valdiation* foi utlizado devido a variação de tamanho das bases de dados utilizadas. Na implementação foi utlizada

a linguagem de programação JAVA e o software CPLEX da IBM.

Nos experimentos foram obtidas taxas de acerto altas para os casos da base de dados Dígitos e Íris, com taxas de acerto 98% e 87%, respectivamente. No caso da base de dados LIBRAS foi obtida uma taxa de acertos de 72%. Para a base de dados Expressões a taxa obtida foi de 34%, esse resultado pode ser atribuído ao vetor de caracteríticas, já que os vetores foram extraídos sem um pré processamento das imagens. Utilizando a base de dados Dígitos foi analisada a influência da quantidade de dados na melhoria da taxa de acertos, foi verificado que essa taxa pode aumentar a medida que a quantidade de dados por padrão aumenta, porém após uma certa quantidade de vetores por padrão, o aumento na taxa de acerto não é tão significante. Já no teste para análise do tempo computacional foi constatado que a quantidade de vetores por padrões e o tamanho dos vetores influenciam no tempo, porém a quantidade de vetores tem uma maior influencia no aumento do tempo computacional.

O presente trabalho expôs uma aplicação na programação linear na computação, mais especificamente na classificação de dados. Foram obtidos bons resultados em alguns experimentos, e em alguns casos, superiores ao resultados obtidos em outros trabalhos que utilizaram a mesma base de dados e um método de classificação diferente. Em outros experimentos os resultados obtidos não foram tão satisfatórios.

Como trabalho futuro propõe-se a melhoria na taxa de acertos, principalmente da base de dados Expressões. Para isso deve haver um estudo mais aprofundado na área da visão computacional, submetendo as imagens a um pre processamento e utilizando um método mais apurado para selecionar as características e reduzir o tamanho do vetor de características.

Um outra melhoria é o tempo computacional, que poderia ocorrer através de um estudo sobre as melhores estrturas de dados a serem utlizadas.

Referências Bibliográficas

ALPAYDIN, E.; ALIMOGLU, F. *Pen-Based Recognition of Handwritten Digits Data Set*. 2013. Disponível em: <http://archive.ics.uci.edu/ml/datasets-/Pen-Based+Recognition+of+Handwritten+Digits>.

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

BALDISSEROTTO, C. *Técnicas de aprendizagem de máquina para previsão de sucesso em implantes dentários*. Dissertação (Mestrado) — Universidade de Pernambuco, 2005.

BENNETT, K. P.; MANGASARIAN, O. L. *Robust Linear Programming Discrimination Of Two Linearly Inseparable Sets*. 1992.

BIXBY, R. E. Implementing the simplex method: The initial basis. *ORSA Journal*, v. 4, p. 267–284, 1992.

DANZITG, G. *Linear Programming and Extensions*. [S.l.]: Princeton University Press, 1963.

DIAS, S. M. P. D. B.; BíSCARO, H. H. *Libras Movement Data Set*. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Libras+Movement>.

DYER, G. G. C. R. Simultaneous Feature Selection and Classifier Training via Linear Programming: A Case Study for Face Expression Recognition. *IEEE Transactions on systems, man, and cybernetics*, v. 35, n. 3, p. 477–488, Junho 2005.

FISHER, R. *Iris Data Set*. 1936. Disponível em: <http://archive.ics.uci.edu/ml-/datasets/Iris>.

GOLDBERG, R. A. E. L. J. P. I.-C. J. H. J. F. O. K. Optimization of HDR brachytherapy dose distributions using linear programming with penalty costs. *The International Journal of Medical Physics Research and Practice*, v. 33, n. 11, 2006.

GUNN, S. R. *Support Vector Machines for Classification and Regression*. 1998.

HADID, X. F. M. P. A. Facial Expression Recognition with Local Binary Patterns and Linear Programming. *Pattern Recognition and Image Analysis*, v. 15, n. 2, p. 546–548, 2005.

HILLIER, F.; LIEBERMAN, G. *Introdução à Pesquisa Operacional*. 8º. ed. [S.l.]: Mc-Graw-Hill, 2006.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: . [S.l.]: Morgan Kaufmann, 1995. p. 1137–1143.

KRUKOSKI, F. A. *Programação linear aplicada ao mercado de opções na criação de um portifólio seguro*. Dissertação (Mestrado) — Universidade Federal do Paraná, 2010.

LANGLEY, W. I. P.; THOMPSON, K. An analysis of bayesian classifiers. In: *IN PROCEEDINGS OF THE TENTH NATIONAL CONFERENCE ON ARTI CIAL INTELLIGENCE*. [S.l.]: MIT Press, 1992. p. 223–228.

LIMA, A. R. G. *Máquinas de Vetores Suporte na Classificaçã de Impressões Digitais*. Dissertação (Mestrado) — Universidade Federal do Ceará, 2002.

LORENA, A. C. P. L. F. d. C. A. C. *Introdução às Máquinas de Vetores Suporte*. 2003.

LYONS, M. K. M. J.; GYOB, J. *Japanese Female Facial Expressions (JAFFE), Database of digital images*. 1997.

MACULAN, N.; FAMPA, M. H. C. *Otimização Linear*. [S.l.]: Editora Universidade de Brasília, 2006.

MCCALLUM, A.; NIGAM, K. A comparison of event models for naive bayes text classification. In: *IN AAAI-98 WORKSHOP ON LEARNING FOR TEXT CATEGORIZATION*. [S.l.]: AAAI Press, 1998. p. 41–48.

MENEZES, L. de L. P. M. A. F. Implementação de algoritmos simplex e pontos interiores para programação linear. *Estudos*, v. 15, n. 2, p. 225–246, 2008.

MORAIS, E. C. *Reconhecimento de padrões e Redes Neurais Artificiais em predição de estruturas secundárias de proteínas*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010.

MOREIRA, F. R. Programação linear aplicada a problemas da área de saúde. *Einstein*, v. 105, n. 1, 2003.

MUNARI, J. P. A. *Técnicas Computacionais para uma implementação eficiente e estável de metódos tipo smplex*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação, ICMP-USP, 2009.

PAMPLONA, E. de O. *Engenharia Econômica II*. 2005. Disponível em: <http:-//www.iepg.unifei.edu.br/edson/download/Engecon2/CAP5EE2PLapost.pdf>. Acesso em: 04/06/2012.

PASSOS, A. N. *Estudos em Programação Linear*. Dissertação (Mestrado) — Instituto de Matemática, Estatística e Computação Científica, UNICAMP, 2009.

POSSAMAI, J. P.; PESCADOR, A. Transporte e estocagem de fumo - Um modelo de programação linear usado na tomada de decisão. *XXXIX Congresso Brasileiro de Educação em Engenharia*, 2011.

SANTOS, F. C. *Variações do método kNN e suas aplicações na classificação automática de textos*. Dissertação (Mestrado) — Universidade Federal de Goiás, 2009.

SHAN, C.; GONG, S.; MCOWAN, P. W. Facial expression recognition based on local binary patterns: A comprehensive study. *Image Vision Comput.*, v. 27, n. 6, p. 803–816, 2009.

SIMõES, A. d. S.; COSTA, A. H. R. Classificação de laranjas baseada em padrões visuais de alto nível. In: *6o. Simpósio Brasileiro de Automação Inteligente - SBAI/SBA, 2003, Bauru, SP. Anais do VI Simpósio Brasileiro de Automação Inteligente*. [S.l.: s.n.], 2003. p. 77–81.

SWAIN SANJIT KUMAR DASH, S. D. M.; MOHAPATRA, A. An approach for iris palnt classification using neural network. *International Journal on Soft Computing*, v. 3, n. 1, 2012.

TODD, M. J. The many facets of linear programming. *Mathematical Programming*, v. 91, p. 417–436, 2002.

TREVISAN, E. P. *O uso da programação linear na separação de pontos*. Dissertação (Mestrado) — Universidade Estadual de Campinas - UNICAMP, 2010.

ZUBEN, F. V.; CASTRO, L. N. de. *Redes Neurais Artificiais para Reconhecimento e Classificação de Padrões*. 2003.

APÊNDICE A – Código JAVA do MPLHS

```
1  package gerador_de_classificadores;
2
3  import ilog.concert.*;
4  import ilog.cplex.*;
5  import java.io.*;
6  import java.util.regex.*;
7
8  public class GeradorDeClassificador {
9
10     static int m;
11     static int k;
12     static int[] e1;
13     static int[] e2;
14     static double[][] A;
15     static double[][] B;
16     static int quantidade_de_padroes = 10;
17
18     static void readData(String filename1, String filename2) throws IloException,
19             IOException, InputDataReader.InputDataReaderException {
20
21         InputDataReader reader1 = new InputDataReader(filename1);
22         m = reader1.readInt();
23         e1 = reader1.readIntArray();
24         A = reader1.readDoubleArrayArray();
25
```

```
26          InputDataReader reader2 = new InputDataReader(filename2);
27          k = reader2.readInt();
28          e2 = reader2.readIntArray();
29          B = reader2.readDoubleArrayArray();
30      }
31
32      public static void gerador() {
33          long tempoInicial = System.currentTimeMillis();
34          try {
35              FileWriter file = new FileWriter("../Arquivos/Classificadores.txt");
36
37              for (int i = 1; i <= quantidade_de_padroes; i++) {
38                  for (int j = i + 1; j <= quantidade_de_padroes; j++) {
39                      String filename1 = "../Arquivos/Dados_Padrao0.dat";
40                      String filename2 = "../Arquivos/Dados_Padrao0.dat";
41                      filename1 = filename1.replaceAll("[0-9]", "" + i);
42                      filename2 = filename2.replaceAll("[0-9]", "" + j);
43
44                      file.write(i + "-" + j + "\n");
45
46                      readData(filename1, filename2);
47
48                      IloCplex cplex = new IloCplex();
49
50                      //VARAVEIS DO MODELO
51                      IloNumVar[] omega = cplex.numVarArray(A[0].length,
52                              Double.NEGATIVE_INFINITY, Double.POSITIVE_INFINITY);
53                      IloNumVar gamma = cplex.numVar(Double.NEGATIVE_INFINITY,
54                              Double.POSITIVE_INFINITY);
55                      IloNumVar[] y = cplex.numVarArray(m, 0.0,
56                              Double.POSITIVE_INFINITY);
57                      IloNumVar[] z = cplex.numVarArray(k, 0.0,
58                              Double.POSITIVE_INFINITY);
59
60                      //FUNCAO OBJETIVO
61                      cplex.addMinimize(cplex.sum(cplex.prod(cplex.scalProd(e1, y), 1.0 / m),
```

```
62                          cplex.prod(cplex.scalProd(e2, z), 1.0 / k)));
63
64                  //RESTRICAO -A*omega + gamma*e1 + e1 <= y
65                  //multiplicando a matriz A por -1
66                  for (int m = 0; m < A.length; m++) {
67                      for (int n = 0; n < A[0].length; n++) {
68                          A[m][n] = A[m][n] * -1.0;
69                      }
70                  }
71                  for (int m = 0; m < A.length; m++) {
72                      cplex.addLe(cplex.sum(cplex.sum(cplex.scalProd(A[m], omega),
73                              gamma), 1), y[m]);
74                  }
75
76                  //RESTRICAO B*omega - gamma*e2 + e2 <= z
77                  for (int l = 0; l < B.length; l++) {
78                      cplex.addLe(cplex.sum(cplex.sum(cplex.scalProd(B[l], omega),
79                              cplex.negative(gamma)), 1), z[l]);
80                  }
81
82                  if (cplex.solve()) {
83                      if (cplex.getObjValue() != 0) {
84                      }
85                      String valor_gamma;
86                      valor_gamma = "" + cplex.getValue(gamma);
87                      file.write(valor_gamma);
88                      file.write("\n");
89                      for (int k = 0; k < omega.length; k++) {
90                          String valor_omega;
91                          valor_omega = "" + cplex.getValue(omega[k]) + ",";
92                          file.write(valor_omega);
93                      }
94                      file.write("\n");
95                  }
96                  cplex.end();
97
```

```
98                  }
99              }
100
101             file.close();
102         } catch (IloException e) {
103             System.err.println("Concert exception caught: " + e);
104         } catch (IOException ex) {
105             System.err.println("Concert exception caught: " + ex);
106         } catch (InputDataReader.InputDataReaderException ex) {
107             System.err.println(ex);
108         }
109         long tempoFinal = System.currentTimeMillis();
110         System.out.println("Gerador: " + (tempoFinal - tempoInicial));
111
112     }
113 }
```