Universidade Federal de Juiz de Fora
Isntituto de Ciências Exatas / Faculdade de Engenharia
Programa de Pós-graduação em Modelagem Computacional

Maurício Archanjo Nunes Coelho

Uma abordagem de predição estruturada baseada no modelo perceptron

Juiz de Fora
2015

Universidade Federal de Juiz de Fora
Isntituto de Ciências Exatas / Faculdade de Engenharia
Programa de Pós-graduação em Modelagem Computacional

Maurício Archanjo Nunes Coelho

Uma abordagem de predição estruturada baseada no modelo perceptron

Juiz de Fora
2015

Maurício Archanjo Nunes Coelho

Uma abordagem de predição estruturada baseada no modelo perceptron

Tese apresentada ao Programa de Pós-graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, na área de concentração Modelagem Computacional, como requisito parcial para obtenção do título de Doutor em Modelagem computacional.

Orientador: Dr. Carlos Cristiano Hasenclever Borges

Coorientador: Dr. Raul Fonseca Neto

Juiz de Fora

2015

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Coelho, Maurício Archanjo Nunes.
Uma Abordagem de Predição Estruturada Baseada no Modelo Perceptron / Maurício Archanjo Nunes Coelho. -- 2015.
136 p. : il.

Orientador: Carlos Cristiano Hasenclever Borges
Coorientador: Raul Fonseca Neto
Tese (doutorado) - Universidade Federal de Juiz de Fora, ICE/Engenharia. Programa de Pós-Graduação em Modelagem Computacional, 2015.

1. Predição Estruturada. 2. Perceptron. 3. Planejamento de Caminhos. 4. Aprendizado de Máquina. I. Borges, Carlos Cristiano Hasenclever, orient. II. Fonseca Neto, Raul, coorient. III. Título.

Resumo

A teoria sobre aprendizado supervisionado tem avançado significativamente nas últimas décadas. Diversos métodos são largamente utilizados para resoluções dos mais variados problemas, citando alguns: sistemas especialistas para obter respostas to tipo verdadeiro/falso, o modelo *Perceptron* para separação de classes, Máquina de Vetores Suportes (SVMs) e o Algoritmo de Margem Incremental (IMA) no intuito de aumentar a margem de separação, suas versões multi-classe, bem como as redes neurais artificiais, que apresentam possibilidades de entradas relativamente complexas. Porém, como resolver tarefas que exigem respostas tão complexas quanto as perguntas?

Tais respostas podem consistir em várias decisões inter-relacionadas que devem ser ponderadas uma a uma para se chegar a uma solução satisfatória e globalmente consistente. Será visto no decorrer do trabalho que existem problemas de relevante interesse que apresentam estes requisitos.

Uma questão que naturalmente surge é a necessidade de se lidar com a explosão combinatória das possíveis soluções. Uma alternativa encontrada apresenta-se através da construção de modelos que compactam e capturam determinadas propriedades estruturais do problema: correlações sequenciais, restrições temporais, espaciais, etc. Tais modelos, chamados de estruturados, incluem, entre outros, modelos gráficos, tais como redes de *Markov* e problemas de otimização combinatória, como *matchings* ponderados, cortes de grafos e agrupamentos de dados com padrões de similaridade e correlação.

Este trabalho formula, apresenta e discute estratégias *on-line* eficientes para predição estruturada baseadas no princípio de separação de classes derivados do modelo *Perceptron* e define um conjunto de algoritmos de aprendizado supervisionado eficientes quando comparados com outras abordagens.

São também realizadas e descritas duas aplicações experimentais a saber: inferência dos custos das diversas características relevantes para a realização de buscas em mapas variados e a inferência dos parâmetros geradores dos grafos de *Markov*. Estas aplicações têm caráter prático, enfatizando a importância da abordagem proposta.

Palavras-chaves: Aprendizado de Máquina. Predição de Dados Estruturados. Perceptron Multi-Classe. Planejamento de Caminhos. Grafos de *Markov*.

Abstract

The theory of supervised learning has significantly advanced in recent decades. Several methods are widely used for solutions of many problems, such as expert systems for answers to true/false, Support Vector Machine (SVM) and Incremental Margin Algorithm (IMA). In order to increase the margin of separation, as well as its multi-class versions, in addition to the artificial neural networks which allow complex input data. But how to solve tasks that require answers as complex as the questions? Such responses may consist of several interrelated decisions to be considered one by one to arrive at a satisfactory and globally consistent solution. Will be seen throughout the thesis, that there are problems of relevant interest represented by these requirements.

One question that naturally arises is the need to deal with the exponential explosion of possible answers. As a alternative, we have found through the construction of models that compress and capture certain structural properties of the problem: sequential correlations, temporal constraints, space, etc. These structured models include, among others, graphical models, such as *Markov* networks and combinatorial optimization problems, such as weighted matchings, graph cuts and data clusters with similarity and correlation patterns.

This thesis formulates, presents and discusses efficient online strategies for structured prediction based on the principle of separation of classes, derived from the Perceptron and defines a set of efficient supervised learning algorithms compared to other approaches.

Also are performed and described two experimental applications: the costs prediction of relevant features on maps and the prediction of the probabilistic parameters for the generating *Markov* graphs. These applications emphasize the importance of the proposed approach.

Key-words: Machine Learning. Perceptron Multi-class. Path Planning. Prediction of Structured Data. Markov Graphs.

Lista de ilustrações

Figura 1 – Modelo de McCulloch-Pitts de um neurônio artificial 17
Figura 2 – Uma correção durante a execução do algoritmo Perceptron para $\eta = 1$. 19
Figura 3 – Interpretação geométrica da margem γ_f (LEITE; NETO, 2007) 21
Figura 4 – A função Φ explicitamente definida num mapeamento $\mathbb{R}^2 \to \mathbb{R}^3$ 24
Figura 5 – Interpretação geométrica de $\gamma_{y_i,y}$ para o caso simples, onde $f(x_i, y)$ retorna coordenadas no plano cartesiano. 33
Figura 6 – Interpretação geométrica de γ_i para o caso simples, onde $f(x_i, y)$ retorna coordenadas no plano cartesiano. 34
Figura 7 – Interpretação geométrica da correção do vetor w para um caso simples, onde o vetor d_i possui somente duas dimensões. 40
Figura 8 – Interpretação geométrica de γ_i para um caso simples, onde $f(y)$ retorna coordenadas no plano cartesiano. 41
Figura 9 – Interpretação geométrica da correção do vetor w para um caso simples, onde o vetor d_i possui somente duas dimensões. 45
Figura 10 – Representação gráfica das matrizes do mapa, caminho e vetor de custos 60
Figura 11 – No primeiro quadro tem-se somente trilha; no segundo, somente rocha; no terceiro: trilha e rocha; no quarto: trilha e vegetação; no quinto: rocha e vegetação; no sexto: trilha, rocha e vegetação (trilha abandonada); no sétimo, somente vegetação e, finalmente, no oitavo tem-se a ausência de características. 65
Figura 12 – Mapas de treinamento 5×5 com seus respectivos caminhos traçados pelo especialista do domínio. 66
Figura 13 – Mapas de treinamento 10×10 com seus respectivos caminhos traçados pelo especialista do domínio. 72
Figura 14 – Mapas de testes 5×5 com os custos já associados a cada característica e todos os caminhos traçados pelo algoritmo A* de acordo com o vetor w associado. 74
Figura 15 – Mapas de testes 5×5 e os caminhos traçados pelo algoritmo A* usando o vetor w para o primal e para o dual. 75
Figura 16 – Mapas de testes 10×10 com os custos já associados a cada característica e os caminhos traçados pelo algoritmo A* 76
Figura 17 – Mapas de treinamento com seus respectivos caminhos escolhidos pelo especialista. *Google maps* foram discretizados com dimensão 55×55 e simplificados para abarcar os oito diferentes tipos de terreno em cada célula. 78

Figura 18 – Mapas de teste com os caminhos definidos pelo algoritmo A* com base no vetor de custos w. 80
Figura 19 – Diferentes intensidades entre as três características. Os valores de intensidades são, nesta ordem: (Rocha | Vegetação | Caminho). 81
Figura 20 – Mapas com seus respectivos caminhos escolhidos por um especialista. . 82
Figura 21 – Conjunto de caminhos possíveis μ cujo custo em F_i foi calculado de acordo com a equação 5.26. 83
Figura 22 – Mapas de testes com seus respectivos melhores caminhos escolhidos entre os apresentados na Figura 21. 83
Figura 23 – Matrizes de adjacência M_{x_B} e M_{y_A} correspondentes aos grafos B e A. . 90
Figura 24 – Matriz de adjacência M_{x_E} e M_{y_E} correspondente ao grafo E. 93
Figura 25 – Matriz de adjacência M_{x_F} correspondente ao grafo F. 94
Figura 26 – Exemplo de matriz resultante do processo de aprendizado de F em A. . 94
Figura 27 – Matriz de adjacência correspondente ao grafo C. 96
Figura 28 – Exemplo de um grafo com custos nas arestas 107
Figura 29 – Esquema de uma expansão de nó em uma busca A* 113
Figura 30 – Condição para uma heurística consistente 113
Figura 31 – Relações primal-dual 115
Figura 32 – Gráfico de uma função convexa 128
Figura 33 – Subderivadas de uma função convexa 129
Figura 34 – Exemplos de alguns subgradietes 130

Lista de tabelas

Tabela 1 – *Custos geométricos do algoritmo MMP.* 68
Tabela 2 – *Custos geométricos do algoritmo MMP com 1000 iterações.* 68
Tabela 3 – *Custos geométricos do algoritmo Perceptron Estruturado.* 68
Tabela 4 – *Custos geométricos do algoritmo Perceptron Estruturado com Margem.* 69
Tabela 5 – *Resultados do treinamento para o Perceptron Estruturado Primal com Margem Incremental.* 70
Tabela 6 – *Resultados do treinamento para o Perceptron Estruturado Dual com Margem Incremental.* 70
Tabela 7 – *Valores de w calculados através dos valores de α da Tabela 6.* 71
Tabela 8 – *Custos geométricos do algoritmo MMP.* 71
Tabela 9 – *Custos geométricos do algoritmo MMP depois de 1000 iterações.* 72
Tabela 10 – *Custos geométricos do algoritmo Perceptron Estruturado.* 73
Tabela 11 – *Custos geométricos do algoritmo Perceptron Estruturado com Margem.* 73
Tabela 12 – *Custos geométricos do algoritmo MMP.* 79
Tabela 13 – *Custos geométricos do algoritmo Perceptron Estruturado.* 79
Tabela 14 – *Custos geométricos do algoritmo Perceptron Estruturado com margem Incremental.* 79
Tabela 15 – *Valores de margens usando um Kernel quadrático.* 82
Tabela 16 – *Valores de margens usando um Kernel cúbico.* 82

Lista de abreviaturas e siglas

SVM	*Support Vector Machine*.
IMA	*Incremental Margin Algorithm* proposto por (LEITE; NETO, 2007).
RNA	Rede Neural Artificial.
MMP	Algoritmo *Maximum Margin Planning* proposto por (RATLIFF; BAGNELL; ZINKEVICH, 2006).
PMF	Perceptron de Margem Fixa.

Lista de símbolos

V	Conjunto de vértices ou nós.
E	Conjunto de arestas ou arcos.
G	Grafo, par ordenado (V, E).
$\|\| \ \|\|$	Norma do vetor a ser definida.
$\|\| \ \|\|_2$	Norma euclidiana de um vetor.
$\langle \ , \ \rangle$	Produto interno.
l_i	Função de perda relacionada a diferença entre algum elemento i do conjunto de treinamento e o calculado atual.
$argMax$	Argumento que maximiza uma função.
$argMin$	Argumento que minimiza uma função.
x_i	Entrada estruturada x_i, pertencente ao conjunto de treinamento, par do elemento y_i.
y_i	Saída estruturada y_i, pertencente ao conjunto de treinamento, par do elemento x_i.
S	Conjunto de treinamento $S = \{(x_i, y_i),\ i = 1, \ldots, m\}$ para problemas estruturados.
Z	Conjunto de treinamento $Z = \{(x_i, y_i)\}_{i=1}^{m},\ (x_i, y_i) \in X \times Y$ para problemas com saída binária, ou seja, onde $Y = \{-1, +1\}$.
Y	Conjunto $Y = \bigcup_{x \in X} Y(x)$ de todas as saídas y possíveis considerando todas as entradas x.
Y_i	Conjunto de todas as saídas y dependente do objeto estruturado x_i.
Y_S	Conjunto $Y_S = \{y_i \in S,\ \forall i\}$ de todas as saídas y_i do conjunto de trinamento S.
$f(y_i) = f(x_i, y_i)$	Função que correlaciona duas estruturas, comumente multiplicação matricial.
y^*	Representa a saída estruturada ótima, tanto para problemas de maximização quanto minimização.

$f(y^*) = f(x_i, y^*)$	Função que correlaciona x_i com sua saída ótima y^*.
w	Vetor de pesos associado principalmente a alguma estrutura.
w^T	O vetor de pesos w transposto, para cálculo matricial.
F_i	Matriz correspondente ao elemento x_i, relacionada ao problema de predição de custos em planejamentos de caminhos.
μ	Matriz correspondente ao elemento y_i relacionada ao caminho no problema de predição de custos.
ν	Matriz correspondente ao elemento y_i relacionada ao caminho no problema de predição de custos, usada na formulação dual para diferenciar do caminho μ.
δ	Incremento mínimo de margem γ usado no algoritmo *IMA*.
Δ	Passo fixo de incremento de uma margem γ usado na abordagem estruturada.
d_i	Vetor diferença que representa a operação: $f_i(y^*) - f_i(y_i),\ y \in Y_S,\ \forall i$.
ϱ	Multiplicadores de lagrange.
Υ	Margem de erro Υ cujo módulo de $\gamma_f^+ - \gamma_f^- < \Upsilon$
γ_f	Margem fixa usada no IMA.
γ	Margem fixa estipulada para a execução do algoritmo *Perceptron Estruturado com Margem*.
γ_i	Margem de $i \in S$ em relação aos outros elementos de S.
γ_z	Margem final se comparado todas as margens de todos o conjunto S: $\gamma_z = Min\gamma_i, \forall i$.
α_i	$\alpha_i \in \mathbb{R}$ é a variável dual.
$J()$	*Joint Kernel*.
$\|\ \|$	Cardinalidade de um conjunto.
M_{x_i}	Matriz de entrada relacionada ao problema de predição de grafos de *Markov*.
M_{y_i}	Matriz de saída relacionada ao problema de predição de grafos de *Markov*.

G_0	Grafo inicial relacionado ao problema de predição de grafos de *Markov*.
G_f	Grafo final relacionado ao problema de predição de grafos de *Markov*.
p	Um dos dois parâmetros de entrada para predição de grafos de *Markov*.
q	Um dos dois parâmetros de entrada para predição de grafos de *Markov*.
ε	Medida de erro absoluto durante a geração de grafos de *Markov*.
ε_r	Medida de erro relativo, calculado em porcentagem, durante a geração de grafos de *Markov*.
ρ	Probabilidade de um evento ocorrer.
α	Quantidade de grafos de *Markov* gerados até atingir β.
β	Um número mínimo de acertos, previamente definido, necessários ao aprendizado, a fim de medir a eficiência do processo.
Ip_a	Intervalo dos valores do parâmetros p, onde ocorrem igualdade entre os grafos.
Ip_t	Intervalo dos valores do parâmetros p, considerando todos os grafos.
Iq_a	Intervalo dos valores do parâmetros q, onde ocorrem igualdade entre as matrizes.
Iq_t	Intervalo dos valores do parâmetros q, considerando todos os grafos.

Sumário

	Introdução	**14**
0.1	**Contexto e Motivação**	**14**
0.2	**Objetivos**	**15**
0.3	**Organização do Trabalho**	**15**
1	**NOÇÕES INTRODUTÓRIAS**	**17**
1.1	**Perceptron**	**17**
1.2	**Perceptron de Margem Fixa - PMF**	**20**
1.3	**Perceptron com Margem Incremental - *IMA***	**22**
1.4	**Perceptron Dual**	**23**
1.5	**Método *Kernel***	**24**
2	**MODELOS DE PREDIÇÃO DE DADOS ESTRUTURADOS**	**28**
2.1	**Modelos de Predição Estruturados**	**28**
2.1.1	Modelos Lineares para Predição Estruturada	29
3	**PREDIÇÃO ESTRUTURADA E FUNÇÕES DE RESTRIÇÃO**	**32**
3.1	**Predição Estruturada no Aprendizado Supervisionado**	**32**
3.2	**Formulação de Máxima Margem**	**36**
4	**TÉCNICAS DE SOLUÇÃO**	**39**
4.1	**Perceptron Estruturado**	**39**
4.2	**Perceptron Estruturado com Margem Zero**	**43**
4.3	**Perceptron Estruturado com Margem**	**44**
4.4	**Perceptron Estruturado com Margem Incremental**	**47**
4.5	**O Método de Subgradiente**	**48**
4.6	**Perceptron Estruturado Dual**	**49**
4.7	**Perceptron Estruturado Dual com *Kernel***	**53**
5	**PREDIÇÃO DE DADOS ESTRUTURADOS EM PLANEJAMENTO DE CAMINHOS**	**58**
5.1	**Proposta da Aplicação**	**58**
5.2	**Problema de Predição de Custos**	**59**
5.2.1	Equacionamento do Problema de Predição de Custos	59
5.3	**Método do Subgradiente Aplicado na Predição de Custos - *MMP***	**61**
5.4	**Métodos Baseados no Perceptron Aplicado na Predição de Custos**	**61**
5.5	**Resultados Experimentais em Mapas Artificiais**	**65**

5.5.1 Resultados do Conjunto de Treinamento . . . 66
5.5.1.1 Exemplo 1 . . . 66
5.5.1.2 Exemplo 2 . . . 70
5.5.2 Resultados do Conjunto de Teste . . . 73
5.5.2.1 Exemplo 1 . . . 73
5.5.2.2 Exemplo 2 . . . 74
5.5.3 Análise dos Resultados . . . 76
5.6 Resultados Experimentais em Mapas Reais . . . 77
5.6.1 Resultados do Conjunto de Treinamento . . . 78
5.6.2 Resultados do Conjunto de Teste . . . 79
5.6.3 Análise dos Resultados . . . 80
5.7 Conclusão dos Experimentos para Problemas Linearmente Separáveis 80
5.8 Resultados Experimentais em Mapas Artificiais Não-Linearmente Separáveis . . . 81
5.8.1 Resultados do Conjunto de Treinamento . . . 81
5.8.2 Resultados do conjunto de teste . . . 82
5.8.3 Análise dos Resultados . . . 83

6 ESTRATÉGIA ON-LINE PARA PREDIÇÃO DE DADOS ESTRUTURADOS EM GRAFOS DE *MARKOV* . . . 85
6.1 Introdução . . . 85
6.2 Formulação Teórica . . . 86
6.3 Simulação do Grafo de *Markov* . . . 88
6.4 Experimentos e Resultados . . . 90
6.4.1 Aprendizado de um Grafo Menos Denso para um Grafo Mais Denso . . . 90
6.4.2 Aprendizado de um Grafo Mais Denso para um Menos Denso . . . 92
6.4.3 Aprendizado para um Grafo Manter-se Estável Utilizando o Processo de Formação *Markoviano* . . . 93
6.4.4 Estipulando uma Topologia Fixa para o Aprendizado . . . 93
6.4.5 Testando a Escalabilidade do Algoritmo . . . 94
6.4.6 Abordagem Estruturada Mista . . . 95
6.5 Considerações . . . 96

7 CONCLUSÕES E TRABALHOS FUTUROS . . . 98

REFERÊNCIAS . . . 100

APÊNDICES 105

APÊNDICE A – BUSCA DE CAMINHOS . . . 106

A.1 Planejamento de Caminhos . . . 106
A.2 Otimização Combinatória e Problema do Caminho Mínimo . . . 106
A.3 Resolução de Problemas por meio da Busca . . . 109
A.4 Medição de Desempenho da Busca . . . 110

APÊNDICE B – DETERMINAÇÃO DE CAMINHOS . . . 111
B.1 Busca *Forward* . . . 111
B.1.1 Algoritmo de Dijkstra . . . 111
B.1.2 Busca A* . . . 112
B.2 Solução *Backward* . . . 114
B.2.1 Conversão Primal-Dual . . . 114
B.2.2 Solução do Problema Dual . . . 116
B.2.3 Equações de *Bellman* . . . 116
B.2.4 Algoritmo de *Bellman-Ford* . . . 118

APÊNDICE C – APRENDIZADO POR CORREÇÃO DE ERROS . 121

APÊNDICE D – OTIMIZAÇÃO . . . 123
D.1 Otimização Não-Linear . . . 123
D.2 Otimização Convexa . . . 123
D.3 Multiplicadores de Lagrange . . . 124
D.4 Programação Quadrática . . . 126

APÊNDICE E – SUBDIFERENCIAIS E SUBGRADIENTES . . . 128
E.1 Função Convexa . . . 128
E.2 Subderivada e Subdiferencial . . . 129
E.3 Subgradiente . . . 130

APÊNDICE F – PROBLEMA INVERSO . . . 133

APÊNDICE G – MMP BOOST E SMO ESTRUTURADO . . . 134

Introdução

0.1 Contexto e Motivação

Problemas estruturados levam em consideração as interdependências entre elementos individuais que compõem diferentes estruturas (definida formalmente no capítulo 2), bem como estas estruturas interagem entre si. Deste modo, o aprendizado ocorre entre um conjunto de pares de estruturas; uma estrutura de entrada A que se deseja obter os parâmetros c e outra B, derivada da primeira a partir de uma fórmula F. A relação toma a forma $B = F(A(c))$. Os valores calculados durante o processo de aprendizado são os parâmetros c correlacionados e condizentes com o conjunto de estruturas avaliadas. Ao se aplicar esses parâmetros novamente na relação acima gerará a estrutura B e ao se aplicar c em novas estruturas quaisquer D, E, F, ..., certas características gerais serão preservadas. Observe que é uma formulação típica de problemas inversos, conforme breve resumo pode ser visto no apêndice F. Porém, as abordagens apresentadas nesta tese não se limitam somente a este caso mais simples, tendo em diversos problemas uma dificuldade adicional, pois um mesmo vetor de parâmetros c deve ser compatível, ao mesmo tempo, para diferentes estruturas de entrada e saída.

O fato de se conseguir calcular os parâmetros armazenando implicitamente as informações de como elementos específicos de uma estrutura interagem entre si, conjuntamente com a comparação entre estas estruturas e não somente a análise de cada uma individualmente, é a grande vantagem desta abordagem.

Os três principais trabalhos de outros autores relacionados são: (TASKAR, 2004), no qual as bases teóricas da Predição Estruturada podem ser encontradas, (RATLIFF, 2009), no qual sua abordagem de predição foi comparada à desenvolvida na primeira aplicação deste trabalho e o livro de (BAKIR et al., 2007), que contém uma boa introdução juntamente com um apanhado de vários artigos da área de predição estruturada.

Problemas de predição estruturados surgem naturalmente em muitas tarefas onde múltiplas decisões inter-relacionadas devem ser medidas e comparadas, relacionando-as umas as outras de modo a alcançar uma solução global satisfatória e consistente. É uma área recente na qual diversas técnicas de soluções estão sendo simultaneamente propostas (BAKIR et al., 2007). Este trabalho estuda a teoria do modelo geral de predição de dados estruturados e propõe novas técnicas de solução. Para isso, foram desenvolvidos e testados diferentes algoritmos de predição estruturada derivados do algoritmo *Perceptron*. Para comprovar sua eficácia, duas aplicações foram desenvolvidas, cada uma com exemplos variados.

0.2 Objetivos

O objetivo geral é apresentar e discutir as novas formulações para técnicas de solução desenvolvidas neste trabalho, baseadas no algoritmo *Percetron*: *Perceptron Estruturado*, *Perceptron Estruturado com Margem*, *Perceptron estruturado com Margem Incremental*, *Perceptron Estuturado Dual* e *Perceptron Estruturado Dual com Kernel*. Posteriormente, testar sua eficiência e aplicabilidade demonstrando que as novas abordagens são válidas e alcançam resultados relevantes.

Na primeira aplicação, o objetivo específico é possibilitar a predição de custos em novos ambientes ou mapas, tornando possível a obtenção de planos ou políticas para novos caminhos em novos ambientes a partir da percepção das características dos mapas do conjunto de treinamento. Já na segunda aplicação, possibilitar a predição de parâmetros probabilísticos em problemas de redes complexas utilizando o modelo de grafos de *Markov*.

Essas abordagens, foram as formas escolhidas para serem estudadas e exploradas neste trabalho, porém as formulações apresentadas bem como suas técnicas de solução podem ser enquadradas, com algumas adaptações, nos mais variados problemas. Tais como o experimento apresentado em (TASKAR et al., 2005) sobre a predição da conectividade do dissulfeto nas proteínas contendo resíduos de cisteína. Ou então o experimento em (TASKAR; GUESTRIN; KOLLER, 2003) sobre a identificação de letras em documentos manuscritos. Ou o processamento da linguagem natural abordado em (TSOCHANTARIDIS et al., 2005). E para finalizar, a extração de certas imagens e sua análise, tal como o reconhecimento de face, visto em (YANN et al., 2006).

0.3 Organização do Trabalho

O Capítulo 1 apresenta alguns conceitos necessários para o entendimento do trabalho. Tais como grafos, *Perceptron* e *Perceptron com Margem*.

No Capítulo 2 tem-se o modelo de predição de dados estruturados. A base para o entendimento das relações entre dados estruturados encontra-se neste capítulo.

Segue-se o aprendizado estruturado no Capítulo 3, abordando agora o ferramental teórico necessário para fazer sua predição. Os modelos estruturados podem ser observados e entendidos através de exemplos. São vistos também a modelagem do problema de maximização sob o ponto de vista estruturado.

A seguir, no Capítulo 4 são apresentadas técnicas de solução para o problema visto no Capítulo 3.

O Capítulo 5 trata especificamente da modelagem de um problema da predição de custos visando embasar o conhecimento aplicado. O exemplo em questão aborda o apren-

dizado dos custos necessários para problemas de planejamento de caminhos. Os resultados experimentais, derivados tanto de mapas artificiais quanto reais, são apresentados. Tabelas e Imagens são usadas para explicar e ilustrar o experimento.

O capítulo 6 aborda a predição de dados estruturados em grafos de *Markov*. Uma breve teoria sobre o processo de geração desses grafos é apresentada e o processo do aprendizado estruturado é feito baseado nesta teoria. O objetivo é predizer os possíveis parâmetros que levam ao surgimento de um grafo com determinada característica.

Finalmente, no Capítulo 7, apresentam-se algumas conclusões e possibilidades de trabalhos futuros.

1 Noções Introdutórias

O objetivo desta seção é contextualizar esse trabalho dentro da grande área da Inteligencia Artificial por meio da apresentação e enquadramento se suas características. Para isso são vistos importantes conceitos como: *Perceptron*, *Perceptron com Margem*, *Perceptron Dual* e método *Kernel*.

1.1 Perceptron

O algoritmo *Perceptron* está dentro do escopo de aprendizado supervisionado e foi desenvolvido por (ROSENBLATT, 1958). É composto pelo neurônio do modelo de McCulloch-Pitts (MCCULLOCH; PITTS, 1943) (Figura 1), com função de limiar. O *Perceptron* é a forma mais simples de uma RNA (Rede Neural Artificial), usada para classificação cujos padrões estão em lados opostos de um hiperplano. Consiste de um único neurônio com pesos sinápticos ajustáveis e um possível bias b cujo efeito é deslocar a fronteira de decisão em relação a origem.

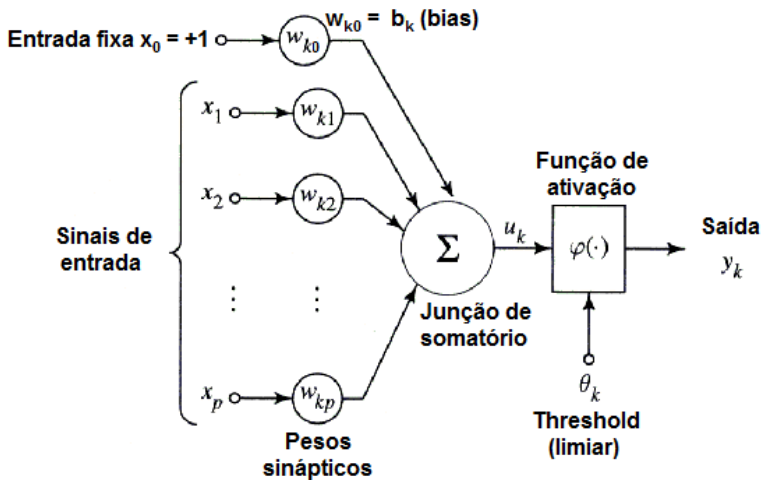

Figura 1 – Modelo de McCulloch-Pitts de um neurônio artificial

Para problemas linearmente separáveis, um classificador linear será representado no espaço de entrada por um hiperplano, chamado de função discriminante, dado pela seguinte equação ($\langle\ ,\ \rangle$ representa o produto interno entre dois vetores em $\mathbb{R}^d$):

$$g(x) = \langle w, x \rangle + b, \tag{1.1}$$

onde $x_{\Xi} \in \mathbb{R}^d$ é o vetor de entrada, $w_{\Xi} \in \mathbb{R}^d$ representa o vetor normal ao hiperplano e $b \in \mathbb{R}$ o valor do bias.

Pode-se considerar a integração do bias da equação em componente adicional do vetor w_Ξ, adicionando também uma componente $+1$ no vetor representativo de cada ponto x_Ξ, conforme Figura 1. Aumenta-se, na verdade, o espaço em uma dimensão. Observe que neste novo espaço só se consideram hiperplanos passando pela origem. Assim a equação geral, toma a forma:

$$f(x) = \langle w, x \rangle, \tag{1.2}$$

onde $x \in \mathbb{R}^{d+1}$ e $w \in \mathbb{R}^{d+1}$.

Geometricamente, considerando o espaço original da equação 1.1, o hiperplano é deslocado em relação a origem, de acordo com um fator w_0, conforme o neurônio da (Figura 1), que assume o papel de bias variável, durante o aprendizado.

Dado o conjunto de treinamento $Z = \{(x_i, y_i)\}_{i=1}^m$, $(x_i, y_i) \in X \times Y$, o qual $X \subseteq \mathbb{R}^n$ é o espaço de entrada e $Y = \{-1, +1\}$ é o espaço de saída; considere, por simplicidade, que x_i já incorpora o componente adicional $+1$, ou seja, $\mathbb{R}^n = \mathbb{R}^{d+1}$. Tem-se que o espaço de hipóteses é restrito a $H = \{h \in Y^X | h = sign(\langle x_i, w \rangle)\}$. A soma do produto entre pesos w, normal ao hiperplano, e entradas x_i, também chamado de **vetor característica**, alimenta o neurônio de saída e seu resultado é comparado com um valor limiar, geralmente 0. A função de perda, neste caso, é uma função limiar $J(u)$ que modela a característica binária deste neurônio. Matematicamente tem-se a função de perda 0 ou 1:

$$J(u) = \sum_{i=1}^{m} Max\{0, \phi(-y_i \langle w, x_i \rangle)\}. \tag{1.3}$$

A função $\phi(-y_i \langle w, x_i \rangle)$ é uma função constante por partes e não diferenciável. Sendo $z = -y_i \langle w, x_i \rangle$ tem-se $\phi(z) = 1$ se $z \geq 0$, caso contrário $\phi(z) = -1$. Onde y_i representa a saída desejada, ou seja, a classe a qual o estímulo pertence. Se y_i e $\langle w, x_i \rangle$ pertencerem a mesma classe, ou seja, se $\langle w, x_i \rangle$ tiver o mesmo sinal de y_i, o resultado de ϕ em 1.3 será negativo, o que resultará num $J(u) = 0$ acarretando a não correção, se for verdade para todo $1, \ldots, m$. Torna-se mais apropriado à utilização de uma nova função de perda $J(w)$, linear por partes, dada pela soma negativa de todos valores funcionais, também chamados de valores de margens, das amostras classificadas incorretamente. Ou seja:

$$J(w) = \sum_{i=1}^{m} Max\{0, -y_i \langle w, x_i \rangle\}. \tag{1.4}$$

No método da descida mais íngreme, os ajustes sucessivos ao vetor de peso w são no sentido oposto ao vetor gradiente. Com o gradiente local $\nabla J(w) = -y_i x_i$ tem-se que a atualização do vetor w se dá na forma: $w(t+1) = w(t) + \eta x_i y_i$. Onde η é a taxa de aprendizado $(0 < \eta \leq 1)$. Tem-se então que a correção só acontece se $J(w) = 1$, ou seja,

se o elemento x_i está sendo classificado na classe errada. Se x_i já estiver sendo classificado na classe certa não há necessidade de correção.

Visto que o y_i representa a classe correta ao qual o elemento x_i deve pertencer e, desta forma, definindo se ocorrerá a correção e seu sinal. O enunciado acima poderia ser reescrito da forma descrita em (HAYKIN, 2001):

Sejam C_1 e C_2 classes hipotéticas, no primeiro caso, se a amostra x_i é corretamente classificada pelo vetor de pesos w, então não há correção:

$w_{(t+1)} = w_{(t)}$ se $\langle w, x_i \rangle > 0$ e x_i pertence a classe C_1.

$w_{(t+1)} = w_{(t)}$ se $\langle w, x_i \rangle \leq 0$ e x_i pertence a classe C_2.

Caso contrário o vetor de pesos w é atualizado de acordo com a regra:

$w_{(t+1)} = w_{(t)} - \eta x_i$ se $\langle w, x_i \rangle > 0$ e x_i pertence a classe C_2.

$w_{(t+1)} = w_{(t)} + \eta x_i$ se $\langle w, x_i \rangle \leq 0$ e x_i pertence a classe C_1.

Durante o processo de treinamento do *Perceptron*, busca-se encontrar um conjunto de pesos que defina um hiperplano ortogonal a w que separe as diferentes classes, de forma que a rede classifique corretamente cada entrada x_i que está sendo incluída no processo de aprendizado (Figura 2).

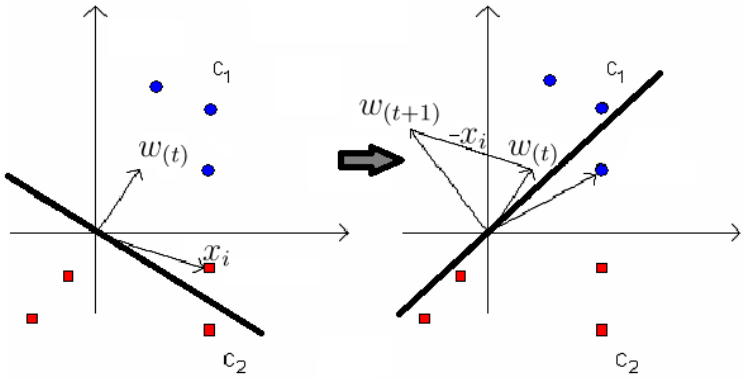

Figura 2 – Uma correção durante a execução do algoritmo Perceptron para $\eta = 1$

Assim, facilitando a implementação do algoritmo, a adaptação do vetor do peso w pode ser resumida adequadamente na regra de aprendizagem por correção de erro. Tem-se que se y_i for a classe resposta desejada (1 ou -1) e $d_{(t)} = sinal\langle w, x_i \rangle$ for a classe calculada atual:

$$w_{(t+1)} = w_{(t)} + \eta \; sinal[y_{(t)} - d_{(t)}]x_i. \tag{1.5}$$

O algoritmo de treinamento do *Perceptron* sempre chega a uma solução para o problema de separação de duas classes linearmente separáveis em um tempo finito. Em (NOVIKOFF, 1962) este comportamento foi provado através do Teorema da Convergência do *Perceptron*.

1.2 Perceptron de Margem Fixa - PMF

Em (DUDA; HART; STORK, 2001) é proposta uma versão alternativa para o algoritmo *Perceptron* onde um valor de margem γ é aplicado. Um erro ocorre quando $y_i(\langle w, x_i\rangle) \leq \gamma$. No entanto, desde que o problema seja linearmente separável, pode-se encontrar uma solução viável para qualquer γ, bastando para isso aumentar o valor das componentes do vetor w, ou seja, o valor de sua norma, mesmo que sua direção não seja alterada. Portanto, esta margem não cria qualquer tipo de restrição adicional para o problema original e sua formulação para a obtenção de uma solução viável para um sistema de inequações lineares é a seguinte:

$$y_i(\langle w, x_i\rangle) \geq \gamma. \tag{1.6}$$

Para resolver este problema é necessário estabelecer alguma forma de regularização no sentido de controlar ou de limitar o valor do vetor w e sua norma (LEITE; NETO, 2007). Caso não haja uma limitação, por exemplo, a adição de uma restrição adicional de normalização tal como: $||w|| = 1$, nem algum tipo de controle, como o que será visto adiante na equação 1.7; o sistema de inequações, se linearmente separável, apresentará sempre uma solução viável considerando o crescimento da norma e, consequentemente, do valor do produto interno na equação 1.6, para qualquer valor de margem γ.

Assim, a formulação de margem fixa busca uma nova formulação para o modelo *Perceptron* no sentido de garantir que o conjunto de exemplos guarde uma distância geométrica mínima em relação ao hiperplano separador, sem limitar diretamente o valor da norma do vetor w. Para tanto, é considerada a restrição de que cada amostra deva possuir um valor de margem geométrica euclidiana: $\langle w, x_i\rangle/||w||_2$, superior ou igual ao valor estabelecido como distância fixa mínima ao hiperplano separador. Observe que esta distância geométrica pode ser interpretada como a realização do produto interno do vetor x_i pelo vetor unitário de direção w, representado por $w/||w||_2$. Neste sentido, a solução só será viável se satisfizer o seguinte sistema de inequações não-lineares para determinado valor de margem fixa representado pelo parâmetro γ_f (LEITE; NETO, 2007):

$$\frac{y_i(\langle w, x_i\rangle)}{||w||_2} \geq \gamma_f \;\; ou \;\; y_i(\langle w, x_i\rangle) \geq \gamma_f.||w||_2. \tag{1.7}$$

Em função desta modificação, torna-se necessário reescrever a função de perda do modelo de forma a possibilitar a obtenção de uma nova regra de correção. A nova função será equivalente à soma dos valores das respectivas margens geométricas dos exemplos, subtraídos do valor da margem fixa. Ou seja:

$$J(w) = \sum_{i=1}^{m} Max\left\{0, \gamma_f - \frac{y_i(\langle w, x_i\rangle)}{||w||_2}\right\}, \;\; (x_i, y_i) \in Z, \tag{1.8}$$

ou, de outra forma:

$$J(w) = \sum_{i=1}^{m} Max(0, \gamma_f.||w||_2 - y_i(\langle w, x_i \rangle)), \ (x_i, y_i) \in Z. \tag{1.9}$$

Note que se x_i estiver classificado na classe errada, a interpretação da equação é óbvia, tem-se que o resultado de $-y_i(\langle w, x_i \rangle)$ será positivo e participará do somatório da função de perda $J(w)$, aditado pela margem γ_f. Contudo, mesmo que o hiperplano $\langle w, x_i \rangle$ divida as classes corretamente, ou seja, mesmo que $-y_i(\langle w, x_i \rangle)$ seja negativo, pode ocorrer da distância do ponto x_i ao hiperplano de separação, representado por $\langle w, x_i \rangle / ||w||_2$, ser menor que a margem γ_f, então a saída será positiva e contará para o somatório da função de erro $J(w)$. Portanto, ao contrário do algoritmo básico do perceptron, considera-se também como erro, aqueles exemplos que, embora classificados corretamente, não estejam a uma distância mínima, no sentido geométrico, do hiperplano separador (Figura 3). Em (KIVINEN; SMOLA; WILLIAMSON, 2002) foi definido este tipo de correção como a ocorrência de erros de margem.

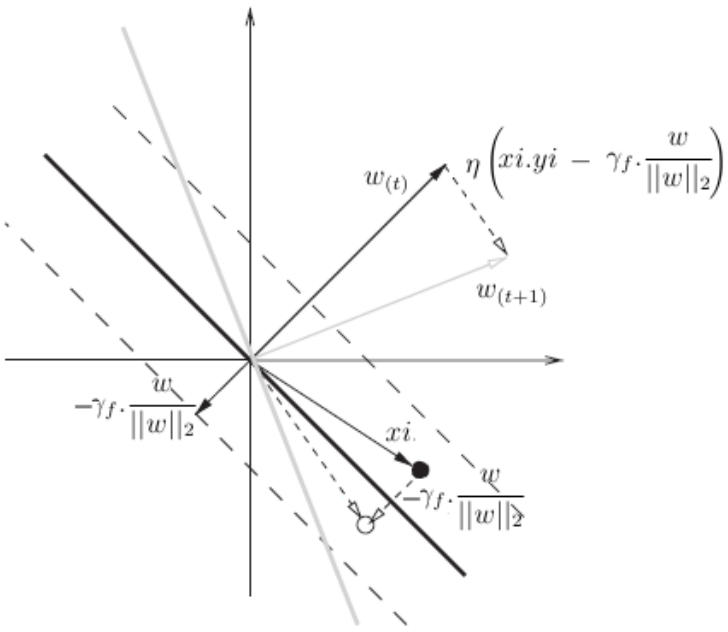

Figura 3 – Interpretação geométrica da margem γ_f (LEITE; NETO, 2007)

A solução do sistema de inequações é determinada pela minimização da função de erro $J(w)$. Neste sentido, tomando-se o oposto do gradiente da função em relação ao vetor w, tem-se a seguinte regra de correção caso ocorra um erro, ou seja, caso $y_i(\langle w, x_i \rangle) < \gamma_f.||w||_2$.

$$w_{(t+1)} = w_{(t)} + \eta \left(x_i.y_i - \gamma_f.\frac{w}{||w||_2} \right), \ (x_i, y_i) \in Z, \tag{1.10}$$

Essa equação de correção possui duas interpretações diretas. A primeira, baseada na observação de que o termo $-w/||w||_2$ representa o vetor unitário de sentido oposto

ao w, sugere que o erro gerado pelo exemplo x_i seja somado na direção oposta do vetor normal (Figura 3). A segunda, relacionada à forma alternativa da equação:

$$w_{(t+1)} = w_{(t)} \left(1 - \frac{\eta.\gamma_f}{||w||_2}\right) + \eta.x_i.y_i. \tag{1.11}$$

Esta sugere que antes de cada correção do vetor w seja feito um escalonamento no valor do mesmo, proporcional ao valor da margem fixa dividido pela norma do vetor w. Neste sentido, pode-se afirmar que a forma de regularização empregada para o controle da norma consiste em uma espécie de decaimento no valor dos componentes de w. Também, devido ao emprego do conceito de margem geométrica percebe-se que esta regularização já esta implícita na própria função de perda a ser minimizada. A prova de convergência pode ser encontrada em (LEITE; NETO, 2007).

1.3 Perceptron com Margem Incremental - *IMA*

O problema de maximizar a margem pode ser interpretado como o problema de achar o w^* tal que;

$$w^* = argMax\{\gamma(w)\}, \tag{1.12}$$

onde $\gamma(w) = Min\left\{\gamma_i | \gamma_i = \frac{y_i.\langle w, x_i\rangle}{||w||},\ i = 1, \ldots, m\right\}$ corresponde a uma margem geométrica.

Tem-se então o conhecido resultado:

$$\begin{aligned} &\gamma^+(w) = \gamma^-(w) = \gamma(w^*) \\ &\gamma^+(w) = Min\left\{\gamma_i | \gamma_i = \frac{y_i.\langle w, x_i\rangle}{||w||},\ y_i = +1,\ i = 1, \ldots, m\right\} \\ &\gamma^-(w) = Min\left\{\gamma_i | \gamma_i = \frac{y_i.\langle w, x_i\rangle}{||w||},\ y_i = -1,\ i = 1, \ldots, m\right\}. \end{aligned} \tag{1.13}$$

Este resultado pode ser verificado pelas condições de *Karush-Kuhn-Tucker* obtidas da formulação apresentada em (VAPNIK, 1998) para o *SVM* (*Support Vector Machine*). Entretanto, sabe-se que se $\gamma^+(w) \neq \gamma^-(w)$, então a margem γ atual não é ótima. Adicionalmente, w^* é também uma solução para o problema de maximizar as distâncias entre as classes, então:

$$w^* = argMax\{(\gamma^+(w) + \gamma^-(w))\}. \tag{1.14}$$

Como consequência:

$$2\gamma(w^*) \geq \gamma^+(w) + \gamma^-(w). \tag{1.15}$$

Baseado nestes resultados que o IMA (*Incremental Margin Algorithm)* foi desenvolvido (LEITE; NETO, 2007). Tem-se que a atualização ocorre iterativamente na forma:

$$w \leftarrow PMF(S, w, \gamma_f, \eta, TMAX) \tag{1.16}$$
$$\gamma_f \leftarrow Max((\gamma^+(w) + \gamma^-(w)/2, (1+\delta)\gamma_f),$$

onde $\delta \in (0, 1)$ é algum incremento mínimo de margem e PMF é uma execução do *Perceptron de Margem Fixa*. O algoritmo prossegue com a execução até convergir para algum valor ou até um número máximo $TMAX$. Repare na semelhança com uma busca binária. Os critérios de convergência, suas provas e corolários podem ser vistos em (LEITE; NETO, 2007).

1.4 Perceptron Dual

Uma interessante característica do *Perceptron* é que a solução final do vetor w é sempre uma combinação linear dos pontos de entrada.

$$w = \sum_{j=1}^{m} \alpha_j . x_j, \tag{1.17}$$

onde $\alpha_j \in \mathbb{R}$ é um coeficiente associado ao exemplo x_j. Pode-se considerar $\alpha_j \geq 0$ se for acrescentado a seguinte modificação:

$$w = \sum_{j=1}^{m} \alpha_j . y_j . x_j. \tag{1.18}$$

Estimar os coeficientes $\alpha_1, \ldots, \alpha_m$, é equivalente a estimar w. Adicionalmente, a projeção dos exemplos x_i sobre w pode ser computada usando os coeficientes α_j e os exemplos de entrada x_j, onde $j \in \{1, 2, \ldots, m\}$:

$$\langle w, x_i \rangle = \langle \sum_{j=1}^{m} \alpha_j . x_j \rangle = \sum_{j=1}^{m} \alpha_j . \langle x_j, x_i \rangle \tag{1.19}$$

Então a função discriminante (hiperplano), ao substituir a equação 1.17 em 1.2, toma a seguinte forma:

$$f(x_i) = \sum_{j=1}^{m} \alpha_j . \langle x_j, x_i \rangle, \tag{1.20}$$

ou, alternativamente, considerando o facilitador visto na equação 1.18 para $\alpha_j \geq 0$:

$$h(x_i) = \sum_{j=1}^{m} \alpha_j . y_j \langle x_j, x_i \rangle, \tag{1.21}$$

Esta computação da forma dual, a primeira vista, é mais custosa que a forma primal. Entretanto, pode-se agilizar a estimação do coeficiente α pré-computando o produto interno entre os exemplos de entrada e armazenando-os antecipadamente em uma matriz $G_{ij} = \langle x_i, x_j \rangle$, conhecida como *Matriz de Gram*. Além disto, esta representação traz vantagens em termos de poder de expressividade, vistas a seguir na seção 1.5.

1.5 Método *Kernel*

Enquanto a simplicidade do *Perceptron* torna-o muito atrativo, seu poder de expressividade, só resolvendo problemas linearmente separáveis, é uma desvantagem. Muitas aplicações envolvem relacionamentos não-lineares. Uma técnica para tratar relacionamentos não-lineares usando algoritmos lineares é transformar os relacionamentos dos dados apropriadamente, tornando-os separáveis linearmente em espaços diferentes do original.

É possível fazer um mapeamento explícito aplicando-se uma transformação no espaço de entrada original, tornando-o um espaço de mais alta dimensão, denominado espaço de características ($\Phi - space$) e representado por P. Com a realização deste mapeamento, $\Phi : \mathbb{R}^n \to P$, é possível a representação do conjunto de amostras neste novo espaço, $x \mapsto \Phi(x)$, no qual o problema se torna linearmente separável. Na Figura 4, tem-se explicitamente a função de transformação: $\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$.

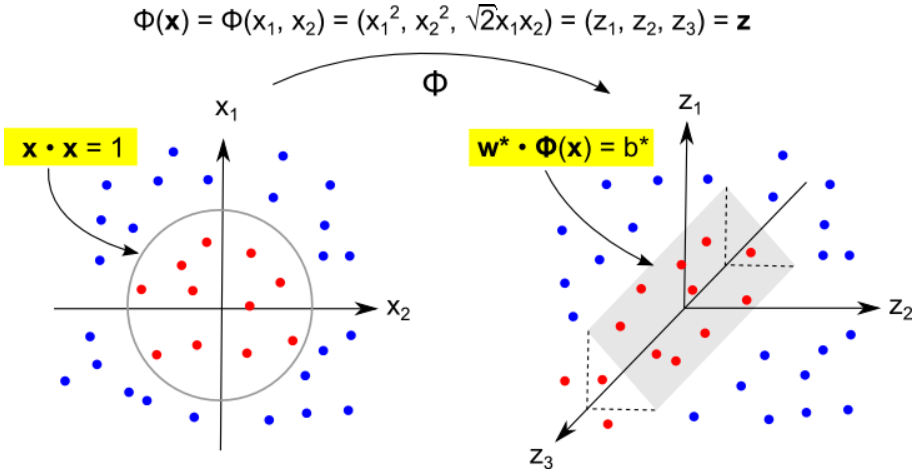

Figura 4 – A função Φ explicitamente definida num mapeamento $\mathbb{R}^2 \to \mathbb{R}^3$

Similarmente, relacionamentos mais complexos podem ser linearizados usando um mapeamento apropriado. Entretanto, se for alta a dimensionalidade dos dados de entrada, este mapeamento tende a ficar computacionalmente intratável. Por exemplo, é comum, em problemas envolvendo visão computacional, ter-se imagens de 256×256 *pixels*, resultando num vetor de dimensionalidade $m = 65536$. O número D de todos os monômios de segundo grau neste caso é $(65537 \times 65536)/2 = 214756416$. Aumentando-se exponencialmente de acordo com o tamanho dos dados de entrada e o grau dos monômios. Generalizando, tem-se:

$$D = \frac{(\varpi + d - 1)!}{(\varpi - 1)!d!}, \tag{1.22}$$

onde ϖ é a dimensionalidade dos exemplos de entrada e d é o grau.

Entretanto, o algoritmo dual do *Perceptron* não requer que os dados sejam explicitamente mapeados. Em vez disso, basta mapear a *Matriz de Gram* no espaço transformado

(espaço de características), $G_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$, sendo comumente renomeada para *Matriz Kernel*. Similarmente, a classificação de um exemplo de teste x_i requer o produto interno $\langle \phi(x_i), \phi(x_j) \rangle$ e não o cálculo de algum $\phi(x_i)$ isolado. Ou seja, caso seja possível computar o produto interno eficientemente, a forma dual pode ser usada sem problemas em relacionamentos complexos.

Portanto, ao projetar-se os pontos no espaço de características, através do mapeamento obtido pela função Φ, necessita-se definir somente a função *Kernel*, não necessitando avaliar a função Φ explicitamente e nem mesmo conhecê-la. Faz-se isso utilizando uma função $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ no algoritmo de treinamento, obtendo, da mesma forma no processo resolutivo, uma superfície de decisão linear no espaço de características P, a qual corresponde a uma superfície de decisão não-linear no espaço de entrada.

De forma simplificada, pode-se resumir a utilização de funções *Kernel* apresentando o seguinte problema: Seja P um espaço conhecido como espaço de *Hilbert*, um espaço vetorial que possui produto interno e cuja métrica gerada por esse produto interno o torne um espaço completo, em outras palavras, é uma generalização do espaço euclidiano que não precisa estar restrita a um número finito de dimensões. Definindo uma função de mapeamento $\Phi : \mathbb{R}^n \rightarrow P$, pode-se estabelecer uma função *Kernel* tal que $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$, na forma de um produto interno do mapeamento de dois vetores associados a função característica Φ, sendo x_i e $x_j \in \mathbb{R}^n d$.

Basicamente, aplica-se o mesmo algoritmo de treinamento no espaço P, para um conjunto de treinamento formado por: $\{(\Phi(x_1), y_1), (\Phi(x_2), y_2), \ldots, (\Phi(x_m), y_m)\} \in P \times Y$. Tal artifício é conhecido como *Kernel Trick*. Para melhor entendimento deste processo é descrito o seguinte exemplo visto em (MüLLER et al., 2001).

Sejam $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^2$. Define-se uma função *Kernel* K como polinomial quadrática: $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^T.\boldsymbol{y})^2 = \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2$. Portanto, deve haver um espaço P de mais alta dimensão e uma função de mapeamento Φ, que viabilizem a definição deste *Kernel*. De fato, para um espaço tridimensional ($\mathbb{R}^3$), pode-se determinar uma função de mapeamento $\Phi(\boldsymbol{x}) = \Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ que garante a existência da função *Kernel*.

Observando a Figura 4, nota-se que os pontos projetados no espaço de características permitem uma separabilidade linear, ao contrário do espaço de entrada, que exige um elipsoide como superfície separadora. Ora, estes valores podem ser computados diretamente com o uso da função *Kernel* K, não necessitando avaliar, explicitamente, a função Φ. Para tanto, é necessária a avaliação de produtos internos relacionados a função

de mapeamento no espaço de características, ou seja:

$$\begin{aligned}\left(\Phi(\boldsymbol{x})^T.\Phi(\boldsymbol{y})\right) &= \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right).\left(y_1^2, \sqrt{2}y_1y_2, y_2^2\right)^\top \\ &= \left((x_1, x_2).(y_1, y_2)^T\right)^2 \\ &= \left(\boldsymbol{x}^T.\boldsymbol{y}\right)^2 \\ &= K(\boldsymbol{x}, \boldsymbol{y}).\end{aligned} \tag{1.23}$$

Portanto, o produto interno dos vetores no espaço de características pode ser substituído pela avaliação da função *Kernel* tendo como argumento o produto interno dos vetores no espaço de entrada. Entretanto, nem todas as funções podem ser utilizadas como *Kernel*, ou seja, nem todas as funções garantem uma transformação na forma $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, tal que $\Phi : \mathbb{R}^n \to P$ e $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \rangle,\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

Também, a unicidade da função Φ e do espaço P, não são garantidos para um determinado *Kernel*. Neste exemplo, para $P = \mathbb{R}^n 4$, pode-se definir, também, um mapeamento alternativo na forma $\Phi(\boldsymbol{x}) = (x_1^2, x_1x_2, x_2x_1, x_2^2)$.

Matematicamente, se $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ é uma função *Kernel* contínua de uma integral positiva num espaço de *Hilbert* com norma L_2 em $\mathbb{R}^n$, ou seja, (MüLLER et al., 2001)

$$\forall f \in L_2(\mathbb{R}^n) : \int K(\boldsymbol{x}, \boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y}) dxdy \geq 0, \tag{1.24}$$

então existe um espaço P e um mapeamento $\Phi : \mathbb{R}^n \to P$ tal que $K(\boldsymbol{x}^T, \boldsymbol{y}) = (\Phi(\boldsymbol{x}).\Phi(\boldsymbol{y}))$. Estas considerações podem ser derivadas através do Teorema de *Mercer* (MERCER, 1909).

Usando a função *Kernel* no aprendizado do *Perceptron* dual, substituindo-a na equação 1.21, tem-se a seguinte função discriminante:

$$h(x) = \sum_{j=1}^{m} \alpha_j.y_j \langle \Phi(x_j), \Phi(x_i) \rangle. \tag{1.25}$$

Uma dificuldade comum é escolher qual função *Kernel* utilizar. Se for possível obter certas características do espaço de entrada então é possível analisar melhores funções *Kernel* para o problema em questão, ou seja, a melhor escolha da função *Kernel* está diretamente relacionada ao problema que se quer abordar. Sendo assim, é de fundamental importância ter-se em mente que a função a ser escolhida será aquela que dará o formato do discriminante no espaço de entrada do problema. Assim, se o problema requer uma função discriminante quadrática, seja uma hipérbole ou elipse, para a sua solução, deve-se utilizar uma função *Kernel* polinomial, que possibilitará a utilização de uma representação linear, deste discriminante, no espaço de características.

Esta transformação pode ser obtida através do uso de várias funções de mapeamento como exemplo: polinomiais, logísticas, gaussianas, etc. Após esta transformação,

é viável a separação dos dados de forma linear no espaço de características através da construção de um hiperplano separador.

Existe um grande ferramental presente na literatura sobre funções *Kernel*, sobre sua teoria e aplicabilidade; por exemplo, os trabalhos de (AIZERMAN; BRAVERMAN; ROZONOER, 1964), (BOSER; GUYON; VAPNIK, 1992) e (SCHoeLKOPF; SMOLA, 2002). Na tese aqui apresentada é visto somente as definições e propriedades básicas necessárias para transformar um aprendizado baseado em um hiperplano linear em um algoritmo com maior poder de expressividade, capaz de achar decisões não-lineares, controlando a complexidade em altas dimensões. Para um estudo aprofundado sobre a formulação e desenvolvimento da função *Kernel*, englobando espaço de *Hilbert* e as condições de *Mercer* para que uma função seja *Kernel*, pode-se utilizar as referências (HALMOS, 1957), (ARONSZAJN, 1950), (MERCER, 1909) e (MüLLER et al., 2001).

2 Modelos de Predição de Dados Estruturados

Nesse capítulo são apresentadas as formulações matemáticas envolvidas na predição dos modelos estruturados.

2.1 Modelos de Predição Estruturados

Os chamados dados estruturados possuem um conjunto de tipos de dados definidos como elementares, de tal modo que exista uma relação estrutural entre seus valores.

Um **modelo de predição estruturado** é aquele cuja saída não é um simples escalar $z_j,\ j = 1, \ldots, num$, mas um conjunto estruturado desses dados ou valores $y = (z_1,\ \ldots,\ z_{num})$, tendo um número de elementos máximo num (TASKAR, 2004). O termo estruturado se refere a existência de um conjunto de restrições e correlações que moldam um espaço de saída $Y \subseteq Z_1 \times \ldots \times Z_{num}$ definido como um subconjunto do produto de espaços de saída das variáveis elementares. Num exemplo simples, com um espaço discreto, tem-se que um escalar z_j é um elemento que corresponde a alguma letra do alfabeto $Z_j,\ j = 1, \ldots, num$ (espaço de saída de z_j), y é uma palavra formada pela junção dos elementos discretos e Y é um dicionário, para um num grande o suficiente e já com as restrições do idioma inseridas. Algumas dessas restrições podem ser facilmente visualizadas através das correlações entre os elementos, tais como, antes de p ou b nunca deve vir a letra n. Se nenhum tipo especial de restrição for estabelecida e considerando que a cardinalidade de todos os alfabetos Z_j é a mesma, ou seja, $|Z_j| = 26, \forall j$, então Y seria todas as combinações de letras que formam palavras de tamanho máximo num ($26^{num} + 26^{num-1} + 26^{num-2} + \cdots + 26^{num-m},\ m \geq 1$). Um espaço como o citado, por exemplo, com suas restrições e correlações recebe o nome de **espaço estruturado**.

Note que, conforme definição, a cardinalidade de Y é diferente da cardinalidade de Z_j. Se np é a quantidade de palavras no dicionário, então $|Y| = np$, diferentemente de $|Z_j| = 26$, quantidade de letras. De forma contrária, a maior parte da teoria de aprendizado supervisionado focou-se na análise de algoritmos de classificação para o caso em que se tem $num = 1$, tendo um número k multi-classe de saídas elementares pré-determinadas, sem correlações e restrições entre as mesmas, $|Y| = |Z_j| = k$; ademais muitos casos se limitam a resultados binários, $k = 2$. Note que nos casos tradicionais a quantidade de saídas elementares possíveis é que determina a cardinalidade de Y.

Neste trabalho, o foco está em tarefas de previsão que não envolvem uma única decisão com um pequeno conjunto de resultados elementares, mas um conjunto estrutu-

rado e inter-relacionado de decisões. A principal questão, segundo (BAKIR et al., 2007), quando se trata de predizer dados estruturados está no porquê de se fazer predições de um conjunto estruturado de valores ao invés de simplesmente predizer cada saída individualmente. A resposta está justamente em que ao se adotar uma saída estruturada, leva-se em conta as interdependências observadas na forma de restrições e correlações no conjunto estruturado de saída Y, aprimorando a qualidade de predição (TASKAR, 2004) e abordando problemas até então difíceis de serem visualizados e tratados nos modelos tradicionais de predição, como os que serão vistos nas seções 5.5, 5.6 e 6.4.

2.1.1 Modelos Lineares para Predição Estruturada

Segundo (TASKAR, 2004), de forma geral, nos problemas de predição estruturada, a entrada $x \in X$ é um objeto estruturado arbitrário e a saída é um objeto, também estruturado, de valores inter-relacionados $y = (z_1, \ldots, z_{num})$, tais como árvores, grafos ou cadeias de caracteres. O tamanho de num e a estrutura de y dependem deterministicamente da entrada x. Ao se denotar o espaço de saída para um determinado x como Y_x, tem-se o espaço completo de saída como $Y = \bigcup_{x \in X} Y_x$.

Na classe de modelo estruturado H, se assume que uma apropriada associação $f : X \times Y \rightarrow \mathbb{R}^n$ está disponível (BAKIR et al., 2007). Dada uma função f, pode-se então definir H como o conjunto de funções lineares g, parametrizadas por um vetor de pesos $w \in \mathbb{R}^n$ e $w \neq 0$, como segue:

$$g(x, y) = \langle w, f(x, y) \rangle = w^T . f(x, y). \tag{2.1}$$

A função f também pode ser indicada como a função de compatibilidade entre entradas x e saídas y. Esta função implicitamente define um mapeamento h de entradas para saídas, parametrizada pelo vetor w. Então, seu modelo linear de predição estruturado H corresponde seguinte família linear $h_w(x)$ de hipóteses:

$$h_w(x) = argMax_{y \in Y_x} \{g(x, y)\}. \tag{2.2}$$

É importante notar que se o vetor w e a entrada estruturada x forem conhecidas, tem-se um problema de maximização linear direto, sujeito a alguma restrição, o qual no pior caso seriam testados todos elementos estruturados de Y, a fim de se descobrir o y ótimo. Porém, o objetivo principal não é este. Nos problemas de aprendizado estruturado, o ponto mais importante é descobrir o vetor w, dados um conjunto de entradas x e suas saídas correspondentes y. Só então, adquirindo um w com boa capacidade de generalização, predizer qual seria o y ótimo para uma nova entrada x. Pode-se ver tal definição aplicada no exemplo abaixo (TASKAR et al., 2005):

Considere a modelagem da tarefa de atribuição de revisores de artigos como um problema de correspondência de peso máximo bipartido, onde o peso representa o grau de conhecimento do revisor em relação ao artigo em questão.

Em outras palavras, existem exatamente R revisores por artigo onde cada um assina no máximo um número A de artigos. Para cada artigo e revisor tem-se um valor $s_{j,k}$ indicando o nível de qualificação do revisor j para a avaliação do artigo k.

O objetivo é achar uma atribuição para os revisores de artigos que maximiza o peso total. A correspondência é representada por um conjunto binário de variáveis $y_{j,k}$ representando o valor 1 se o revisor j assinou o artigo k, ou seja, se o artigo k foi atribuído ao revisor j, e 0 caso contrário. O valor resultante advindo das atribuições é o seguinte somatório: $s(y) = \sum_{j,k} s_{j,k} \cdot y_{j,k}$.

O peso máximo pode ser resolvido através do seguinte problema de otimização:

$$\begin{aligned} &Max \sum_{j,k} \{s_{j,k} \cdot y_{j,k}\} \\ &Sujeito\ a: \\ &\sum_{j} y_{j,k} = R,\ \sum_{k} y_{j,k} \leq A,\ 0 \leq y_{j,k} \leq 1. \end{aligned} \tag{2.3}$$

A solução deste problema é garantida por ter soluções de integral para qualquer valor da função $s(y)$, visto que A e R são inteiros (SCHRIJVER, 2003).

Porém, em um problema mais complexo, o qual é levado em conta as palavras que aparecem no *site* do revisor e sua comparação com as palavras do artigo a ser revisado, há também a necessidade de aumentar o peso de certas palavras, dependendo do grau de relevância das mesmas, modificando o valor $s_{j,k}$ do revisor.

Considera-se, assim, $webpage(j)$ o conjunto de palavras que ocorrem na *homepage* do revisor j, e $resumo(k)$ as palavras que aparecem no resumo do artigo k. Então $x_{j,k}$ denota a interseção do conjunto de palavras como $webpage(j) \bigcap resumo(k)$. Tem-se agora que $s_{j,k} = \sum_d w_d \psi(palavra_d \in x_{j,k})$, onde ψ é uma função indicadora ou função característica, ou seja, indica se o elemento de índice d pertence ao subconjunto, e w_d é o o vetor de pesos que se deseja determinar com dimensão igual a quantidade total de termos $x_{j,k}$.

Define-se então $f_d(x, y) = \sum_{j,k} y_{j,k} \psi(palavra_d \in x_{j,k})$ e pode ser interpretada como o número de vezes que um autor j assinou um artigo k de modo que a $palavra_d$ estava tanto na $webpage(j)$ do revisor, quanto no $resumo(k)$. Associando-se o vetor de pesos w_d à função $f_d(x, y)$, tem-se: $w_d^T f_d(x, y)$. Deste modo, pode-se representar o objeto $s(y)$ visto anteriormente como uma combinação ponderada de um conjunto de características tal que $s(y) = \sum_d \{w_d^T f_d(x, y)\}$. Repare que seria mais correto, agora, representá-lo como $s(w, x, y)$.

No capítulo 3, será vista mais a fundo a formulação matemática desse processo de predição estruturada com a definição de suas restrições relacionadas.

3 Predição Estruturada e Funções de Restrição

Pode-se definir o espaço de saída para uma entrada estruturada x usando um conjunto de funções de restrição:

$$\Psi(x, y) : X \times Y \longmapsto \mathbb{R}^n. \tag{3.1}$$

E como já visto em (2.2), a classe dos modelos de predição estruturado H corresponde à seguinte família linear: $h_w(x) = argMax_{y \in Y_x}\{w^T.f(x, y)\}$. A saída y está sujeita a alguma função $\Psi(x, y)$ restritiva e a função $f(x, y)$ é um vetor de funções na forma $f : X \times Y \longmapsto \mathbb{R}^n$. Como exemplo, pode-se ter como regra restritiva $Y_x = \{y : \Psi(x, y) \leq 0\}$. Observe a dependência do espaço de saída em relação a entrada x.

Esta formulação é bem geral, para muitas escolhas f e Ψ achar o y ótimo, dado um x, é computacionalmente intratável. Isto acarreta a busca por modelos onde o problema de otimização pode ser tratado em tempo polinomial. Isto inclui certo tipos de gramáticas livre de contexto, bem como problemas de otimização convexa linear ou quadrática; em relação as duas últimas, uma explicação resumida é apresentada no Apêndice D. Em casos intratáveis, pode-se usar uma aproximação de tempo polinomial que provê um limite superior ou inferior para a solução, por exemplo, nas aproximações de *matching* máximo em grafos de determinadas topologias específicas. Note, contudo, que o objetivo principal neste trabalho é estimar o vetor w de modo que $h_w(x)$ retrate, alguma saída desejada y pré-determinada, como será visto a seguir.

3.1 Predição Estruturada no Aprendizado Supervisionado

Dentro do contexto de aprendizado supervisionado, a predição estruturada também necessita de um **conjunto de treinamento**, um **conjunto de testes** e busca o ajuste de pesos do vetor w.

Dado um conjunto de treinamento $S = \{(x_i, y_i),\ i = 1, \ldots, m\}$ formado por uma coleção de pares, sendo cada par formado por uma amostra representada por um objeto estruturado x_i e uma solução estruturada desejada y_i deseja-se obter um vetor de parâmetros w tal que a hipótese $h_w(x_i)$ da equação 2.2 seja justamente o y_i, par de x_i:

$$argMax_{y \in Y_i}\{w^T.f(x_i, y)\} = y_i,\ i = 1, \ldots, m, \tag{3.2}$$

sendo Y_i o espaço de saída de todas as soluções possíveis dependente do objeto estruturado x_i. Considere que $y_i \in Y_S$, onde Y_S refere-se ao seguinte conjunto: $Y_S = \{y_i \in S,\ i =$

$1, \ldots, m\}$. Note que $Y_i \supseteq Y_S$. Ou seja, o aprendizado do vetor de parâmetros w permite que a melhor solução encontrada para cada par da coleção reflita a solução proposta no conjunto de treinamento.

Embora a cardinalidade de Y_i possa ser muito elevada, é possível, através da utilização do conceito de margem para problemas estruturados, resolver de forma eficiente a determinação do falso-exemplo y. Seja a margem $\gamma_{y_i,y}$ de uma amostra (x_i, y_i) sobre algum $y \in Y_i$ interpretada como:

$$\gamma_{y_i,y} = \frac{w^T.f(x_i, y_i) - w^T.f(x_i, y)}{||w||_2}, \ y \in Y_i, \tag{3.3}$$

onde, se $y = y_i$, tem-se margem 0. Note também que deve-se ter $w \neq 0$, evitando desse modo que $||w||_2$ seja igual a zero. É importante observar que $w/||w||_2$ é um vetor unitário.

A Figura 5 ilustra o caso mais simples para o cálculo do $\gamma_{y_i,y}$, em um espaço euclidiano de duas dimensões.

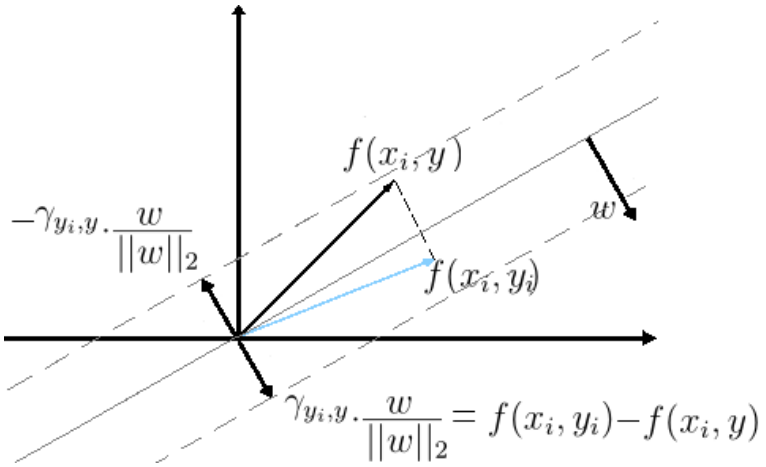

Figura 5 – Interpretação geométrica de $\gamma_{y_i,y}$ para o caso simples, onde $f(x_i, y)$ retorna coordenadas no plano cartesiano.

A dificuldade de se utilizar este conceito de margem em técnicas de predição está no fato de ser necessário calcular cada $f(x_i, y)$ e então compará-lo com todos os $f(x_i, y_i)$ para cada instância i. O número de comparações a serem efetuadas caem no caso exponencial. Contudo, não há necessidade de se calcular a margem de cada elemento y_i do conjunto de treinamento em relação a todos os outros elementos $y \in Y_i$ possíveis. Basta saber qual a menor margem γ_i para (x_i, y_i), dado determinado w^T, se comparada juntamente com todos os outros elementos $y \in Y_i$ e $y \neq y_i$, em outras palavras, o quanto a classe verdadeira *vence* sobre as erradas. Formalmente, segundo (TASKAR, 2004):

$$\gamma_i = \frac{Min_{y \neq y_i}\{w^T.f(x_i, y_i) - w^T.f(x_i, y)\}}{||w||_2}, \ \forall y \in Y_i. \tag{3.4}$$

Esta definição é inspirada no conceito de margem geométrica de (VAPNIK, 1998) e semelhante a definição de (LEITE; NETO, 2007) visto na seção 1.2 e principalmente nas

definições de margem para os problemas multi-classe vistos em (CRAMMER; SINGER, 2001) e (WESTON; WATKINS, 1998). A Figura 6 ilustra um caso simples para o cálculo do γ_i, em um espaço euclidiano de duas dimensões. Note que $\frac{w^T}{||w||_2}$ é um versor de tamanho 1, desse modo, o que determina o tamanho de γ_i. $\frac{w^T}{||w||_2}$ é o próprio γ_i, ou seja, $\frac{w^T}{||w||_2}$ determina somente a direção do vetor.

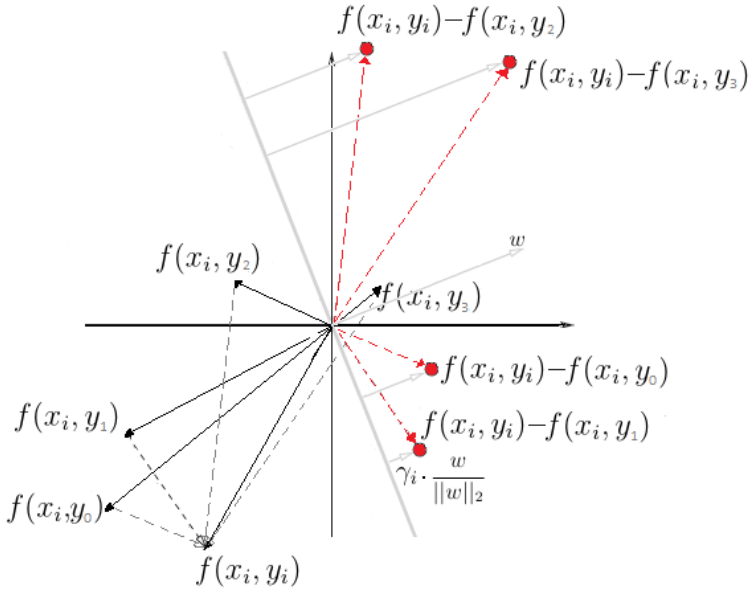

Figura 6 – Interpretação geométrica de γ_i para o caso simples, onde $f(x_i, y)$ retorna coordenadas no plano cartesiano.

Observe novamente a equação 3.2, onde o argumento desejado que maximiza a função é o próprio y_i. Tem-se então que neste aprendizado supervisionado, w^T refletirá a solução y_i, proposta no conjunto de treinamento S, se resultar em uma margem $\gamma_i \geq 0$. Com base nesta afirmação, é possível empregar uma outra definição para margem, o qual pode ser vista em (TSOCHANTARIDIS et al., 2005):

$$\gamma_i = \frac{w^T.f(x_i, y_i) - Max_{y \in Y_i, y \neq y_i}\{w^T.f(x_i, y)\}}{||w||_2}, \ \forall y \in Y_i. \tag{3.5}$$

Note que com a restrição de $\gamma_i \geq 0$, ou seja, $w^T.f(x_i, y_i) \geq w^T.f(x_i, y)$, $\forall y \in Y_i$, a equação 3.5 é equivalente a 3.4. Logo, ao minimizar a diferença em 3.4, se está implicitamente calculando o y que resulta no máximo em 3.5 e o subtraindo de $w^T.f(x_i, y_i)$.

Para simplificar a notação, visto que x_i é o mesmo em ambas as parcelas, tem-se que $f(x_i, y_i) = f(y_i)$ e $f(x_i, y) = f(y)$, de modo que a equação 3.5 toma a forma:

$$\gamma_i = \frac{w^T.f(y_i) - Max_{y \in Y_i, y \neq y_i}\{w^T.f(y)\}}{||w||_2}, \ \forall y \in Y_i. \tag{3.6}$$

Dado esta definição para γ_i e considerando que se deseja obter uma margem de separação $\gamma_i \geq 0$, o objetivo deste aprendizado supervisionado é achar uma função

$g(x_i, y_i) = \langle w, f(x_i, y_i)\rangle$, através do aprendizado do vetor de parâmetros w^T, cujo valor para y_i resulte numa solução em g maior ou igual a todos os outros possíveis $y \in Y_i$ em g.

$$w^T.f(y_i) \geq Max_{y \in Y_i, y \neq y_i}\{w^T.f(y)\}. \tag{3.7}$$

Alternativamente, se não for restringido o valor de $y \neq y_i$, tem-se que o cálculo do máximo deve corresponder exatamente ao valor da outra parcela, pois o máximo em $y \in Y_i$ deve corresponder ao próprio y_i, conforme definido em 3.2:

$$w^T.f(y_i) = Max_{y \in Y_i}\{w^T.f(y)\}, \tag{3.8}$$

onde a margem γ_i em 3.6 valerá sempre 0.

Foi tratado até aqui o caso onde o objetivo, visto em 3.2, é que o argumento máximo reflita o y_i do conjunto S. Porém, caso fosse desejado que y_i retratasse o argumento mínimo, a estrutura seria análoga:

$$argMin_{y \in Y_i}\{w^T.f(y)\} = y_i \tag{3.9}$$

$$\gamma_i = \frac{Min_{y \in Y_i, y \neq y_i}\{w^T.f(y)\} - w^T.f(y_i)}{||w||_2}, \ \forall y \in Y_i, \tag{3.10}$$

$$w^T.f(y_i) \leq Min_{y \in Y_i, y \neq y_i}\{w^T.f(y)\}, \tag{3.11}$$

onde ocorre somente a inversão das parcelas, de modo a permanecer uma margem $\gamma_i \geq 0$. Tem-se agora tanto a opção de se minimizar possíveis custos em um objeto estruturado quanto a de maximizar suas recompensas, dependendo somente do contexto no qual o problema geral estará inserido.

Observe que o problema de $Max_{y \in Y_i}\{w^T.f(y)\}$ ou então de $Min_{y \in Y_i}\{w^T.f(y)\}$ é um problema de otimização convexa em w, se f for uma função convexa (SCHRIJVER, 2003), o que torna sua complexidade polinomial. Como será visto no decorrer deste trabalho, através de exemplos, em vários casos é de fácil solução, tal como um problema de caminho mínimo sobre um grafo de estados.

Ao se considerar a margem de todos os γ_i e supondo que todas elas sejam maiores que zero, a margem de separação final γ_z obtida será dada por:

$$\gamma_z = Min\{\gamma_i\}, \forall i. \tag{3.12}$$

Considere $\forall i = 1, \ldots, m$.

É possível também, como sugere (TASKAR, 2004) ao afirmar que aumenta a eficiência do algoritmo de predição, definir uma função de perda $l_i(y) = l_i(y_i, y)$ que escalona a margem geométrica de γ em função de um falso-exemplo y e de sua relação à amostra y_i:

$$\gamma_i = \frac{w^T.f(y_i) - (Max_{y \in Y_i, y \neq y_i}\{w^T.f(y)\} + l_i(y))}{||w||_2}, \ \forall y \in Y_i, \tag{3.13}$$

Então, para $\gamma_i \geq 0$, tem-se:

$$w^T.f(y_i) \geq Max_{y \in Y_i, y \neq y_i}\{w^T.f(y)\} + l_i(y),\ \forall y \in Y_i. \tag{3.14}$$

Note que a recompensa de $w^T.f(y_i)$ deve ser maior que o valor da alternativa $w^T.f(y)$ escalonado pela função de perda $l_i(y)$, ou seja, se o valor advindo da função de perda é pequeno, então requer-se que a função em y_i tenha um valor levemente maior que os outros $y \in Y_i$. Alternativamente, se o valor retornado por $l_i(y)$ é grande, então é requerido que a recompensa em y_i deveria ser substancialmente mais alta que em y.

Para um problema de minimização de custos:

$$w^T.f(y_i) \leq w^T.f(y) - l_i(y),\ \forall y \in y_i, \tag{3.15}$$

Pode-se também relaxar a margem inserindo variáveis de folga ξ, como em outros problemas de aprendizado, como por exemplo em (CRAMMER; SINGER, 2001) e (WESTON; WATKINS, 1998), a fim de permitir erros no conjunto de treinamento, tanto na formulação com a função de perda 3.13, quanto nas formulações sem a mesma. Embora seja necessário um cuidado maior ao se utilizá-la juntamente com a função de perda, para que uma não acabe tirando o propósito da outra. Neste caso seria interessante estudar o problema em específico e analisar também a possibilidade de um $\xi(y_i, y)$. Deste modo, o relaxamento da margem poderia estar associada a alguma característica importante da estrutura, enquanto $l_i(y)$ atuaria como perda associada a alguma outra caraterística específica.

$$w^T.f(y_i) \geq w^T.f(y) + l_i(y) - \xi,\ \forall y \in y_i. \tag{3.16}$$

Alternativamente para minimização:

$$w^T.f(y_i) \leq w^T.f(y) - l_i(y) + \xi,\ \forall y \in y_i. \tag{3.17}$$

3.2 Formulação de Máxima Margem

Na seção anterior foi descrita a teoria necessária para desenvolver o conceito de margem para problemas estruturados. Observe novamente a equação 3.4 da seção 3.1, tem-se que minimizando $||w||_2$, a margem γ_i será maximizada. A abordagem mais utilizada na solução deste problema, segundo (TASKAR et al., 2005), é a seguinte:

$$\begin{aligned} &Min\frac{1}{2}\|w\|^2 \\ &Sujeito\ a: \\ &w^T.f(y_i) \geq Max_{y \in Y_i}\{w^T.f(y) + l_i(y)\}. \end{aligned} \tag{3.18}$$

Tal caso é análogo aos casos de separabilidade da formulação das máquinas de vetores suportes de (VAPNIK, 1998), considerando seu caso multi-classe visto em (CRAMMER; SINGER, 2001), então pode-se expressar a solução ótima do vetor w para cada instância i como o problema de otimização quadrática, conforme descrito em 3.18. Foi assumido que as características são completas e variadas o suficiente para satisfazer as restrições.

Em problemas estruturados, onde a predição é feita através de um conjunto de dados que obedecem determinada estrutura, a função de perda possui uma função diferente dos problemas de aprendizado tradicionais, tendo a característica de penalizar ainda mais os erros, ou seja, a diferença entre a saída da estrutura atual y e a desejada y_i. Neste caso, segundo (TASKAR et al., 2005), uma função de perda natural seria o cálculo do número de variáveis elementares preditas incorretamente, um tipo de distância de *Hamming* entre y_i e o falso-exemplo y.

Observe também que $Max_{y \in Y_i}\{w^T.f(y) + l_i(y)\}$ tem precisamente a mesma forma do problema de predição para o qual se necessita aprender os parâmetros. Porém, como já mencionado na seção anterior, em vários problemas sua complexidade é polinomial. Um exemplo simples e que será utilizado neste trabalho é a cálculo do caminho mínimo.

Em uma versão modificada pode-se considerar a utilização direta da margem γ, a qual deve ser maximizada com a restrição adicional de controle da norma do vetor de parâmetros w. Assim o problema de otimização convexa, sugerido em (TSOCHANTARIDIS et al., 2005), pode ser reescrito na forma:

$$\begin{aligned} &Max\ \gamma \\ &||w|| = 1 \\ &Sujeito\ a: \\ &w^T.f(y_i) - Max_{y \in Y_i}\{w^T.f(y) + l_i(y)\} \geq \gamma. \end{aligned} \tag{3.19}$$

Desta forma, a minimização da norma do vetor w ou a equivalente maximização da margem γ resulta na obtenção de uma solução de máxima margem que satisfaz o conjunto de restrições. Para os casos de não separabilidade pode-se admitir a introdução de variáveis de folga como utilizado nas máquinas de vetores suporte.

Esta formulação é um problema de programação quadrática convexo em w desde que $Max_{y \in Y_i}\{w^T.f(y) + l_i(y)\}$ seja convexo em w. O próximo capítulo aborda as técnicas de solução necessárias para resolver os problemas apresentados neste capítulo.

Da mesma forma que mencionado na seção 3.1 em especial na equação 3.11, caso fosse necessário maximizar a margem para um problema de minimização, a formulação

seria a seguinte:

$$\begin{aligned} & Min\ \frac{1}{2}\|w\|^2 \\ & Sujeito\ a: \\ & w^T.f(y_i) \leq Min_{y \in Y_i}\{w^T.f(y) + l_i(y)\}. \end{aligned} \tag{3.20}$$

4 Técnicas de Solução

Os métodos de soluções abordados são: o *Perceptron estruturado*, o *Perceptron Estruturado com Margem Zero*, o *Perceptron estruturado com Margem Incremental* e o método baseado no subgradiente, este último, segundo (BERTSEKAS, 2003), pode ser implementado de forma simples. Os três primeiros foram desenvolvidos pelo autor e orientadores em (COELHO; NETO; BORGES, 2009) e (COELHO; NETO; BORGES, 2012). O ferramental necessário para a solução do problema de maximização, sob o ponto de vista da otimização pode ser encontrado no Apêndice D. Uma breve explanação sobre subgradientes pode ser encontrado no Apêndice E.

4.1 Perceptron Estruturado

Utilizando os conceitos apresentados no Capítulo 2 sobre dados estruturados juntamente com os princípios da formulação *Perceptron*, seção (1.1), foi elaborado o *Perceptron Estruturado* (COELHO; NETO; BORGES, 2009). Espaços estruturados de saídas são tipicamente aprendidos usando extensões de algoritmos de classificação para simples estruturas.

Nesta abordagem, dado um conjunto de treinamento $S = \{(x_i, y_i)\};\ i = 1, \ldots, m$, deseja-se obter um vetor de parâmetros w tal que:

$$argMax_{y \in Y_S}\{w^T.f(y)\} = y_i, \tag{4.1}$$

onde $Y_S = \{y_i \in S,\ \forall i\}$. Considere $\forall i \equiv (i = 1, \ldots, m)$.

Resumidamente, no *Perceptron Estruturado*, deseja-se uma margem de separação maior ou igual a zero, ou seja, de acordo com a definição de margem vista em 3.6 e reescrita aqui como:

$$\gamma_i = \frac{w^T.f(y_i) - Max_{y \in Y_S, y \neq y_i}\{w^T.f(y)\}}{||w||_2}, \tag{4.2}$$

é desejada uma margem $\gamma_i \geq 0, \forall i$, isto é, $w^T.f(y_i) - Max_{y \in Y_S, y \neq y_i}\{w^T.f(y)\} \geq 0, \forall i$. Desse modo, o conjunto de treinamento S é linearmente separável se $\gamma_i \geq 0, \forall i$.

Dessa forma, o processo de atualização básica do *Perceptron Estruturado* quando ocorre um erro ($\gamma_i < 0$) é descrito a seguir para um problema relacionado a recompensas. Seja:

$$y^* = argMax_{y \in Y_S, y \neq y_i}\{w^T.f(y)\}. \tag{4.3}$$

onde $w^T.f(y)$ pode ser considerado a recompensa da associação de uma entrada x com uma saída y, lembrando que $f(y) = f(x_i, y)$.

Então, se ocorrer um erro na margem ($\gamma_i < 0$), ou seja, se $w^T.(f(y_i) - f(y^*)) < 0$ então a correção dá-se por:

$$w^{(k+1)} = w^{(k)} + \eta.d_i^{(k)}. \tag{4.4}$$

onde $d_i^{(k)} = f(y_i) - f(y^*)$ da iteração k atual.

A Figura 7 ilustra o caso onde é necessária a correção do vetor w para que o vetor d_i passe a não mais violar a margem. Perceba que d_i deve ficar acima do hiperplano separador para se obter uma margem $\gamma_i > 0$. Observe também a Figura 8, a qual ilustra o caso completo, logo após a correção aqui demonstrada.

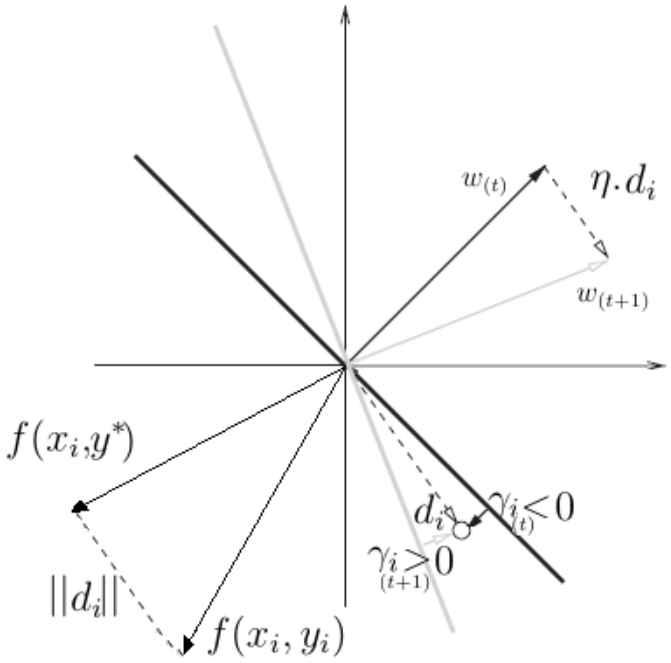

Figura 7 – Interpretação geométrica da correção do vetor w para um caso simples, onde o vetor d_i possui somente duas dimensões.

Note a semelhança em $d_i^{(k)}$ com o processo de correção por erros da Regra Delta (WIDROW; HOFF, 1960), cujo resumo pode ser visto no Apêndice C.

Então, para o cálculo de y^*, as saídas $y \in Y_S$ do conjunto de treinamento S são comparadas entre si e como resultado y^* é a melhor solução analisada advinda desta comparação. Dessa forma tem-se que a **complexidade é linear**, relacionada a quantidade de pares em S. Posteriormente, será mostrado nos experimentos bons resultados associados a esta abordagem.

A Figura 8 ilustra o caso onde não é mais necessária a correção do vetor w, para um caso simples onde $f(y)$ retorna um ponto no plano cartesiano, tendo como exemplos

somente quatro vetores diferenças. Na Figura 8 está representado somente um x_i a fim de facilitar a visualização, mas é necessário que $\gamma_i > 0$ ocorra $\forall x_i$.

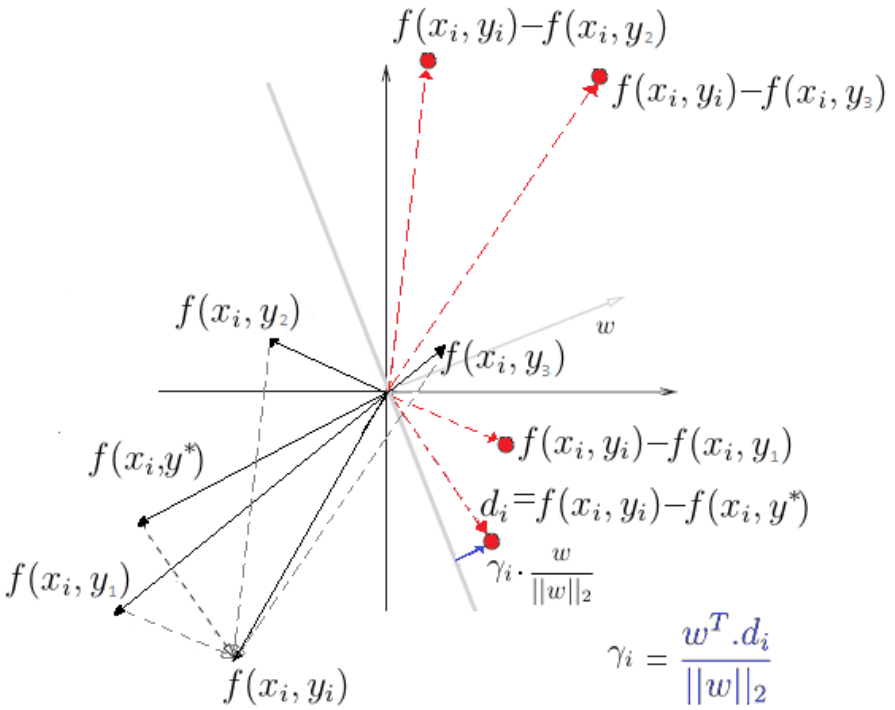

Figura 8 – Interpretação geométrica de γ_i para um caso simples, onde $f(y)$ retorna coordenadas no plano cartesiano.

Se o problema for relacionado a custos (principal forma abordada no decorrer da obra), ele assume a seguinte forma. Seja:

$$y^* = argMin_{y \in Y_S, y \neq y_i}\{w^T.f(y)\}. \tag{4.5}$$

onde $w^T.f(y)$ pode ser considerado o custo da associação de uma entrada x com uma saída y.

Seja o vetor diferença redefinido para cada elemento do conjunto S na forma:

$$d_i = f(y^*) - f(y_i), \tag{4.6}$$

Então, para o cálculo da margem alcançada tem-se:

$$\begin{aligned} \gamma_i &= \frac{Min_{y \in Y_S, y \neq y_i}\{w^T.f(y)\} - w^T.f(y_i)}{||w||_2} \\ &= \frac{w^T.d_i}{||w||_2}, \end{aligned} \tag{4.7}$$

desse modo continua-se requerendo uma margem $\gamma_i \geq 0, \forall i$, isto é, $Min_{y \in Y_S, y \neq y_i}\{w^T.f(y)\} - w^T.f(y_i) \geq 0, \forall i$. Note que w^T e d_i devem possuir a mesma dimensão.

Então se $\gamma_i < 0$, em outras palavras, se $w^T.d_i < 0$, então é necessária a correção, ou seja:

$$w^{(k+1)} = w^{(k)} + \eta.d_i, \tag{4.8}$$

onde $w^T.d_i$ pode ser interpretado como a diferença entre o custo de $w^T.f(y^*)$ e o custo de $w^T.f(y_i)$.

Note que a derivação foi efetuada de modo que a condição para a ocorrência de correção, em ambos os casos (maximização e minimização), é que a margem de separação γ_i seja violada, ou seja, $\gamma_i < 0$. Tal margem foi definida nas equações 4.2 e 4.7, com a importante diferença sendo a troca da ordem das parcelas e consequentemente de seus sinais. O parâmetro η é uma taxa de aprendizado constante ($0 < \eta \leq 1$) e y^* representa sempre a melhor alternativa, desconsiderando o y_i correto, ou seja, $y^* \neq y_i$. O critério de parada é definido através da necessidade de se ter margens positivas para todo o conjunto de treinamento, o qual implica que a correlação (x_i, y_i) está sendo satisfeita.

A regra de correção obtida para o algoritmo *Perceptron Estruturado* para um problema de minimização de custos pode ser facilmente derivada seguindo o raciocínio descrito a seguir (COELHO; NETO; BORGES, 2009).

A condição de viabilidade do algoritmo *Perceptron* é dada por, $\forall i$:

$$\begin{aligned} & w^T.d_i \geq 0 \\ & 0 \geq -w^T.d_i. \end{aligned} \tag{4.9}$$

Desta forma, a função de erro relacionada à estratégia de minimização de custos será dada por:

$$J(w) = \sum_{i=1}^{m} Max\{0, -w^T.d_i\}. \tag{4.10}$$

Esta função deve ser minimizada. Portanto, pode-se definir o gradiente local relativo a *i-ésima* amostra como:

$$\nabla_w J(w) = -d_i = -(f(y^*) - f(y_i)) = f(y_i) - f(y^*). \tag{4.11}$$

Note que a inserção de uma perda l_i na condição de viabilidade 4.9 não influência o gradiente local, desde que seja independente do vetor w.

Assim, caso ocorra um erro relacionada a *i-ésima* amostra, ou seja:

$$\begin{aligned} & w^T.d_i < 0 \\ & w^T.(-d_i) > 0, \end{aligned} \tag{4.12}$$

utiliza-se a seguinte regra de correção para o vetor de custos, com $0 < \eta \leq 1$ e no sentido oposto ao vetor gradiente $\nabla_w J(w)$:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \eta.(-d_i) \\ w_{t+1} &\leftarrow w_t + \eta.d_i. \end{aligned} \tag{4.13}$$

4.2 Perceptron Estruturado com Margem Zero

É possível obter uma variante do *Perceptron Estruturado* padrão, seção 4.1, ao se desconsiderar a obrigatoriedade de $y \neq y_i$, presente na equação 4.6. Neste caso, para o vetor diferença, tem-se mudança somente no cálculo do y^*:

$$d_i = f(y^*) - f(y_i), \tag{4.14}$$

o qual, considerando a equação 4.1 modificada para minimização: $y_i = argMin_{y \in Y_S}\{w^T.f(y)\}$ e a equação 4.5 sem a condição $y \neq y_i$: $y^* = argMin_{y \in Y_S}\{w^T.f(y)\}$, acarretará num $y^* = y_i$, ou seja, é esperado que o y^* ótimo seja justamente o y_i contido no conjunto de treinamento S.

A nova condição de viabilidade do algoritmo será dada por:

$$w^T.d_i = w^T.(-d_i) = 0, \forall i. \tag{4.15}$$

Neste caso, tem-se a seguinte função de erro relacionada à estratégia de minimização de custos:

$$J(w) = \sum_{i=1}^{m} |w^T.d_i|. \tag{4.16}$$

Ou seja:

$$J(w) = \begin{cases} \sum_{i=1}^{m} w^T.d_i; \text{ se } w^T.d_i \geq 0 \\ \sum_{i=1}^{m} -w^T.d_i; \text{ se } w^T.d_i < 0. \end{cases} \tag{4.17}$$

Esta função deve ser minimizada. Portanto, pode-se definir o gradiente local relativo a *i-ésima* amostra como:

$$\nabla_w J(w) = \begin{cases} d_i = \quad (f(y^*) - f(y_i)) = -(f(y_i) - f(y^*)); \text{ se } w^T.d_i > 0 \\ -d_i = -(f(y^*) - f(y_i)) = \quad (f(y_i) - f(y^*)); \text{ se } w^T.d_i < 0. \end{cases} \tag{4.18}$$

Assim, caso ocorra um erro relacionada a *i-ésima* amostra, ou seja:

$$w^T.d_i \neq 0 \tag{4.19}$$

utiliza-se a seguinte regra de correção para o vetor de custos, com $0 < \eta \leq 1$ e no sentido oposto ao vetor gradiente $\nabla_w J(w)$:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \eta.d_i; \text{ se } w^T.d_i > 0 \\ w_{t+1} &\leftarrow w_t + \eta.d_i; \text{ se } w^T.d_i < 0, \end{aligned} \tag{4.20}$$

ou seja,

$$w_{t+1} \leftarrow w_t - sinal(w^T.d_i).\eta.d_i, \tag{4.21}$$

onde o operador *sinal* retorna 1 caso $w^T.d_i > 0$ e -1 caso $w^T.d_i < 0$.

Uma constatação importante é que neste caso perde-se os meios usuais de se trabalhar com o aumento da margem, visto que a minimização do erro através do processo de correção ocasionará um valor de margem igual a zero, conforme pode ser verificado comparando-se a equação de margem 4.7 com a seguinte equação modificada com a não exigência de $y \neq y_i$:

$$\gamma_i = \frac{Min_{y \in Y_S}\{w^T.f(y)\} - w^T.f(y_i)}{||w||_2} = \frac{0}{||w||_2} = 0 \tag{4.22}$$

Esta variante pode ser entendida como um novo objetivo de se obter sempre uma margem igual a zero, desejando-se $y^* = y_i$ para cada elemento do conjunto S, podendo ser usada no caso em que é preciso descobrir o vetor w para certas correspondências exatas entre os objetos estruturados. Este modelo de margem zero funciona mesmo que o conjunto S tenha um único par (x_i, y_i), aliás este é o caso mais indicado, desse modo o y^* pode ser calculado como $w^T.x_i$ e enquanto $y^* \neq y_i$ significa que o vetor w precisa de correção. É fácil observar que a medida que o número de elementos em S cresce, fica mais difícil se obter $|w^T.d_i| = 0, \forall i$, isso porque essa condição é altamente restritiva, frequentemente ocasionando a inviabilidade de qualquer w mesmo para um S pequeno. Neste caso, poder-se-ia permitir um erro ε ao invés de se forçar sempre uma margem zero para todos o elementos, ou seja, a condição de viabilidade seria dada por $|w^T.d_i| = \varepsilon$. Ora, note que essa é a base para algoritmos de regressão. No entanto, o restante de sua formulação e exemplos serão deixados para trabalhos futuros.

4.3 Perceptron Estruturado com Margem

Através da união da teoria do *Perceptron com Margem* visto na seção 1.2 e do *Perceptron Estruturado* padrão, seção 4.1, se obtém a regra de correção para o algoritmo *Perceptron Estruturado com Margem*. Ela pode ser derivada a partir do desenvolvimento a seguir.

Seja o mesmo vetor diferença definido na equação 4.6: $d_i = f(y^*) - f(y_i)$. A condição de viabilidade do algoritmo *Perceptron* agora é dada por (observe novamente as

definições de margem nas equações 3.3 e 3.10), $\forall i$:

$$\begin{aligned} &\gamma_i \geq \gamma. \\ &w^T.d_i \geq \gamma.||w||_2. \\ &w^T.d_i - \gamma.||w||_2 \geq 0, \end{aligned} \tag{4.23}$$

onde γ é um valor determinado *a priori*.

A Figura 9 ilustra para o caso mais simples, no plano cartesiano, a influência da margem γ no processo de correção do vetor w.

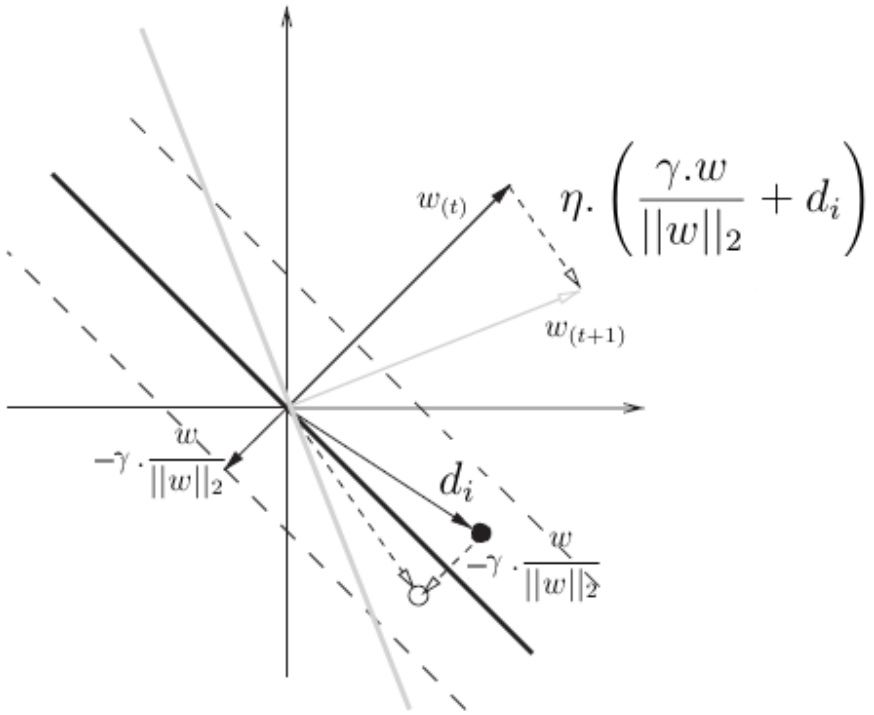

Figura 9 – Interpretação geométrica da correção do vetor w para um caso simples, onde o vetor d_i possui somente duas dimensões.

Desta forma, a função de erro relacionada à estratégia de minimização de custos será dada por:

$$J(w) = \sum_{i=1}^{m} Max\{0, \gamma.||w||_2 - w^T.d_i\}. \tag{4.24}$$

Portanto, pode-se definir o gradiente local, em relação ao vetor w, relativo a *i-ésima* amostra como:

$$\nabla J(w) = \frac{\gamma.w^T}{||w||_2} - d_i = \frac{\gamma.w^T}{||w||_2} + f(y_i) - f(y^*). \tag{4.25}$$

Assim, caso ocorra um erro relacionada a *i-ésima* amostra, ou seja:

$$\begin{aligned} &w^T.d_i < \gamma.||w||_2 \\ &w^T.(-d_i) > -\gamma.||w||_2, \end{aligned} \tag{4.26}$$

utiliza-se a seguinte regra de correção para o vetor de custos, com $0 < \eta \leq 1$:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \eta.\left(\frac{\gamma.w}{||w||_2} - d_i\right) \\ w_{t+1} &\leftarrow w_t.\left(1 - \frac{\eta.\gamma}{||w||_2}\right) - \eta.(-d_i) \\ w_{t+1} &\leftarrow w_t.\left(1 - \frac{\eta.\gamma}{||w||_2}\right) + \eta.d_i. \end{aligned} \tag{4.27}$$

A correção do vetor w poderia ter o sentido alterado caso $\left(1 - \frac{\eta.\gamma}{||w||_2}\right) \leq 0$, no entanto, em (LEITE; NETO, 2007) foi demonstrado que para o algoritmo *Perceptron com Margem Fixa* tem-se $||w||_2 > \eta.\gamma$. Uma importante constatação é que a derivação em (LEITE; NETO, 2007) é análoga para o caso estruturado, basta substituir $y_i\langle x_i, w\rangle$ por $\langle d_i, w\rangle$ e o vetor $y_i x_i$ pelo vetor d_i na prova em questão. Como consequência, o teorema de convergência do *Perceptron de Margem Fixa* também estende-se para a sua versão estruturada. Desse modo tem-se o número de iterações t:

$$t \leq \frac{R^2 - \gamma^2}{(\gamma_i(w^*) - \gamma)^2}, \tag{4.28}$$

onde $R = Max_{i\in\{1,\ldots,m\}}||d_i||$ e $\gamma_i(w^*) = Max_w(\gamma_i(w)),\ \forall w$ viável.

Existe a possibilidade de se acrescentar uma função $l_i = f_{tam}(y^*) - f_{tam}(y_i), \forall i$ nas formulações apresentadas a fim de incluir uma perda específica em $w.d_i, \forall i$ dependendo das característica estruturais de cada y^* e y_i. Ou seja, a condição de viabilidade redefinida para cada par i do conjunto S toma a forma:

$$\begin{aligned} w^T.d_i &\geq \gamma.||w||_2 + (-l_i) \\ w^T.d_i + l_i &\geq \gamma.||w||_2 \\ w^T.(f(y^*) - f(y_i)) + f_{tam}(y^*) - f_{tam}(y_i) &\geq \gamma.||w||_2 \\ (w^T.f(y^*) + f_{tam}(y^*)) - (w^T.f(y_i) + f_{tam}(y_i)) &\geq \gamma.||w||_2, \end{aligned} \tag{4.29}$$

onde f_{tam} pode ser considerado um custo intrínseco relacionado a uma estrutura y independente do vetor de custos w^T e que $(w^T.f(y) + f_{tam}(y))$ é o custo total relacionado a um par estrutural qualquer (x, y).

É importante ressaltar que ao se alterar dessa maneira a condição de viabilidade, não se altera nenhuma das derivações para as regras de correção vistas até aqui, pois o gradiente da função de erro $\nabla J(w)$ permanece inalterada. O único ponto a ser reavaliado é o cálculo do y^*, o qual é fácil ver que pode ser reescrito como:

$$y^* = argMin_{y\in Y_S, y\neq y_i}\{w^T.f(y) + f_{tam}(y)\}. \tag{4.30}$$

Ao término do aprendizado, é bem provável que os valores de cada $\gamma_i = w^T.d_i/||w||_2$ para cada entrada em S, calculado conforme a equação 3.10, não sejam exatamente iguais

ao γ, e nem é preciso, lembrando que a condição de viabilidade é $w^T.d_i/||w||_2 \geq \gamma$. Neste caso, a margem de separação final alcançada pelo aprendizado pode ser calculada conforme a equação 3.12: $\gamma_z = Min\{\gamma_i\}, \forall i$. Observe que $\gamma_z \geq \gamma$.

4.4 Perceptron Estruturado com Margem Incremental

Usando uma abordagem baseada no IMA, seção 1.3, foi desenvolvida uma estratégia específica para o caso estruturado. Para o cálculo aproximado do valor da máxima margem de parada adota-se uma estratégia de incremento na qual sucessivos sistemas de inequações 4.23 são resolvidos para valores crescentes do parâmetro γ.

Executa-se inicialmente o algoritmo padrão do *Perceptron Estruturado* e calcula-se a margem correspondente obtida. Lembrando-se que a margem final de cada iteração do *Perceptron Estruturado com Margem* é $\gamma_z = Min\{\gamma_i\}, \forall i$, conforme equação 3.12. Tem-se então o primeiro γ_z para a entrada do processo de incremento, sendo, logicamente, esse γ_z inicial viável. O valor do parâmetro seguinte é obtido incrementando-se (multiplicando-se ou somando-se uma taxa δ fixa) o valor da margem de parada anterior computado. Pode então ocorrer dois casos distintos.

1 - Executa-se o algoritmo do *Perceptron Estruturado com Margem*, esta nova margem incrementada continua sendo viável e o algoritmo prossegue com os incrementos sucessivos.

2 - Após uma sequência de incrementos, na execução do algoritmo do *Perceptron Estruturado com Margem*, verifica-se que o novo valor de γ_z não permite a convergência do algoritmo, ou seja, supera o valor da margem máxima admissível. Dessa forma, ficam determinados dois limites de margem, um inferior, com a última margem γ_z^- viável, e um superior, com a margem γ_z^+ inviável. Nesta etapa, determina-se a margem como $(\gamma_z^- + \gamma_z^+)/2$ verificando a viabilidade, se a margem for inviável, repete-se o procedimento, diminuindo continuamente o valor da margem γ_z^+ inviável, até alcançar uma margem viável ou até um limite máximo de iterações T. Caso encontre uma nova margem viável, tem-se um novo limite inferior e pode-se executar novamente o mesmo processo.

A diferença entre γ_z^- e γ_z^+ diminui rapidamente, pois a ordem de complexidade é a mesma de uma busca binária. Porém, como a busca é feita num espaço $\mathbb{R}$ infinito, teoricamente tem-se infinitos elementos no conjunto a ser pesquisado, no entanto, é possível definir uma margem de erro Υ onde $\gamma_z^+ - \gamma_z^- < \Upsilon$. Foi constatado experimentalmente que mesmo para valores muito pequenos de Υ, na ordem de 10^{-4} e 10^{-5}, dependendo do experimento em questão, o algoritmo convergia para este limite geralmente em algumas dezenas e, no máximo, em poucas centenas de execuções do *Perceptron Estruturado com Margem*. Vale observar que os valores do vetor de custo w referentes à solução de cada problema são retidos e servem como solução inicial para o problema posterior.

Ou seja, ao resolver o problema de forma incremental para valores fixos do parâmetro de margem, tem-se como objetivo a maximização da margem γ:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &w^T.d_i \geq \gamma.||w||_2, \end{aligned} \tag{4.31}$$

Considere $Max\ \gamma = \gamma^*$. Tem-se que a margem real alcançada é viável e contém a seguinte relação: $\gamma_z^- > \gamma^* - \Upsilon$.

4.5 O Método de Subgradiente

Este método foi usado no algoritmo *Maximum Margin Planning* (*MMP*) de (RATLIFF; BAGNELL; ZINKEVICH, 2006), no intuito de obter custos para determinadas características do ambiente e deste modo auxiliar no planejamento de caminhos, como será visto na seção 5.3. Foi abordado no trabalho como fator de comparação com os outros métodos já expostos.

Considerando a abordagem apresentada na seção 3.2, em especial na equação 3.18, onde $||w||_2$ deve ser minimizada para maximizar-se a margem, sua formulação é transcrita aqui como:

$$\begin{aligned} &Min\ \frac{1}{2}\|w\|^2 \\ &Sujeito\ a: \\ &w^T.f(y_i) \geq Max_{y \in Y_i}\{w^T f(y) + l_i(y)\}. \end{aligned} \tag{4.32}$$

Para a solução deste problema de maximização de recompensas, (RATLIFF; BAGNELL; ZINKEVICH, 2006) propõem a minimização de uma função objetiva não diferenciável mas convexa, obtida da relaxação do problema de programação quadrática acima. Assim deve-se minimizar a função:

$$L(w) = \frac{1}{2}||w||^2 + C.\sum_i Max_{y \in Y_i}\{w^T f(y) + l_i(y)\} - w^T.f(y_i), \tag{4.33}$$

Isto é feito através de uma técnica de subgradiente (para mais detalhes veja Apêndices E e D):

$$g = w + C.(\sum_i f(y^*) - f(y_i)), \tag{4.34}$$

onde $y^* = Max_{y \in Y_i}\{w^T f(y) + l_i(y)\}$, ou seja, o espaço de saída para o ótimo corresponde ao conjunto Y_i, espaço de todas as possíveis saídas dado o objeto estruturado x_i.

Assim, o vetor w fica atualizado na forma:

$$\begin{aligned} w &\leftarrow w - \eta_v.g \\ w &\leftarrow w.(1-\eta) - \eta_v.(C.(\sum_i f(y^*) - f(y_i))), \end{aligned} \tag{4.35}$$

onde η_v é uma taxa de aprendizado que é reduzida de forma gradativa de acordo com o número de iterações e C é um parâmetro de regularização positivo.

Nesta abordagem, as equações são computadas em função da determinação da política ótima $y^* = Max_{y \in Y_i}\{w^T f(y) + l_i(y)\}$, o qual diferentemente do *Perceptron estruturado* e suas variantes, (RATLIFF; BAGNELL; ZINKEVICH, 2006) considera o espaço de saída $Y_i \neq Y_S$ como o espaço completo de todas as saídas possíveis dado um x_i, para o problema em questão.

4.6 Perceptron Estruturado Dual

A formulação dual do problema de predição estruturada se baseia na utilização da forma expandida do vetor w, ou seja, w é representado como uma combinação linear dos vetores característicos. Inicialmente optou-se pela expansão do vetor w utilizando-se os vetores diferença retratando de maneira mais próxima a forma como é realizado na formulação padrão dual do modelo Perceptron, vista na seção 1.4. Na seção seguinte será visto também uma expansão dessa abordagem que permite o uso de funções *Kernel*.

Seja inicialmente o problema em sua forma primal, $\forall y \in Y_S$:

$$w^T.f(y) - w^T.f(y_i) \geq \gamma.||w||_2. \tag{4.36}$$

Seja o vetor diferença definido como: $d(y) = f(y) - f(y_i), \forall y \in Y_S$, então:

$$w^T.d(y) \geq \gamma.||w||_2. \tag{4.37}$$

Analisando as equações 4.13 e 4.27, referentes a atualização do vetor w, respectivamente sem margem e com margem, pode-se notar em ambos os casos que esse vetor nada mais é que uma combinação linear dos vetores diferença d_i. Nos casos do *Perceptron Estruturado com Margem* e do *Perceptron Estruturado com Margem Incremental*, basta considerar a propriedade que uma combinação linear de combinações lineares também é uma combinação linear. Então, o vetor w pode ser expresso como:

$$w = \sum_{k=1}^{m \times m} \alpha_{d_k}.(d_k). \tag{4.38}$$

Onde $\alpha_{d_k} \geq 0$ é o vetor de variáveis duais associadas a cada restrição do problema. Note que, nesta formulação, a quantidade total de vetores diferença $d(y)$ a serem verificados

na inequação 4.39 dá-se por $k = 1, \ldots, m \times m$. Observe também que os valores para $d(y) = d_k$ são fixos e podem ser pré-computados. Deste modo, substituindo a equação 4.38 na formulação 4.37, tem-se o seguinte problema de otimização dual:

$$\forall y \in S : \sum_{k=1}^{m \times m} \alpha_{d_k} . \langle d_k, d_i(y)) \rangle \geq \gamma . ||w||_2; \; i = 1, \ldots, \; m. \tag{4.39}$$

Se resolvido de forma incremental para valores fixos do parâmetro de margem, tem-se como objetivo:

$$\begin{aligned} & Max \; \gamma \\ & Sujeito \; a : \\ & \forall y \in S : \sum_{k=1}^{m \times m} \alpha_{d_k} . \langle d_k, d_i(y)) \rangle \geq \gamma . ||w||_2; \; i = 1, \ldots, \; m. \end{aligned} \tag{4.40}$$

Uma abordagem similar foi proposta por (TASKAR et al., 2005), o qual utiliza para solução uma adaptação do método *SMO* (*Sequential Minimal Optmization*), de (PLATT, 1999). Também, (RATLIFF et al., 2007), com sua teoria estendida em (RATLIFF, 2009), propõe a obtenção de um conjunto de características ampliado alterando a dimensão do espaço de entrada. Emprega uma técnica conhecida como *Structured Boosting*, derivada principalmente do trabalho de (FRIEDMAN, 2000), tal algoritmo foi denominado como *MMP Boost*. Uma rápida e resumida explicação dessas abordagens pode ser vista no apêndice G.

A formulação anterior padece do problema relacionado ao número exponencial de restrições, visto também em 3.3, porém aqui há um complicador a mais, o número também exponencial de variáveis. Neste sentido propõem-se uma técnica de geração que decompõem o problema anterior na solução de um problema mestre e de vários subproblemas de otimização associados a cada classe.

Primeiramente, tem-se o vetor diferença para cada classe em função do mínimo argumento relacionado a escolha dos falsos-exemplos.

$$d_i(y^*) = f(y^*) - f(y_i). \tag{4.41}$$

Note que agora a quantidade total de vetores diferença $d_i(y^*)$ foi reduzida para m, um para cada i.

Considere $S = \{(x_i, y_i)\} = \{(x_j, v_j)\}; \; i, j = 1, \ldots, \; m$. Note que é o mesmo conjunto S visto durante toda a obra, porém, agora os índices i e j podem variar numa mesma formulação, por exemplo na equação 4.42, o qual trabalha com os dois índices simultaneamente.

Assim, para cada subproblema de otimização, obtêm-se o y ótimo:

$$y^* = argMin_{y \in Y_S, y \neq y_i}\{w^T.f(y)\}.$$
$$y^* = argMin_{y \in Y_S, y \neq y_i, v^* \in Y_S, v^* \neq v_j}\left\{\sum_{j=1}^{m} \alpha_{v_j}.\langle d_j(v^*), f(y)\rangle\right\}. \tag{4.42}$$

onde v^* é justamente o y^* computado na iteração anterior. Deste modo, cada v^* é a escolha ótima para seu próprio índice j. Note que agora tem-se um único α para cada elemento correspondente no conjunto S.

Em seguida, resolve-se o problema mestre restrito somente ao sub-conjunto dos vetores diferença ótimos computados em cada subproblema para um dado vetor w:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &w^T.d_i(y^*) \geq \gamma.||w||_2;\ i = 1,\ \ldots,\ m. \end{aligned} \tag{4.43}$$

Que na sua forma dual apresenta-se como:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\sum_{j=1}^{m} \alpha_{v_j}.\langle d_j(v^*).d_i(y^*)\rangle \geq \gamma.||w||_2, i = 1,\ \ldots,\ m. \end{aligned} \tag{4.44}$$

onde se $i = j$ então $d_j(v^*) = d_i(y^*)$.

O vetor w também pode ser reescrito como:

$$w = \sum_{j=1}^{m} \alpha_{v_j}.(d_j(v^*)). \tag{4.45}$$

Note que isso representa implicitamente que para cada $d_k \neq d_j(v^*), \forall j$ tem-se $\alpha_{d_k} = 0$.

Observa-se que agora o conjunto de variáveis duais está restrito ao número de classes, assim como o número de restrições. Cada solução de um novo problema mestre fornecerá um novo vetor α que deverá ser repassado aos subproblemas de otimização. Considerando a condição imposta pelas restrições de minimização, o número máximo de vetores diferença que serão suportes na solução ótima do problema mestre ficam limitados pela quantidade de classes ou de restrições.

Note que o vetor α atua como o peso que cada vetor $d_i(y^*)$ possui durante o processo de aprendizado. Depois de concluído o aprendizado, o vetor α refletirá o quanto cada par (x_i, y_i) influência para maximizar a margem γ. Isso implica que para valores de $\alpha_{y_i} = 0$, o correspondente (x_i, y_i) não tem qualquer influência na margem alcançada e os correspondentes (x_i, y_i) onde $\alpha_{y_i} \neq 0$ seriam os suportes da margem.

A norma quadrática do vetor w pode ser computada a partir do vetor α de variáveis duais. Ou seja:

$$||w||_2 = \sqrt{w.w^T} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{y_i}.\alpha_{v_j}.\langle d_i(y^*), d_j(v^*)\rangle}. \tag{4.46}$$

O vetor w final pode ser expresso a partir do vetor ótimo de variáveis duais e dos respectivos vetores diferença:

$$w = \sum_{i=1}^{m} \alpha_{y_i}.d_i(y^*). \tag{4.47}$$

Como consequência da solução dos subproblemas, o sub-conjunto de vetores diferença que formam o conjunto de treinamento do problema mestre é alterado a cada iteração.

Observe que a equação 4.47 assegura que somente os vetores diferença $d_i(y^*)$ terão contribuição na solução final de w e já que w está sendo representado segundo a equação 4.45 como uma combinação linear dos vetores diferença $d_i(y^*)$, então para um w_{t+1} da iteração seguinte a w_t, conforme equação 4.13 no *Perceptron Estruturado*, com $\alpha_t(v_j) = \alpha_{v_j}$, tem-se:

$$\sum_{j=1}^{m} \alpha_{t+1}(v_j).d_j(v^*) \leftarrow \sum_{j=1}^{m} \alpha_t(v_j).d_j(v^*) + \eta.d_i(y^*). \tag{4.48}$$

Lembrando que os valores dos vetores diferenças são todos constantes, o que pode mudar é o v^*, suponha que começou-se com um v_1^* e durante o aprendizado ele deixou de ser $ArgMin$ da equação 4.42. O novo ótimo, chamemos de v_2^*, representa unicamente que outro vetor $d_{k2} = f(v_2^*) - f(v_j) = d_j(v_2^*)$ é que passou a ter um $\alpha(d_{k2}) = \alpha(v_2^*) \neq 0$ e tornou-se o novo $d_j(v^*)$, enquanto o $d_j(v_1^*)$ anterior passou a ter um α associado igual a zero, ou seja, na verdade a mudança é no α, pois como já mencionado, para um $d_k \neq d_j(v^*), \forall j$ tem-se automaticamente $\alpha_{d_k} = 0$. Note que o mesmo vale para $d_i(y^*)$. Então a correção do valor de cada variável dual é dada pela expressão:

$$\alpha_{t+1}(y_i) \leftarrow \alpha_t(y_i) + \eta.1, \tag{4.49}$$

ou seja, quando passa-se da iteração t para $t+1$, altera-se os α's em função de uma taxa de aprendizado η. Observe que a correção independe do sinal do próprio $d_i(y^*)$. Seja $d_i(y^*) < 0$ então o α_{y_i} relacionado a ele será incrementado ocasionando um peso maior para o vetor negativo $d_i(y^*)$ e o somatório $\sum_{j=1}^{m} \alpha_{t+1}(v_j).d_j(v^*) = w_{t+1}$ será decrementado. Agora Seja $d_i(y^*) > 0$ então o α_{y_i} relacionado a este dado será incrementado ocasionando um peso maior para o vetor positivo $d_i(y^*)$ e o somatório $\sum_{j=1}^{m} \alpha_{t+1}(v_j).d_j(v^*) = w_{t+1}$ será incrementado. Atentando-se para o fato que se $i = j$ então $d_j(v^*) = d_i(y^*)$.

Porém, a correção para o dual levando em conta a margem, correspondente ao *Perceptron Estruturado com Margem* da seção 4.3, não é tão direta quanto a anterior em

virtude da entrada própria margem γ na formulação, sendo agora necessário escalonar o vetor α pelo fator $\left(1 - \frac{\eta.\gamma}{||w||_2}\right)$ a cada correção do próprio α. Avaliando novamente a equação 4.47 nota-se que segundo a formulação dual:

$$w_t.\left(1 - \frac{\eta.\gamma}{||w||_2}\right) = \sum_{j=1}^{m}\left(1 - \frac{\eta.\gamma}{||w||_2}\right).\alpha_t(v_j).(d_j(v)) \tag{4.50}$$
$$w_{t+1} \leftarrow \sum_{j=1}^{m}\left(1 - \frac{\eta.\gamma}{||w||_2}\right).\alpha_t(v_j).(d_j(v)) + \eta.d_i(y^*).$$

O raciocínio para a correção do vetor α considerando-se a margem é quase análogo à correção sem considerar a margem apresentada em 4.48 e 4.49, com a única diferença que todos os valores de α_{v_j} são escalonados a cada correção. Uma vez que $\left(1 - \frac{\eta.\gamma}{||w||_2}\right).\alpha_{v_j}$ é a parcela não constante nesta correção, tem-se:

$$\alpha_{t+1}(v_j) \leftarrow \left(1 - \frac{\eta.\gamma}{||w||_2}\right)\alpha_t, \forall\alpha \tag{4.51}$$
$$\alpha_{t+1}(y_i) \leftarrow \alpha_t(y_i) + \eta.1.$$

O fator escalonador traz um efeito interessante. Ao decrementar o valor da cada multiplicador por um fator antes de cada atualização, tem-se em essência a escolha dos vetores suporte (LEITE; NETO, 2007). Em outras palavras, aqueles que não serão os vetores suportes finais, precisam ser corrigidos poucas vezes e, eventualmente, seus valores vão para zero.

4.7 Perceptron Estruturado Dual com *Kernel*

Unindo a formulação anterior, seção 4.6, com o conceito de funções *Kernel*, seção 1.5, tem-se a base teórica necessária para o desenvolvimento do *Perceptron Estruturado Dual com Kernel*.

Uma extensão natural do modelo dual seria a tentativa de usar uma função *Kernel* diretamente na formulação 4.44, de forma idêntica ao problema padrão do *Perceptron Dual*, apresentando-se como:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\sum_{j=1}^{m}\alpha_{v_j}.K(d_j(v^*), d_i(y^*)) \geq \gamma.||w||_2,\ i = 1,\ \ldots,\ m. \end{aligned} \tag{4.52}$$

Desse modo, com $K(d_j(v^*), d_i(y^*)) = \langle\theta(d_j(v^*)), \theta(d_i(y^*))\rangle$, tem-se:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\sum_{j=1}^{m}\alpha_{v_j}.\langle\theta(d_j(v^*)), \theta(d_i(y^*))\rangle \geq \gamma.||w||_2,\ i = 1,\ \ldots,\ m. \end{aligned} \tag{4.53}$$

Para uma transformação linear θ, tem-se a possibilidade da utilização dos próprios vetores diferença, trazendo uma simplificação no processo de mapeamento *Kernel*. Assim, considerando o equacionamento proposto, tem-se:

$$\begin{aligned} & K(d_i(y*), d_j(v*)) \\ =& K(f(y_i) - f(y*), f(v_j) - f(v*)) \\ =& \langle \theta(f(y_i)) - \theta(f(y_j)), \theta(f(v_j)) - \theta(f(v*)) \rangle \\ =& (\langle \theta(d_i(y^*)), \theta(d_j(v^*)) \rangle) \\ =& k(\langle d_i(y^*), d_j(v^*) \rangle) \end{aligned} \tag{4.54}$$

Com a possibilidade de $\theta f(y_i) = f(y_i)$, a forma padrão adotada na literatura (WESTON; SCHÖLKOPF; BOUSQUET, 2005), tem-se simplesmente:

$$k(\langle d_i(y^*), d_j(v^*) \rangle) = \langle d_i(y^*), d_j(v^*) \rangle. \tag{4.55}$$

Entretanto, a abordagem acima 4.54 só é correta para um θ linear, pois o vetor diferença é invariante em relação a ele. Porém, para um θ que não seja linear, tal igualdade $\langle \theta(f(y_i)) - \theta(f(y_j)), \theta(f(v_j)) - \theta(f(v*)) \rangle = \langle \theta(d_i(y^*)), \theta(d_j(v^*)) \rangle$ não está correta. Por exemplo, para um θ polinomial quadrático é fácil visualizar que o quadrado das diferenças é diferente da diferença dos quadrados. Será abordado a seguir como superar essa limitação.

Existem duas formas clássicas de introdução do mapeamento *Kernel* em modelos de predição estruturada. Ambas utilizam a definição de *joint Kernel*, uma adaptação de *Kernel* para problemas de altas dimensões com dependências entre os dados.

Segundo (WESTON; WATKINS, 1998): *joint Kernel* é uma metodologia para resolver problemas de dependência entre pares de dados em altas dimensões, o que é possível no caso em que a saída de interesse tem dimensão elevada, por exemplo, milhares de dimensões. Isto é conseguido através do mapeamento dos objetos em espaços contínuos e discretos. Entre correlações conhecidas de entrada e saída podem ser definidos diferentes tipos de *Kernel*, alguns dos quais podem manter linearidade na saída.

Sua derivação matemática e algumas aplicabilidades podem ser encontradas neste mesmo trabalho (WESTON; WATKINS, 1998). Um exemplo de *joint Kernel* linear aplicado em um problema com saída estruturada pode ser encontrado em (TSOCHANTARIDIS et al., 2004). Aqui restringe-se às definições apresentadas a seguir.

Primeiramente pode-se considerar o produto interno em separado dos vetores de entrada e dos vetores de saída que formam os vetores característicos, ou seja:

$$J((x_i, y_i), (x_j, y_j)) = K(x_i, x_j).K^-(y_i, y_j) = k(\langle x_i, x_j \rangle).k^-(\langle y_i, y_j \rangle), \tag{4.56}$$

onde J se refere ao *joint Kernel*. As funções k e k^- são relacionadas à função de mapeamento e K e K^- aos respectivos valores obtidos.

A segunda alternativa é considerar a utilização dos vetores característicos em sua forma explícita. O *joint Kernel* apresentado em (WESTON; WATKINS, 1998) nada mais é que o produto interno de pares das funções $f()$:

$$J((x_i, y_i), (x_j, y_j)) = f(x_i, y_i)^T . f(x_j, y_j) = \langle f(x_i, y_i), f(x_j, y_j) \rangle. \tag{4.57}$$

Porém, neste trabalho considera-se que a função f pode ser modificada através de uma função θ, consequentemente será redefinida a função J como:

$$\begin{aligned} J((x_i, y_i), (x_j, y_j)) &= \theta(f(x_i, y_i))^T . \theta(f(x_j, y_j)) \\ J(y_i, y_j) &= \theta(f(y_i))^T . \theta(f(y_j)) = \langle \theta(f(y_i)), \theta(f(y_j)) \rangle = k\langle f(y_i), f(y_j) \rangle. \end{aligned} \tag{4.58}$$

considerando a simplificação $(x_i, y_i) = y_i$, conforme já definido nas seções anteriores.

Deve-se agora considerar a expansão dos vetores diferença, apresentada também em (WESTON; WATKINS, 1998), e aplicar na função *Kernel* o produto interno dos vetores mapeados. Então, deriva-se a seguinte expressão:

$$\begin{aligned} & K(d_i(y^*), d_j(v^*)) \\ =& K(f(y_i) - f(y^*), f(v_j) - f(v*)) \\ =& \langle \theta(f(y_i)) - \theta(f(y_j)), \theta(f(v_j)) - \theta(f(v*)) \rangle \\ =& \langle \theta(f(y_i)), \theta(f(v_j)) \rangle - \langle \theta(f(y^*)), \theta(f(v_j)) \rangle - \langle \theta(f(y_i)), \theta(f(v^*)) \rangle + \langle \theta(f(y^*)), \theta((f(v^*)) \rangle \\ =& k(\langle f(y_i), f(v_j) \rangle) - k(\langle f(y^*), f(v_j) \rangle) - k(\langle f(y_i), f(v^*) \rangle) + k(\langle f(y^*), (v^*) \rangle), \\ =& J(y_i, v_j) - J(y^*, v_j) - J(y_i, v^*) + J(y^*, v^*). \end{aligned} \tag{4.59}$$

A partir dessa derivação é possível apresentar a formulação do modelo *Perceptron Estruturado Dual* para uma função *Kernel* da seguinte forma:

$$\begin{aligned} & Max\ \gamma \\ & Sujeito\ a: \\ & \forall y_i \in S : \sum_{j=1}^{m} \alpha_{v_j} . K(d_j(v^*), d_i(y^*)) \geq \gamma . ||w||2. \end{aligned} \tag{4.60}$$

Considere que todas as formulações estejam sendo conduzidas no espaço de *Hilbert*. Ao se mapear cada parcela do vetor diferença diretamente através da função θ pode ser definido um novo vetor diferença como:

$$\Delta_i(y^*) = \theta(f(y^*)) - \theta(f(y_i)). \tag{4.61}$$

O vetor w pode então ser expresso a partir do vetor de variáveis duais α_{v_j} e dos respectivos vetores diferença:

$$w = \sum_{j=1}^{m} \alpha_{v_j} . \Delta_j(v^*). \tag{4.62}$$

Note que é possível calcular o y^*, visto na equação 4.5, em função do *Kernel* sem precisarmos utilizar a função θ diretamente:

$$\begin{aligned} y^* &= argMin_{y\in Y_S, y\neq y_i}\{w^T.f(y)\}. \\ &= argMin_{y\in Y_S, y\neq y_i, v^*\in Y_S, v^*\neq v_j}\left\{\sum_{j=1}^{m}\alpha_{v_j}.\Delta_j(v^*).\theta f(y)\right\} \\ &= argMin_{y\in Y_S, y\neq y_i, v^*\in Y_S, v^*\neq v_j}\left\{\sum_{j=1}^{m}\alpha_{v_j}.(\theta(f(v^*)) - \theta(f(v_j))).\theta f(y)\right\} \\ &= argMin_{y\in Y_S, y\neq y_i, v^*\in Y_S, v^*\neq v_j}\left\{\sum_{j=1}^{m}\alpha_{v_j}.k(\langle f(v_j), f(y)\rangle) - k(\langle f(v^*), f(y)\rangle)\right\} \\ &= argMin_{y\in Y_S, y\neq y_i, v^*\in Y_S, v^*\neq v_j}\left\{\sum_{j=1}^{m}\alpha_{v_j}.J(v_j, y) - J(v^*, y)\right\}. \end{aligned} \tag{4.63}$$

Sendo assim, a formulação do modelo *Perceptron Estruturado Dual com Kernel* torna-se:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\forall y_i \in S: \sum_{j=1}^{m}\alpha_{v_j}.\langle\Delta_j(v^*), \Delta_i(y^*)\rangle \geq \gamma.||w||_2. \end{aligned} \tag{4.64}$$

Observe as igualdades em 4.59, podemos reescrever a formulação como:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\forall y_i \in S: \\ &\sum_{j=1}^{m}\alpha_{v_j}.(k(\langle f(y_i), f(v_j)\rangle) - k(\langle f(y^*), f(v_j)\rangle) - k(\langle f(y_i), f(v^*)\rangle) + k(\langle f(y^*), (v^*)\rangle)) \geq \gamma.||w||_2, \end{aligned} \tag{4.65}$$

ou ainda,

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\forall y_i \in S: \sum_{j=1}^{m}\alpha_{v_j}.(J(y_i, v_j) - J(y^*, v_j) - J(y_i, v^*) + J(y^*, v^*)) \geq \gamma.||w||_2. \end{aligned} \tag{4.66}$$

É importante frisar que a correção do vetor α permanece a mesma da equação 4.51, pois a mudança vista na equação 4.62 permanece com somente o α escalonado, não sendo constante ao longo do aprendizado.

Note que agora pode-se utilizar uma função k não-linear, por exemplo uma função quadrática, mesmo sem que θ seja conhecido. É importante perceber que não conhecendo-se o θ não é possível calcular o vetor w, contudo, atenta-se também que não é preciso

conhecer o vetor w para o cálculo de $w^T.f(y)$ conforme visto nas equações em 4.63. Além disso, mostrou-se que conhecer o $w^T.f(y)$ é suficiente para o aprendizado e, conforme veremos, por exemplo, no capítulo 5, também é suficiente para a aplicação desse aprendizado em problemas variados.

Finalizando, a utilização do *Kernel* possibilita o aprendizado em ambientes estruturados não-lineares, porém a própria constatação que um problema estruturado específico é ou não-linearmente separável já representa dificuldade. Talvez, uma maneira prática de se visualizar isto seja executando o problema na forma primal, se o mesmo não apresentar solução num tempo viável, provavelmente o problema não é linearmente separável. Caso este mesmo problema seja resolvido rapidamente na forma dual para algum *Kernel* não-linear, confirma-se, pelo menos empiricamente, de que o problema não é linearmente separável.

5 Predição de Dados Estruturados em Planejamento de Caminhos

Um problema que abrange diversos conceitos é o de encontrar o caminho mínimo entre dois nós de um grafo ou de uma rede, sendo considerado um problema clássico da Ciência da Computação. O objetivo consiste, normalmente, em encontrar o caminho de menor custo entre dois nós de uma rede, considerando a soma dos custos associados aos arcos percorridos. Porém, geralmente, a dificuldade maior está em definir os custos de transição entre esses arcos e em se determinar qual é o custo de cada característica relevante presente na transição.

5.1 Proposta da Aplicação

O objetivo é possibilitar a predição de custos em novos ambientes ou mapas tendo como base a predição de dados estruturados definindo um aprendizado funcional entre domínios de entrada e saída, estruturados e variados. Aplicado ao problema de planejamento de caminhos, este aprendizado torna possível a obtenção de planos ou políticas a partir da percepção das características dos mapas, sendo de grande importância, por exemplo, a sua utilização em sistemas de navegação de forma geral. Frequentemente, nestes sistemas de navegação, ocorre uma clara distinção entre os níveis de percepção do ambiente e de planejamento, sendo o planejamento de caminhos obtido somente a partir do prévio conhecimento da matriz de custos relacionada ao espaço de estados-ações do problema. Em resumo, este capítulo se propõe a pesquisar, formular e formalizar como é feita a predição desses custos e como o agente os explora e aprende o caminho.

Neste problema, relacionado à predição de custos, tem-se como dados de entrada um conjunto de caminhos escolhidos por um especialista referentes a determinados mapas. Estes caminhos são escolhidos de forma a beneficiarem algum tipo de estratégia relacionada à presença ou não de determinadas características, servindo de base ou de exemplo na definição do mapeamento de custos de novos mapas. Desta forma, o mapeamento obtido possibilitará o planejamento de novos caminhos em novos ambientes de forma similar ao tipo de estratégia escolhida pelo especialista.

O problema de aprendizado em questão é formulado como um problema de otimização convexa de máxima margem estruturado bastante similar a formulação de máquinas de vetores suporte multi-classe (WESTON; WATKINS, 1998).

A implementação das técnicas de solução apresentadas no capítulo 4 são adapta-

das para esse problema especificamente. Com o intuito de comprovar a eficiência desta abordagem bem como a corretude dos algoritmos implementados, foram realizados diversos testes com diferentes dados de entrada refletidos na escolha de diferentes mapas, caminhos e características.

Depois da descoberta dos custos tem-se o processo de determinação de caminhos. Resolve-se o problema do caminho mínimo através de um processo de exploração denominado busca *forward*, que toma como ponto de partida o estado inicial ou raiz do problema e vai até o estado final. Os resultados transcritos aqui referem-se aos da busca A*, pois é otimamente eficiente (RUSSEL; NORVING, 2004).

Nas próximas seções será abordado a predição de dados estruturados utilizando as técnicas de solução do capítulo 4 e inserindo-as dentro do problema de predição de custos.

5.2 Problema de Predição de Custos

Primeiramente apresenta-se o problema no contexto de maximização de recompensas e em seguida é mostrada sua equivalência com um problema de predição de custos, formulado no contexto de uma coleção de mapas e caminhos. Neste problema cada mapa é representado por um grafo de estados formando um conjunto de pares estados-ações, onde cada estado dos mapas é caracterizado por um conjunto de características.

5.2.1 Equacionamento do Problema de Predição de Custos

Primeiramente deve-se observar como a formulação nó-arco foi aplicada ao problema de predição de custo. Cada caminho é representado por um vetor de frequências em uma matriz $y_i = \mu_i$ que indica a ocorrência ou não das diversas ações ou arcos para cada possível par estado-ação ou transição definido pelo mapa. Cada mapa é representado por uma matriz $x_i = F_i$ de características, distribuída para todo conjunto de pares estado-ações, onde tem-se o seguinte conjunto de treinamento: $S = \{(F_i, \mu_i),\ \forall i\}$.

Note que é possível realizar uma correlação com a estrutura de problemas inversos seguindo a seguinte formulação: $\mu_i = G(w(F_i))$, conforme visto na seção 0.1. Onde μ_i é o caminho, G representa uma função que calcula o caminho de menor custo em um mapa, F_i é a estrutura da matriz do mapa em questão e w é o vetor de custos das características. Porém, não se tem somente um caminho μ_i e um mapa F_i correlacionados por um vetor w. Tem-se na verdade a necessidade de que este vetor w seja condizente para todos os mapas F_i e respectivos caminhos μ_i ao mesmo tempo.

Na Figura 10, pode-se ver uma representação gráfica dessas matrizes.

Assim, o produto $Fi.\mu = Fi\mu$ representa a quantidade de cada característica presente no i-ésimo mapa em função da escolha do caminho representado por μ. Finalmente o

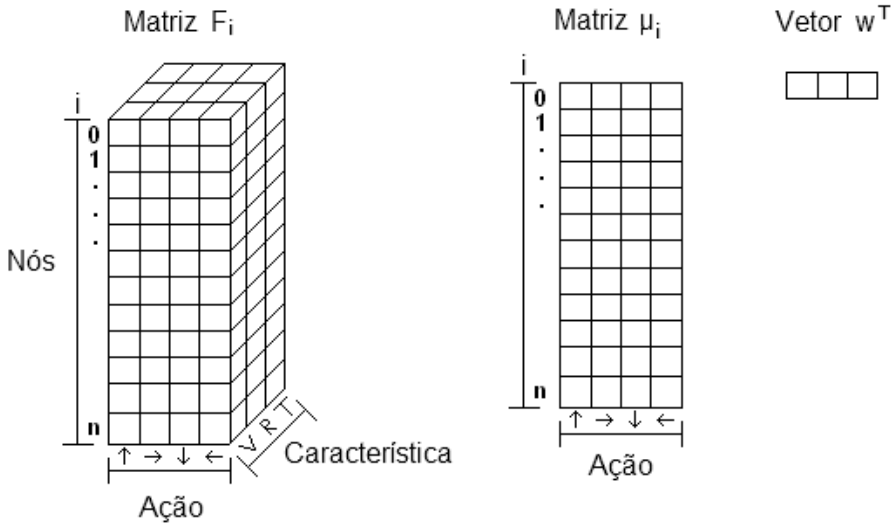

Figura 10 – Representação gráfica das matrizes do mapa, caminho e vetor de custos

produto $w^T.F_i\mu$ representa a recompensa total do caminho μ no *i-ésimo* mapa F_i computado em função dos valores apresentados pelo vetor de parâmetros w. O equacionamento deste problema foi apresentado por (RATLIFF; BAGNELL; ZINKEVICH, 2006), tendo a forma:

$$Min\ \frac{1}{2}||w||^2 \tag{5.1}$$

$$Sujeito\,a:$$

$$w^T.F_i\mu_i \geq Max_{\mu \in Y_i}\{w^T.F_i\mu + \iota_i\mu\},\ \forall i. \tag{5.2}$$

Sendo Y_i a representação de todas as escolhas possíveis de caminhos referentes ao *i-ésimo* mapa e $\iota_i\mu$ alguma função dependente da estrutura de μ_i.

Neste trabalho, utilizando as diferentes técnicas de solução relacionadas ao *Perceptron Estruturado*, tem-se um problema de minimização na restrição com um espaço de saída restrito ao conjunto Y_S. Neste caso, $w^T.F_i\mu$ representa o custo total do par (F_i, μ_i). Assim, o equacionamento toma a forma:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\,a: \\ &Min_{\mu \in Y_S}\{w^T.F_i\mu + \iota_i\mu\} - w^T.F_i\mu_i \geq \gamma.||w||_2,\ \forall i, \end{aligned} \tag{5.3}$$

onde maximizar γ e minimizar $||w||$ são equivalentes.

Em (RATLIFF; BAGNELL; ZINKEVICH, 2006) cita-se a possibilidade da utilização da distância de *Hamming* para a função $\iota_i\mu$, calculada de acordo com a diferença

entre estados do caminho do especialista e do caminho do treinamento.

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\,a: \\ &Min_{\mu \in Y_S}\{w^T.F_i\mu + (f_{tam}\mu - f_{tam}\mu_i)\} - w^T.F_i\mu_i \geq \gamma.||w||_2,\ \forall i, \end{aligned} \tag{5.4}$$

onde $f_{tam}\mu_i$ é calculado em função do tamanho do caminho. Como essa função é constante durante a minimização, a seguinte formulação é equivalente:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\,a: \\ &Min_{\mu \in Y_S}\{w^T.F_i\mu + f_{tam}\mu\} - w^T.F_i\mu_i - f_{tam}\mu_i \geq \gamma.||w||_2,\ \forall i, \end{aligned} \tag{5.5}$$

5.3 Método do Subgradiente Aplicado na Predição de Custos - *MMP*

Como já visto na seção 4.5, somente repassando para a notação matricial estabelecida, deve-se minimizar a função:

$$L(w) = \frac{1}{2}||w||^2 + C.\sum_i\{F_i\mu^* - F_i\mu_i\}. \tag{5.6}$$

Esta função possui como subgradiente:

$$g = w + C.(\sum_i\{F_i\mu^* - F_i\mu_i\}). \tag{5.7}$$

Computado em função da determinação da política ótima μ^* relativa ao caminho ótimo da matriz de custos de cada mapa de entrada obtida da equação: $w^T.F_i\mu + l_i^T$. O espaço de saída de μ^* no *MMP* é Y_i, para tanto se utiliza o algoritmo A* que apresenta complexidade quadrática em relação à quantidade de estados ou células. Assim, atualiza-se o vetor w:

$$\begin{aligned} &w_{t+1} \leftarrow w_t - \eta.g \\ &w_{t+1} \leftarrow w_t.(1-\eta) - \eta.(C.(\sum_i F_i\mu^* - F_i\mu_i)). \end{aligned} \tag{5.8}$$

Para uma taxa η de aprendizado reduzida de forma gradativa de acordo com o número de iterações e um parâmetro de regularização positivo C.

5.4 Métodos Baseados no Perceptron Aplicado na Predição de Custos

Inicialmente, não impondo à condição de maximização da margem, o problema de predição com minimização de custos é formulado como um problema de viabilidade de

um sistema de inequações na forma:

$$w^T.F_i\mu_i \leq Min_{\mu \in Y_S}\{w^T.F_i\mu\},\ \forall i, \tag{5.9}$$

ou, de modo alternativo:

$$Min_{\mu \in Y_S}\{w^T.F_i\mu\} - w^T.F_i\mu_i \geq 0,\ i = 1, ..., m. \tag{5.10}$$

Para a solução deste problema pode-se empregar a formulação do *Perceptron Estruturado*, da seção 4.1, com a seguinte regra de correção, $\forall i$:

$$Se(Min_{\mu \in Y_S}\{w^T.F_i\mu\} - w^T.F_i\mu_i) < 0, \tag{5.11}$$

ou seja, se uma inequação não está sendo satisfeita, então atualiza-se o vetor w, sendo $0 < \eta \leq 1$, na forma:

$$w_{t+1} \leftarrow w_t + \eta.(F_i\mu^* - F_i\mu_i). \tag{5.12}$$

Sendo η uma taxa de aprendizado constante e o vetor μ^* determinado através da comparação do conjunto de caminhos propostos no conjunto de treinamento S, ou seja:

$$\mu^* = argMin_{\mu \in S\ e\ \mu \neq \mu_i}\{w^T.F_i\mu\}. \tag{5.13}$$

Assim, o vetor μ^* representa sempre a melhor alternativa de caminho desconsiderando o caminho correto ou escolhido μ_i associado a cada mapa fornecido. Caso a diferença não seja positiva, certifica-se que o caminho escolhido tem um custo inferior a melhor alternativa, não sendo neste caso necessária a correção do vetor de parâmetros w.

A vantagem de se utilizar a escolha da melhor alternativa de caminho entre um conjunto de caminhos existentes e não sobre todos os caminhos possíveis está na redução do esforço computacional. Neste novo processo, a complexidade é linear e relacionada à quantidade de mapas existentes no conjunto de treinamento, lembrando-se que os custos dos caminhos em um mapa são resultados do produto interno do vetor w pelos vetores pré-computados de características dos caminhos representados por $F_i\mu$.

A equação 5.11, relacionada ao erro da amostra, pode ser ajustada da seguinte maneira: $w^T.F_i\mu_i > w^T.Fi\mu^*$. E sua interpretação é a seguinte: se o custo do caminho escolhido pelo especialista for maior do que o custo dos outros caminhos do conjunto de treinamento, então deve-se corrigir o vetor w.

Pode-se também incluir a margem na formulação do problema, conforme seção 4.3:

$$Min_{\mu \in Y_S}\{w^T.F_i\mu\} - w^T.F_i\mu_i \geq \gamma.||w||_2,\ \forall i, \tag{5.14}$$

Tem-se, agora, que o cálculo da margem de um elemento i sobre os outros é:

$$\gamma_i = \frac{w^T.(F_i\mu^* - F_i\mu_i)}{||w||_2}. \tag{5.15}$$

A condição de viabilidade do algoritmo agora é dada por:

$$w^T.\mu_i \geq \gamma.||w||_2,\ \ i = 1, ..., m. \tag{5.16}$$

Caso ocorra um erro relacionada a i-ésima amostra, ou seja:

$$w^T.(F_i\mu^* - F_i\mu_i) < \gamma.||w||_2. \tag{5.17}$$

Utiliza-se a seguinte regra de correção para o vetor de custos:

$$w_{t+1} \leftarrow w_t.\left(1 - \frac{\eta.\gamma}{||w||_2}\right) + \eta(F_i\mu^* - F_i\mu_i),\ \ 0 < \eta \leq 1. \tag{5.18}$$

É interessante observar a similaridade entre a equação 5.18, relacionada à minimização de custo, e a equação de correção do algoritmo *MMP*, maximização de recompensa, representada em 5.8.

Após a primeira execução do algoritmo *Perceptron Estruturado com Margem* há uma grande possibilidade da margem de parada não ser a máxima, então o que pode ser feito é incrementar a margem a cada execução do algoritmo, retendo os valores do vetor de custos w calculados e inicializando uma nova iteração com um valor de margem acima da última testada, no intuito de maximizá-la.

Nesta busca pela máxima margem escolheu-se dobrar seu valor a cada execução do algoritmo *Perceptron Estruturado com Margem*. Quando a solução torna-se inviável, tem-se uma margem atual que não determina corretamente os caminhos para os respectivos mapas em um determinado limite de iterações. Utiliza-se então um processo semelhante a uma busca binária como explicado na seção 4.4.

O número de vezes que o algoritmo *Perceptron Estruturado com Margem* será executado depende do grau de precisão que pretende-se obter. Repare também que o algoritmo tem sempre armazenado a última margem viável. A formulação incremental para o problema na predição de custos apresenta-se como segue:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &w^T.(F_i\mu^* - F_i\mu_i) \geq \gamma.||w||_2, \end{aligned} \tag{5.19}$$

Existe a possibilidade de se acrescentar uma função $l_i = p.f_{tam}(\mu^*) - p.f_{tam}(\mu_i), \forall i$, distância de *Hamming*, a fim de incluir uma perda específica à inequação, conforme visto

na seção 4.3, especificamente nas equações 4.29 e 4.30. Nesta formulação de predição de custos, a função f_{tam} retorna o tamanho do caminho, ou seja, a quantidade correspondente de células no caminho μ escolhido e p é um parâmetro definido a priori correspondente ao peso que o tamanho do caminho terá. Note que se o tamanho do caminho para μ_i for maior que o tamanho do caminho de μ^* tem-se um l_i negativo, ou seja, pode-se considerar que $\gamma.||w||_2$ esteja sofrendo um incremento. Dentro desse contexto tem-se:

$$w^T.(F_i\mu^* - F_i\mu_i) + l_i \geq \gamma.||w||_2. \tag{5.20}$$

Sendo que a margem γ_i para alguma amostra i pode ser escrita como:

$$\gamma_i = \frac{w^T.(F_i\mu^* - F_i\mu_i) + l_i}{||w||_2}. \tag{5.21}$$

Tem-se também que:

$$y^* = argMin_{y \in Y_S, y \neq y_i}\{w^T.f(\mu) + f_{tam}(\mu)\}. \tag{5.22}$$

Então, dentro do contexto do *Perceptron Estruturado com Margem Incremental*, assume-se a seguinte formulação:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &w^T.(F_i\mu^* - F_i\mu_i) + l_i \geq \gamma.||w||_2, \end{aligned} \tag{5.23}$$

Note que a equação 5.23 e a equação 5.3 são equivalentes para a distância de *Hamming* em l_i.

Passando da formulação primal para dual, apresentada nas seções 4.6 e 4.7, com $S = \{(F_i, \mu_i)\} = \{(F_j, \nu_j)\},\ i, j = 1, \ldots,\ m$, tem-se:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\forall \mu_i \in S : \sum_{j=1}^{m} \alpha_{\nu_j}.K((F_j\nu^* - F_j\nu_j), (F_i\mu^* - F_i\mu_i)) + l_i \geq \gamma.||w||2. \end{aligned} \tag{5.24}$$

De outro modo:

$$\begin{aligned} &Max\ \gamma \\ &Sujeito\ a: \\ &\forall \mu_i \in S : \\ &\sum_{j=1}^{m} \alpha_{\nu_j}.(J(F_i\mu_i, F_j\nu_j) - J(F_i\mu^*, F_j\nu_j) - J(F_i\mu_i, F_j\nu^*) + J(F_i\mu^*, F_j\nu^*)) + l_i \geq \gamma.||w||_2. \end{aligned} \tag{5.25}$$

Lembrando-se que, em qualquer um dos casos apresentados, a margem de separação final alcançada pelo aprendizado pode ser calculada conforme a equação 3.12: $\gamma_z = Min\{\gamma_i\}, \forall i$. Observa-se também que é necessário um $\gamma_z \geq \gamma$ a fim de satisfazer as condições de viabilidade.

5.5 Resultados Experimentais em Mapas Artificiais

Para validar os conceitos e algoritmos desenvolvidos neste trabalho optou-se pela solução de um problema prático de planejamento relacionado à determinação de caminhos em um *grid* (mapa quadriculado). O problema foi definido como um problema de minimização de custos ou de obtenção do caminho mínimo. Os mapas foram simulados com o uso de matrizes com dimensões 5×5 e 10×10. Cada componente de uma matriz define uma célula ou estado contendo um tipo de terreno. Para a realização de movimentos foram permitidas quatro ações possíveis relacionadas, respectivamente, as direções: norte, leste, sul e oeste. Também, determinou-se uma célula de origem e uma célula de destino para todos os caminhos a fim de conseguir uma melhor visualização e comparação dos resultados.

No primeiro exemplo, foram utilizados seis mapas com dimensões 5×5 para o conjunto de treinamento. Já o segundo exemplo foi realizado com quatro mapas com dimensões 10×10. Cada mapa contém as características do terreno e o melhor caminho dado pelo especialista do domínio. Para o conjunto de teste, no exemplo 1, subseção 5.5.1.1, foram utilizados doze mapas, já no exemplo 2, subseção 5.5.1.2, foram utilizados quatro mapas, ambos contendo somente as características dos terrenos.

Nos mapas pode-se ter a presença ou não de cada característica bem como a sua intensidade. Desta forma, pode-se combinar a utilização de características na definição de um tipo de terreno para cada célula. Neste primeiro experimento, para uma melhor visualização e simplificação dos resultados, optou-se somente pela presença ou não de cada característica e suas combinações possíveis, definindo assim oito tipos diferentes de terreno de acordo com a cardinalidade do conjunto. As características e suas combinações utilizadas estão representadas na Figura 11:

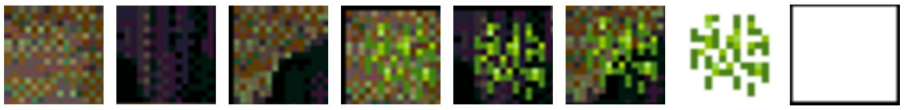

Figura 11 – No primeiro quadro tem-se somente trilha; no segundo, somente rocha; no terceiro: trilha e rocha; no quarto: trilha e vegetação; no quinto: rocha e vegetação; no sexto: trilha, rocha e vegetação (trilha abandonada); no sétimo, somente vegetação e, finalmente, no oitavo tem-se a ausência de características.

Nos vetores de custos w, a primeira componente se refere ao custo da rocha (r), a segunda ao custo da vegetação (v) e a terceira se refere ao custo da trilha (t).

5.5.1 Resultados do Conjunto de Treinamento

5.5.1.1 Exemplo 1

A Figura 12 apresenta os mapas com dimensão 5×5 utilizados no conjunto de treinamento incluindo os caminhos traçados pelo especialista. Pode-se observar nesta figura que todos os caminhos possuem a célula (1,1) da matriz como origem e a célula (5,2) da matriz como destino. Pelos resultados alcançados constatou-se que ambos os algoritmos convergiram para os mesmos caminhos definidos pela estratégia do especialista, indicando desta forma que o aprendizado foi bem sucedido e as matrizes de custos obtidas refletem a estratégia de planejamento adotada pelo mesmo.

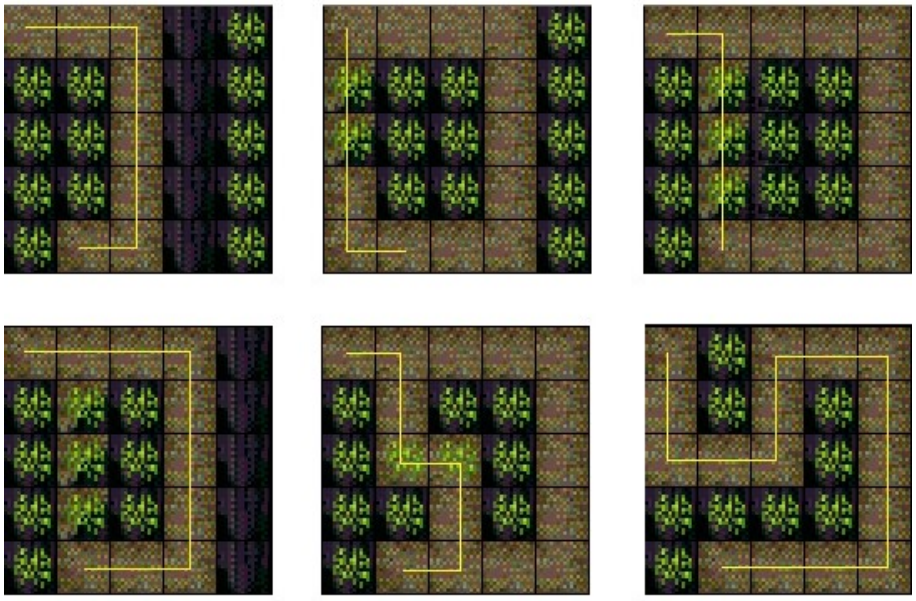

Figura 12 – Mapas de treinamento 5×5 com seus respectivos caminhos traçados pelo especialista do domínio.

Na análise da Figura 12, pode-se concluir diretamente que, segundo o especialista, geralmente é melhor escolher um caminho que segue uma trilha (t) sem vegetação (v) e sem rocha(r). Entretanto, caso o caminho requeira uma trilha (t) de maior comprimento, o especialista poderá sugerir que o caminho passe por uma trilha com vegetação (t,v), com rocha (t,r) ou com vegetação e rocha (t,v,r).

É possível também observar com mais cuidado algumas análises quantitativas em relação às escolhas. Observa-se que o especialista sempre evitou as células contendo somente (v,r), mesmo no mapa 6, onde a volta a ser dada por (t) é a maior possível. Comparando os mapas 3 e 4, verifica-se um desvio de sete células contendo (t) em vez

de passar por três células com (t,v,r). Porém, se o desvio for de nove células (t), então é melhor passar por três (t,v,r). Já no mapa 5 é preferível atravessar dois (t) e dois (t,v) do que dar a volta por oito (t). Considerando-se que essas são as únicas informações que o especialista desejou passar com a elaboração dos mapas, tem-se que novas situações deverão ser generalizadas pelo aprendizado.

Inicialmente foram comparados variações do algoritmo *MMP* com os diferentes algoritmos baseados no modelo *Perceptron*. Testou-se todos os algoritmos com taxas de aprendizado 0.2, 0.5 e 0.8. Os melhores resultados alcançados pelo *MMP* foram obtidos com a taxa de aprendizado 0.5, enquanto para os algoritmos baseados no *Perceptron*, a taxa de aprendizado 0.2 foi a mais eficiente.

Os testes aqui transcritos utilizam as melhores taxas de aprendizado obtidos em cada abordagem. Como parâmetros do algoritmo *MMP* foram utilizados: taxa de aprendizado 0.5 e parâmetro de regularização com valor 1. O vetor w foi inicializado com $[0.1, 0.1, 0.1]$. O algoritmo *MMP* necessitou de 17 iterações para convergir com tempo de execução aproximado de 0.031 segundos. Também foi estipulado uma máximo de 1000 iterações a fim de analisar o efeito sobre a maximização da margem, ou seja, como o algoritmo e sua margem se comportam tendo um alto número de iterações fixos. Seu tempo de execução é 0.402 segundos.

Como parâmetros do algoritmo *Perceptron Estruturado* foram utilizados: taxa de aprendizado 0.2. O vetor w também foi inicializado com $[0.1, 0.1, 0.1]$. O algoritmo necessitou de somente 4 iterações para convergir com tempo de execução aproximado de 0.015 segundos.

Os parâmetros do algoritmo *Perceptron Estruturado com Margem Incremental* foram os mesmos do *Perceptron Estruturado* e a separação das classes se dá de maneira idêntica, com mesmo número de iterações e tempo. Foi também estipulado inicialmente um máximo de 1000 iterações para analisar e comparar com o *MMP*. O tempo de execução foi de 0.125 segundos.

Para o algoritmo *MMP* até a separação das classes obteve-se o vetor de custos w, $w[0] = 0.819397$, $w[1] = 0.209730$, $w[2] = -5.461417$, com $||w||_2 = 5.526524$, já com as 1000 iterações obteve-se: $w[0] = 0.530317$, $w[1] = 0.450290$, $w[2] = -5.367245$, com $||w|| = 5.409307$.

Para o algoritmo *Perceptron Estruturado* obteve-se o vetor de custos: $w[0] = 1.1000$, $w[1] = 0.9000$, $w[2] = -4.700000$, com $||w||_2 = 4.910193$. O *Perceptron com Margem* em suas 1000 iterações obteve os seguintes valores para as características: $w[0] = 0.942207$, $w[1] = 0.813118$, $w[2] = -4.900445$ com $||w||_2 = 5.056014$ e margem máxima final γ: 0.171629.

A escolha do especialista reflete o melhor caminho, ou aquele de menor custo,

quando comparado a todas alternativas presentes no conjunto de treinamento. Ou seja, pode-se citar o fato de que, para cada mapa, nenhuma escolha alternativa dos caminhos apresentados possuirá menor custo que o caminho sugerido pelo especialista para o mesmo mapa. Para demonstrar esta propriedade do planejamento foram elaboradas quatro tabelas (Tabelas 1, 2, 3 e 4) que mostram os valores de custos geométricos para todos os mapas e caminhos do conjunto de treinamento para cada algoritmo. Estes valores são obtidos dividindo-se os valores dos custos dos caminhos pelas respectivas normas do vetor w. Uma medida de qualidade da solução proposta para cada treinamento pode ser considerada como a diferença entre o custo geométrico do caminho do especialista e o custo geométrico do caminho alternativo de menor custo, ou seja, a margem. Os valores referentes a estes cálculos estão destacados em negrito.

Tabela 1 – *Custos geométricos do algoritmo MMP.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**0.68218**	5.18501	4.01057	6.55951	**3.03105**	13.13026	**2.34887**
2	4.20548	**0.85970**	4.01057	**0.87708**	5.37992	11.22972	**0.01738**
3	4.20548	5.18501	**1.04591**	4.40039	**3.40348**	6.34577	**2.35756**
4	4.20548	5.18501	**1.04591**	**0.87708**	3.40348	12.02819	**0.16883**
5	1.89456	5.18501	**1.69965**	4.40039	**0.75807**	5.06101	**0.94157**
6	3.03105	**2.83614**	4.01057	5.57482	4.20548	**1.46181**	**1.37432**

Tabela 2 – *Custos geométricos do algoritmo MMP com 1000 iterações.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**0.82247**	5.27904	4.10615	6.50595	**3.16825**	13.32495	**2.34578**
2	4.34114	**0.94985**	4.10615	**1.05746**	5.51403	11.50793	**0.10760**
3	4.341142	5.27904	**1.13104**	4.57613	**3.53062**	6.63518	**2.39958**
4	4.34114	5.27904	**1.13104**	**1.05746**	3.53062	12.08367	**0.07357**
5	2.07855	5.27904	**1.84356**	4.57613	**0.98885**	5.44749	**0.85470**
6	3.16825	**2.93326**	4.10615	5.74902	4.34114	**1.76243**	**1.17082**

Tabela 3 – *Custos geométricos do algoritmo Perceptron Estruturado.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**1.85328**	6.78181	5.41730	8.28887	**4.58230**	17.24983	**2.72901**
2	5.94681	**2.13840**	5.41730	**2.38279**	7.31132	15.70202	**0.24439**
3	5.94681	6.78181	**2.54572**	6.47632	**5.39693**	9.83668	**2.85121**
4	5.94681	6.78181	**2.54572**	**2.38279**	5.39693	15.74276	**0.16292**
5	3.40108	6.78181	**2.87157**	6.47632	**2.21987**	8.43144	**0.65170**
6	4.58230	**4.05279**	5.41730	7.84083	5.94681	**3.97133**	**0.08146**

Dentre os menores valores dos diferentes algoritmos apresentados nas 4 primeiras tabelas, o algoritmo *Perceptron Estruturado com Margem* apresentou o maior, e por isso, o melhor valor de margem: 0.17162.

Tabela 4 – *Custos geométricos do algoritmo Perceptron Estruturado com Margem.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**1.52232**	6.35300	5.03659	7.73519	**4.15513**	16.10454	**2.63281**
2	5.47154	**1.78172**	5.03659	**1.95727**	6.78794	14.48772	**0.17554**
3	5.47154	6.35300	**2.12890**	5.90649	**4.84948**	8.87492	**2.72058**
4	5.47154	6.35300	**2.12890**	**1.95727**	4.84948	14.65284	**0.17162**
5	2.99955	6.35300	**2.56460**	5.90649	**1.84396**	7.53298	**0.72063**
6	4.15513	**3.72018**	5.03659	7.22289	5.47154	**3.26211**	**0.45806**

O algoritmo *MMP* e os algoritmos baseados no *Percetron* apresentam uma importante diferença. Embora o *MMP* utilize um espaço de saída Y_i, geralmente bem maior que Y_S, o mesmo não é eficiente quando se deseja maximizar a margem somente entre os próprios elementos do conjunto S. Isso ocorre devido ao fato que a maximização do *MMP* atua sobre uma margem onde se deseja obter a maior separação possível entre cada par (F_i, μ_i) e todos os outros $\mu \in Y_i$. Ou seja, o *MMP* não é indicado se o objetivo for a maximização da margem somente entre os elementos de S.

Foi feito também um estudo com diferentes taxas de aprendizado baseado no *Perceptron Estruturado com Margem Incremental* com um erro máximo $\Upsilon = 10^{-5}$ conforme visto na seção 4.4. Também houve a inserção de diferentes pesos p para funções de perda $l_i(y) = p.f_{tam}(\mu) - p.f_{tam}(\mu_i)$, onde f_{tam} é a função que retorna o tamanho do caminho escolhido e p é um peso definido a priori. O vetor w foi inicializado com $[0.001, 0.001, 0.001]$.

Observa-se que a escolha do especialista reflete o melhor caminho se comparado a todas alternativas. Isto é verificado na Tabela 5 devido a todas as margens terem valores positivos. T_S refere-se ao tempo total em segundos.

Para a maioria das margens calculadas, os resultados obtidos são muito próximos, no entanto quanto menor a taxa de aprendizado η, mais preciso tende a ser o valor da margem. Um fato interessante é que o fator p parece unicamente aumentar o valor das componentes do vetor w na mesma proporção da variação do próprio p. Comparando o vetor unitário $u = w/||w||_2$ e variando-se o peso p para as duas taxas de aprendizado $\eta = 0.1$ da tabela 5 tem-se: $u_1 = 0.2027$, $u_2 = 0.2027$ e $u_3 = -0.958$ para $p = 9$ e $u_1 = 0.2031$, $u_2 = 0.2031$ e $u_3 = -0.9535$ para $p = 6$, ou seja, a direção do vetor é praticamente a mesma e ficam ainda cada vez mais próximas conforme diminui-se a taxa de aprendizado η. Concluindo, embora o uso da função l_i de fato altere a direção do vetor w, um fator p associado a essa função não exerce influência significativa no cálculo da margem. Experimentalmente foi verificado que para valores de p entre 0.1 e 10000 o aprendizado ocorria normalmente e os valores de margem continuavam bem próximos, acima ou abaixo desses valores, os resultados começaram a ter discrepâncias, possivelmente devido a problemas numéricos associados.

Para o mesmo problema também foi utilizado o algoritmo *Perceptron Estruturado*

Dual com margem incremental e com o *Kernel* linear do produto interno. Os resultados podem ser vistos na Tabela 6 com a apresentação dos valores do vetor α. As componentes do vetor α foram inicializados com o valor 10^{-8}. Note que os valores de margem, vistos nesta tabela, são praticamente idênticos aos resultados obtidos no algoritmo primal na Tabela 5.

Os valores das componentes do vetor w podem ser obtidos a partir dos valores do vetor α e dos vetores diferenças finais d_i, conforme a equação 4.45. Observa-se que estes valores, representado na Tabela 7, são próximos dos valores do algoritmo primal, apresentados na Tabela 5,conforme esperado.

Tabela 5 – *Resultados do treinamento para o Perceptron Estruturado Primal com Margem Incremental.*

p	η	w_1	w_2	w_3	Margem	Iteração	T_S
9	10^{-1}	1.506918	1.506918	-7.123090	**0.198013**	2004697	2.2
9	10^{-2}	1.500745	1.500745	-7.124672	**0.201549**	3626112	3.5
9	10^{-3}	1.509037	1.491097	-7.124965	**0.201753**	5556315	6.2
9	10^{-4}	1.518352	1.481665	-7.124996	**0.201770**	4411427	4.4
6	10^{-1}	1.010836	1.010836	-4.745561	**0.196260**	1503764	2.1
6	10^{-2}	1.000643	1.000643	-4.749745	**0.201459**	5570351	5.6
6	10^{-3}	1.000072	1.000072	-4.749965	**0.201743**	4262428	4.6
6	10^{-4}	1.001205	0.998809	-4.749996	**0.201769**	7100893	6.9

Tabela 6 – *Resultados do treinamento para o Perceptron Estruturado Dual com Margem Incremental.*

p	η	α_0	α_1	α_2	α_3	α_4	α_5	Margem	Iteração	T_S
9	10^{-1}	0.19851	35.9814	0	20.5517	0.298805	5.62953	**0.20035**	7004368	4.2
9	10^{-2}	0.10917	36.8631	0	21.1205	0.29774	5.74868	**0.20165**	4076954	3.5
9	10^{-3}	0.10917	35.9498	0	20.5987	0.292627	5.61943	**0.20169**	5535840	3.7
9	10^{-4}	0.10878	2.20861	0	1.31829	0.292208	0.79934	**0.20169**	6123432	4.2
6	10^{-1}	0.99250	24.8124	0	14.1956	0.19938	3.86874	**0.19995**	6504636	4.3
6	10^{-2}	0.07940	24.2387	0	13.8833	0.19855	3.78479	**0.20163**	5548376	6.2
6	10^{-3}	0.07344	16.1736	0	9.27888	0.19443	2.63322	**0.20169**	7751265	5.1
6	10^{-4}	0.07255	4.54154	0	2.63263	0.19480	0.97134	**0.20169**	5291215	4.5

5.5.1.2 Exemplo 2

A Figura 13 apresenta os mapas com dimensão 10×10 utilizados no conjunto de treinamento, incluindo os caminhos traçados pelo especialista. Pode-se observar nesta figura que todos os caminhos possuem a entrada (1,1) da matriz como origem e a entrada (10,8) da matriz como destino.

Pelos resultados alcançados, constatou-se que ambos os algoritmos convergiram para os mesmos caminhos definidos pela estratégia do especialista, indicando, desta forma,

Tabela 7 – *Valores de w calculados através dos valores de α da Tabela 6.*

p	η	w_1	w_2	w_3
9	10^{-1}	1.647237	1.348432	-7.129073
9	10^{-2}	1.635118	1.365018	-7.124996
9	10^{-3}	1.646311	1.353684	-7.125003
9	10^{-4}	1.636197	1.363797	-7.125004
6	10^{-1}	1.097372	0.897991	-4.754316
6	10^{-2}	1.099266	0.900717	-4.750079
6	10^{-3}	1.097217	0.902784	-4.750001
6	10^{-4}	1.097403	0.902597	-4.750000

Tabela 8 – *Custos geométricos do algoritmo MMP.*

Cam|Mapa	1	2	3	4	Margem
1	**2.694849**	25.390744	**6.362368**	15.580350	**3.667519**
2	8.701182	**5.351525**	**7.691073**	18.078462	**2.339548**
3	**12.209404**	30.759886	**0.305382**	17.281973	**11.904022**
4	**5.725177**	14.200993	9.101996	**3.809199**	**1.915978**

que o aprendizado foi bem sucedido e as matrizes de custos obtidas refletem a estratégia de planejamento adotada pelo mesmo.

Como parâmetros do algoritmo *MMP* foram utilizados: taxa de aprendizado de 0.5 e parâmetro de regularização com valor 1. O vetor w é inicializado com $[0.1, 0.1, 0.1]$. O algoritmo *MMP* necessitou de 2 iterações para convergir, com tempo de execução aproximado de 0.015 segundos. Também foi estipulado um máximo de 1000 iterações, a fim de analisar o efeito sobre a maximização da margem com tempo de execução de 0.937 segundos.

Como parâmetros do algoritmo *Perceptron Estruturado* foram utilizados: taxa de aprendizado de 0.2 e vetor w também inicializado com $[0.1, 0.1, 0.1]$. O algoritmo necessitou de 3 iterações para convergir com tempo de execução aproximado de 0.015 segundos.

Os parâmetros do algoritmo *Perceptron Estruturado com Margem* foram os mesmos do *Perceptron Estruturado* e a separação das classes se dá de maneira idêntica, com mesmo número de iterações e tempo. Foi também estipulado um máximo de 1000 iterações para analisar a maximização da margem e o tempo de execução foi de 0.265 segundos.

Novamente, percebe-se que os dois algoritmos apresentaram resultados semelhantes (Tabelas 8, 9, 10 e 11).

Para o algoritmo *MMP* até a separação das classes obteve-se o vetor de custos w, $w[0] = 7.662500$, $w[1] = -2.712500$, $w[2] = -14.962500$, com $||w||_2 = 17.027859$, já com as 1000 iterações obteve-se: $w[0] = 2.516739$, $w[1] = 1.347019$, $w[2] = -13.359360$, com $||w||_2 = 13.659022$.

E para o algoritmo *Perceptron Estruturado* obteve-se o vetor de custos: $w[0] =$

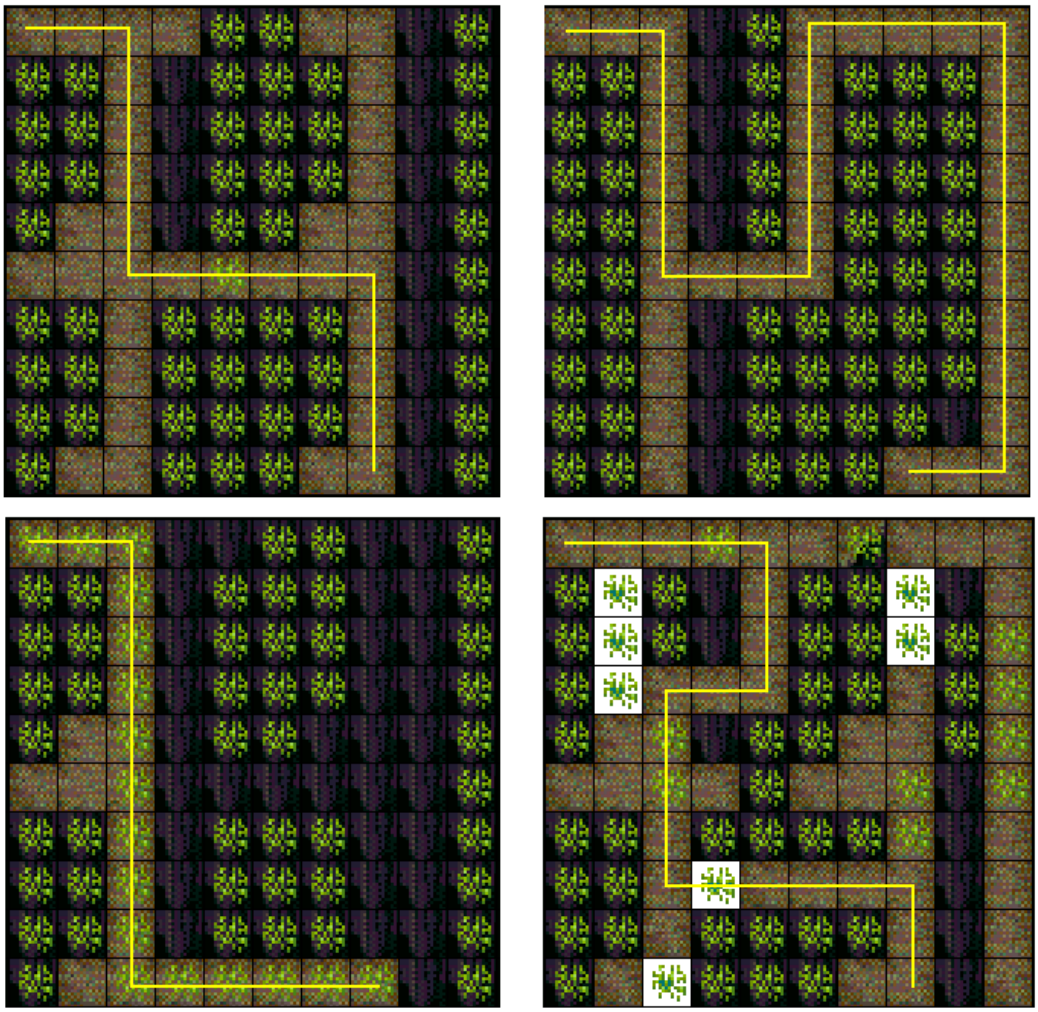

Figura 13 – Mapas de treinamento 10×10 com seus respectivos caminhos traçados pelo especialista do domínio.

Tabela 9 – *Custos geométricos do algoritmo MMP depois de 1000 iterações.*

Cam|Mapa	1	2	3	4	Margem
1	**5.536673**	31.530185	**9.220282**	19.207751	**3.683609**
2	11.741726	**10.196467**	**10.382447**	21.630638	**0.185980**
3	**15.523892**	38.129467	**7.015028**	22.123424	**8.508864**
4	**9.614510**	21.057823	13.015325	**8.169801**	**1.444709**

3.700000, $w[1] = 3.100000$, $w[2] = -13.100000$, com $||w||_2 = 13.961018$. Para o *Perceptron com margem* em suas 1000 iterações obteve-se: $w[0] = 4.365194$, $w[1] = 3.589140$, $w[2] = -16.230822$ com $||w||_2 = 17.186518$ e margem máxima final γ: 2.865498.

Dentre os menores valores dos diferentes algoritmos apresentados nas tabelas, o algoritmo *Perceptron Estruturado com Margem* apresentou o maior, e por isso, o melhor valor de margem: 2.865498.

Tabela 10 – *Custos geométricos do algoritmo Perceptron Estruturado.*

Cam|Mapa	1	2	3	4	Margem
1	**5.837683**	34.539030	**9.891829**	20.829427	**4.054146**
2	12.742624	**10.529318**	**11.095180**	23.458175	**0.565861**
3	**17.018817**	42.332158	**9.168386**	24.568410	**7.850431**
4	**10.780016**	23.529803	14.347092	**8.846060**	**1.933956**

Tabela 11 – *Custos geométricos do algoritmo Perceptron Estruturado com Margem.*

Cam|Mapa	1	2	3	4	Margem
1	**1.855873**	26.802047	**5.868689**	15.713296	**4.012815**
2	8.683122	**3.088197**	**7.067071**	18.318895	**3.978874**
3	**12.904772**	34.464635	**4.988392**	19.363068	**7.916380**
4	**6.704027**	15.852928	10.254019	**3.838529**	**2.865498**

5.5.2 Resultados do Conjunto de Teste

5.5.2.1 Exemplo 1

Na Figura 14 são apresentados os resultados dos algoritmos aplicados ao conjunto de teste e a Figura 15 se refere aos testes do algoritmo *Perceptron com Margem Incremental*, com o uso do fator de perda l_i e com $\Upsilon = 10^{-5}$.

Primeiramente, utilizou-se o algoritmo *MMP* e, em seguida, utilizou-se o algoritmo *Perceptron Estruturado*. Para a obtenção da matriz de custos associada a cada mapa multiplicou-se o vetor w de custos pela sua matriz F de características. A matriz de características possui dimensões $3 \times 25 \times 4$, representando a ocorrência das 3 características para cada um dos possíveis pares estado-ação do mapa, sendo os valores das características relacionados ao tipo de terreno para o qual a ação associada ao estado incide.

Assim, após a realização do produto $w.F$, tem-se uma matriz 25×4 representando o custo de transição relativo a cada par estado-ação a qual, finalmente, é reduzida para uma matriz 5×5 representando o custo final de cada célula ou estado. Esta redução é possível considerando o fato de que todas as transições, para uma mesma célula, possuem o mesmo custo associado, obtido diretamente do custo da célula ou do estado sucessor.

Na Figura 14, a linha contínua amarela são os caminhos achados com base no vetor de custo do algoritmo *MMP* até a separação das classes, ou seja, 17 iterações. Na maioria dos mapas, os caminhos coincidiram para todos os vetores de custos dos vários algoritmos, porém quando não ocorreu, foram traçados desvios com formas diferentes para os quatro algoritmos como segue. O caminho com quadrados laranjas, mapas 3 e 12, pertence aos desvios feitos pelo algoritmo *MMP* com suas 1000 iterações. O caminho com a linha tracejada branca representa o algoritmo *Perceptron Estruturado*, mapas 5, 10 e 12. O algoritmo *Perceptron Estruturado com Margem*, mapas 5, 9, 10 e 12, representado pelo caminho com cruzes azuis claras.

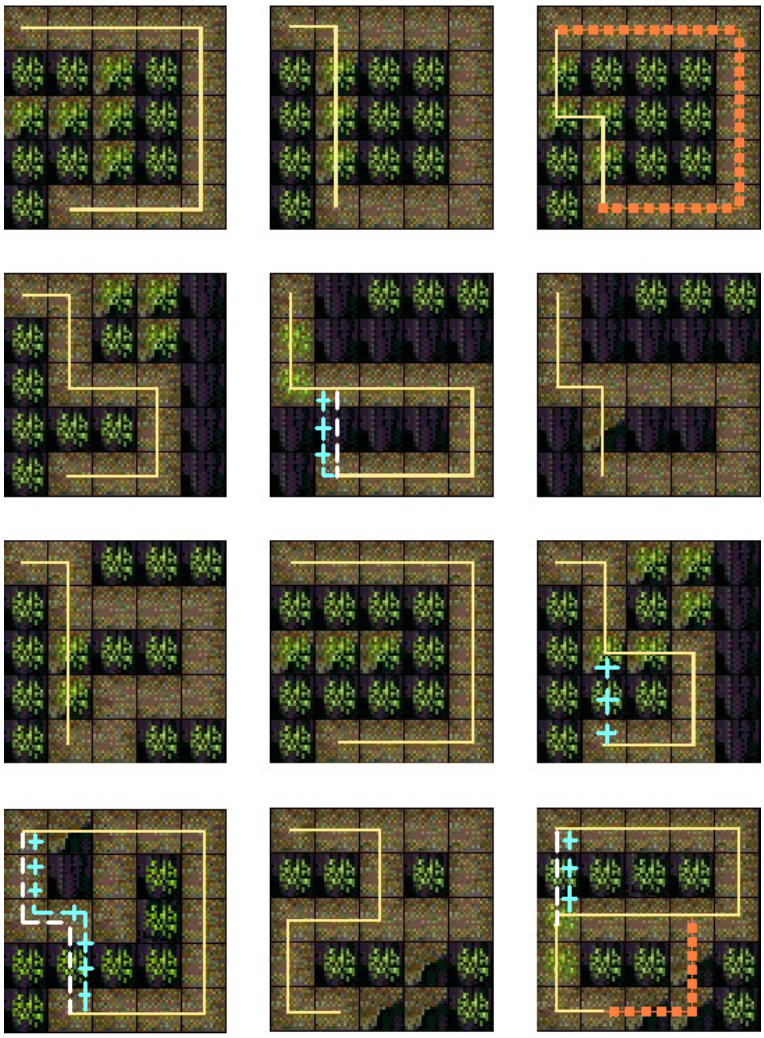

Figura 14 – Mapas de testes 5×5 com os custos já associados a cada característica e todos os caminhos traçados pelo algoritmo A* de acordo com o vetor w associado.

Na Figura 15, a linha contínua amarela representa a maioria dos caminhos calculados para as diferentes taxas de aprendizado η, tanto na formulação primal quanto na dual. A única exceção foi vista no último mapa, onde o tracejado em vermelho representa o desvio feito para a taxa de aprendizado 0.1, tanto na formulação primal quanto na dual.

5.5.2.2 Exemplo 2

Na Figura 16 foram apresentados os resultados dos dois algoritmos aplicados ao conjunto de teste.

Primeiramente, utilizou-se o algoritmo *MMP* e, em seguida, utilizou-se o algoritmo *Perceptron Estruturado*. Para a obtenção da matriz de custos associada a cada mapa multiplicou-se o vetor w de custos pela sua matriz F de características. A matriz de características possui dimensões $3 \times 100 \times 4$, representando a ocorrência das 3 características para cada um dos possíveis pares estado-ação do mapa, estando os valores das

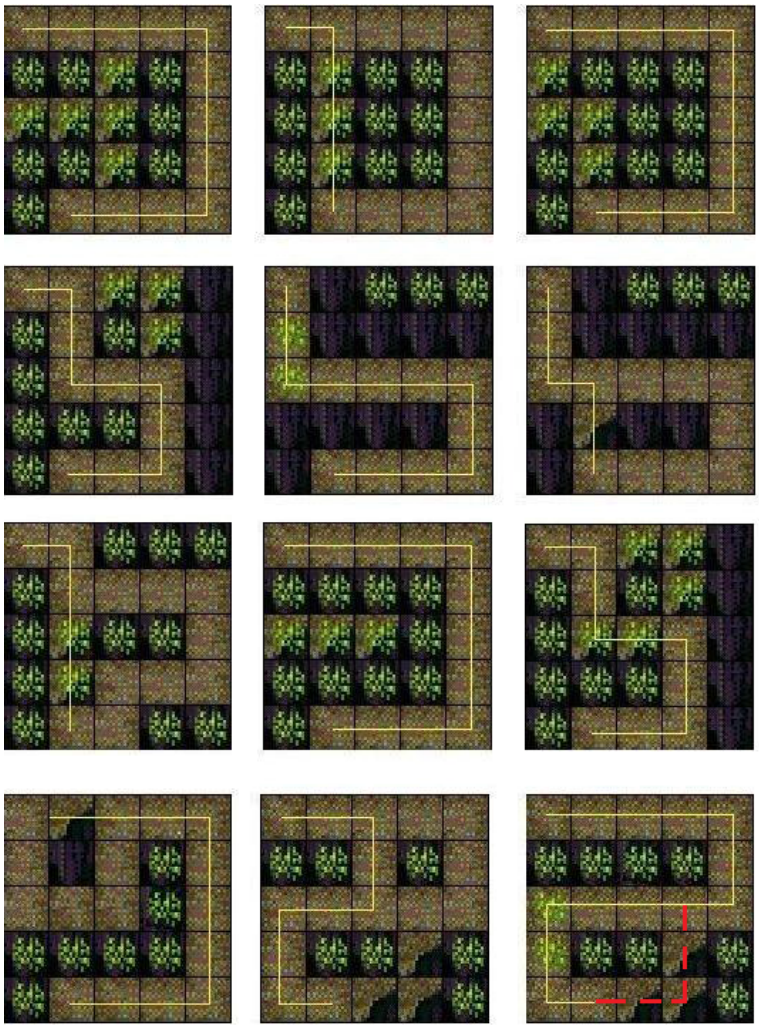

Figura 15 – Mapas de testes 5×5 e os caminhos traçados pelo algoritmo A* usando o vetor w para o primal e para o dual.

características relacionados ao tipo de terreno para o qual a ação associada ao estado incide. Assim, após a realização do produto $w.F$, tem-se uma matriz 100×4 representando o custo de transição relativo a cada par estado-ação a qual é reduzida para uma matriz 10×10 representando o custo final de cada célula ou estado. Esta redução é possível considerando o fato de que todas as transições, para uma mesma célula, possuem o mesmo custo associado, obtido diretamente do custo da célula ou do estado sucessor.

O tipo e formato dos mapas e caminhos é o mesmo do exemplo 1, com a linha contínua amarela representando os caminhos achados com base no vetor de custo do algoritmo *MMP* até a separação das classes, em 2 iterações. Todos os caminhos foram iguais no mapa 3. Já no primeiro mapa, só o *MMP* de duas iterações fez um caminho diferente, como observado na Figura 16.

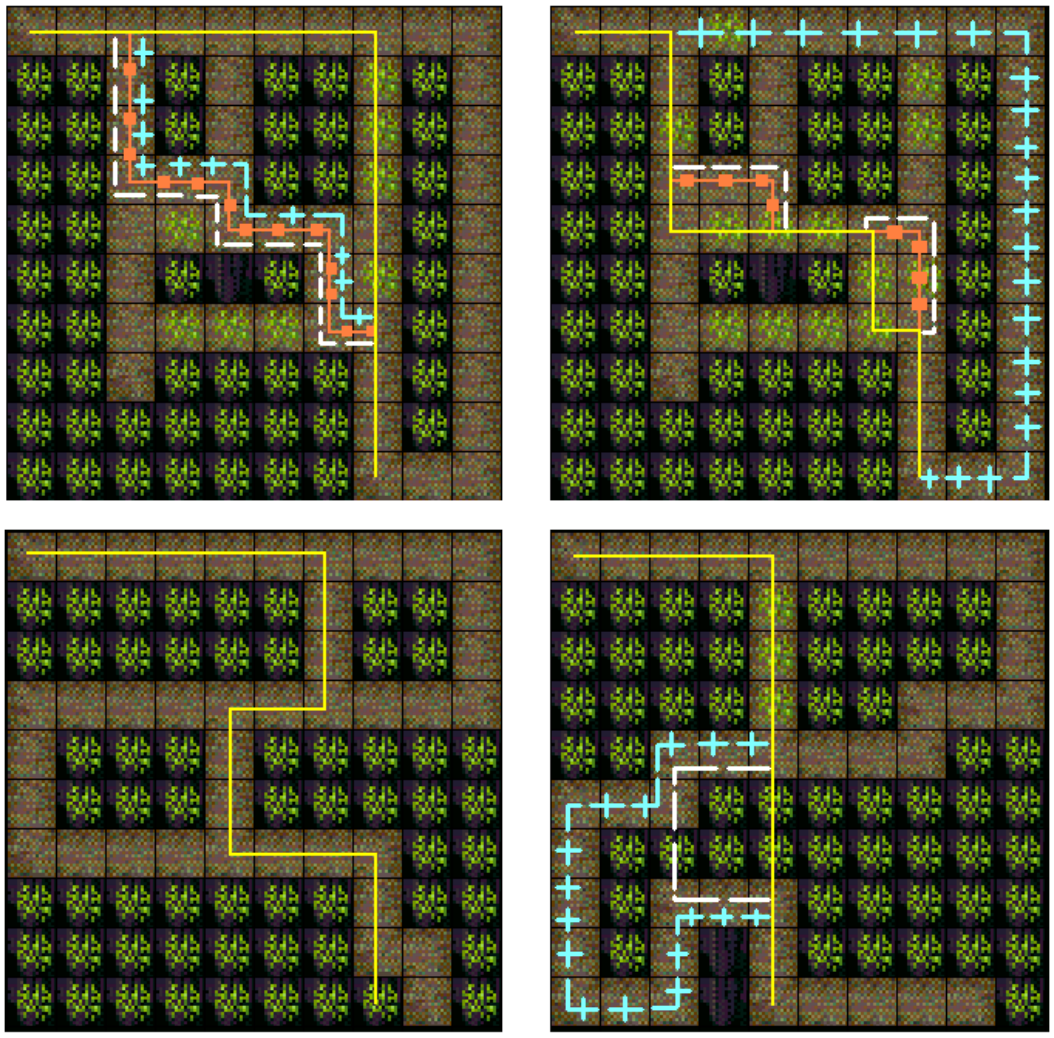

Figura 16 – Mapas de testes 10×10 com os custos já associados a cada característica e os caminhos traçados pelo algoritmo A*

5.5.3 Análise dos Resultados

Na análise dos resultados, pode-se observar que os caminhos traçados corresponderam às expectativas do especialista, onde a avaliação de dar a volta ou seguir pela trilha abandonada correspondeu à estratégia proposta pelo mesmo. Além disso, pode-se notar que os caminhos escolhidos quase sempre evitaram as células sem trilha e com vegetação e rocha.

Como o exemplo 1 é bem simples é possível observar seu funcionamento em detalhes. Analisando somente os mapas 3 e 4 da Figura 12, pode-se perceber o seguinte: não vale a pena dar a volta em 9 células somente de trilha se existe um "atalho"de 3 células com as três características. No entanto, vale a pena desviar-se de três células, com as três características, se o caminho alternativo tiver somente 7 células com trilha. Todos os algoritmos seguem essa premissa depois de treinados.

Observando o resultado nos mapas 1, 3 e 7 da Figura 14, considerando o mapa 1, está especificado explicitamente pelo especialista que dar a volta em 7 células de trilha é mais vantajoso que passar por 3 células com as 3 características. Todos seguiram o especificado, assim como seguiram a informação implícita dada pelo especialista referente ao mapa 7: se passar por 3 células com as 3 características é mais vantajoso que dar a volta em 9 trilhas. Então é melhor ainda passar por 2 células com as 3 características que dar a volta em 9 trilhas.

Pode-se considerar que a diferença dos algoritmos, seus respectivos pesos e melhores caminhos, corresponde somente ao que não puder ser deduzido pela especificação do especialista.

Em relação ao mapa 3 da Figura 14, tem-se a seguinte situação básica: deve-se passar por 4 células com as 3 características ou dar a volta em 10 células de trilha? Esta situação não pode ser deduzida pelas informações dadas pelo especialista. Da mesma maneira, se houvesse um mapa com um caminho passando por 3 características e outro passando por 8 células de trilha, sabendo que se houvesse 9 trilhas deveria-se dar a volta e se tivesse 7 trilhas não deveria dar a volta, os algoritmos provavelmente divergiriam quanto a escolha do caminho.

Percebe-se que todos os algoritmos apresentaram resultados semelhantes com relação aos custos geométricos e as margens. Porém, pode-se observar que o algoritmo *Perceptron Estruturado com Margem* tem uma ligeira supremacia em relação ao valor da margem mínima, tanto em relação ao exemplo 1 quanto ao 2.

O caminho ótimo ou caminho mínimo foi obtido com a utilização do algoritmo A*. Observando-se a Figura 16, pode-se constatar que os resultados obtidos com o algoritmo *MMP* e o *Perceptron* Estruturado foram bastante semelhantes e satisfatórios, diferenciando somente na escolha do caminho relativo ao décimo mapa. Neste caso, o caminho tracejado de branco representa o caminho sugerido pelo algoritmo *Perceptron Estruturado* e o caminho contínuo de amarelo representa o caminho sugerido pelo algoritmo *MMP*.

5.6 Resultados Experimentais em Mapas Reais

Este problema de planejamento está relacionado à determinação de caminhos num mapa quadriculado (mapas do *Google* discretizados) de grandes proporções, (COELHO; NETO; BORGES, 2012). Testou-se, deste modo, a escalabilidade do algoritmo. O esquema geral é o mesmo da seção anterior. Cada célula pode ter ou não a presença de três características bem como diferentes intensidade e suas combinações. Os valores para as intensidades das características variaram de 0 até 9, definindo 1000 tipos de células diferentes. Seis mapas exemplos foram usados tanto no conjunto de treinamento quanto no conjunto de teste.

5.6.1 Resultados do Conjunto de Treinamento

A Figura 17 mostra os mapas usados no conjunto de treinamento e o caminho em preto escolhido pelo especialista. Pode-se observar que normalmente a melhor escolha é um caminho sem rochas (incluindo aqui as construções) e árvores. Entretanto, se o caminho requere uma grande volta, o especialista pode sugerir passar através das árvores. Todos os algoritmos convergiram e refletiram a estratégia de planejamento adotada pelo especialista.

Figura 17 – Mapas de treinamento com seus respectivos caminhos escolhidos pelo especialista. *Google maps* foram discretizados com dimensão 55×55 e simplificados para abarcar os oito diferentes tipos de terreno em cada célula.

Foi definido 1000 iterações para o algoritmo *MMP*, bem como para o *Percetron Estruturado com Margem Incremental*, para efeito de comparação. Os parâmetros usados no *MMP* foram: $\eta = 0.5$ e $C = 1$. O tempo de execução foi de 15.919 segundos. Para o *Perceptron Estruturado*, com $\eta = 0.2$, a convergência ocorreu com 10 iterações e tempo de execução de 0.082 segundos. No *Perceptron estruturado com Margem*, com o mesmo η, o tempo de execução foi de 6.102 segundos.

A escolha do especialista reflete a melhor escolha quando comparado a todas as outras alternativas no conjunto de treinamento. Como demonstração, foram preparadas três tabelas com os respectivos custos geométricos dos caminhos. O valor da margem pode representar uma medida de qualidade de planejamento. A função de perda l_i é a distância de *Hamming* entre μ_i e μ.

O *MMP* obteve: $w[0] = 0.512160$, $w[1] = 0.462381$, $w[2] = -5.283517$, com $||w||_2 = 5.328383$ e margem: 0.130939. O *Perceptron Estruturado* obteve: $w[0] = 0.900000$, $w[1] =$

Tabela 12 – *Custos geométricos do algoritmo MMP.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**2.1841**	5.3702	**4.1807**	6.6904	4.4914	14.8344	**1.9965**
2	4.3685	**0.9630**	4.1089	**1.2101**	5.5523	11.3827	**0.2448**
3	4.2574	5.2741	**1.1459**	4.5357	**3.5148**	6.7055	**2.3688**
4	4.6025	5.3608	**1.3888**	**1.2579**	3.8088	12.3780	***0.1309***
5	2.1424	5.1099	**1.8885**	4.7035	**1.0097**	5.5103	**0.8788**
6	3.2751	**2.9912**	4.1957	5.8630	4.4346	**1.7917**	**1.1994**

Tabela 13 – *Custos geométricos do algoritmo Perceptron Estruturado.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**2.2287**	5.4789	**4.2717**	6.9833	4.5874	15.1739	**2.0430**
2	4.3645	**1.0957**	4.2160	**1.1700**	5.6832	11.6079	***0.0742***
3	4.1974	5.3118	**1.3186**	4.5688	**3.6031**	6.6676	**2.2844**
4	4.7546	5.3675	**1.5415**	**1.3929**	3.9931	12.8151	**0.1485**
5	2.1730	5.3118	**1.9129**	4.7917	**1.0215**	5.5718	**0.8914**
6	3.3245	**3.0645**	4.2717	5.9989	4.5317	**1.9501**	**1.1143**

Tabela 14 – *Custos geométricos do algoritmo Perceptron Estruturado com margem Incremental.*

Cam|Mapa	1	2	3	4	5	6	Margem
1	**3.3829**	6.6112	**5.2659**	8.3090	5.9157	18.3305	**1.8829**
2	5.5315	**1.9982**	5.1080	**2.2138**	6.9453	14.4523	**0.2156**
3	5.3052	6.3849	**2.3823**	5.8656	**4.9545**	9.0026	**2.5721**
4	6.1420	6.5428	**2.7665**	**2.5979**	5.5649	15.5504	***0.1685***
5	3.2250	6.1376	**2.7331**	6.2498	**2.0375**	7.7257	**0.6955**
6	4.4125	**3.9205**	5.2659	7.5951	5.7578	**3.6897**	**0.2308**

0.300000, $w[2] = -5.30000$ com $||w||_2 = 5.384236$ e margem: 0.074291. O *Percetron estruturado com Margem Incremental* obteve: $w[0] = 1.124728$, $w[1] = 0.784634$, $w[2] = -4.777393$ com $||w||_2 = 4.970327$ e margem final: 0.168532. O valor final da margem foi definido como a diferença mínima entre o custo geométrico do caminho do especialista e o caminho alternativo de melhor custo. Este valores estão em negrito nas respectivas tabelas 12, 13 e 14.

5.6.2 Resultados do Conjunto de Teste

A Figura 18 apresenta os resultados dos algoritmos aplicados no conjunto de teste.

A linha preta representa o caminho baseado no vetor de custos do algoritmo *MMP*. Na maioria dos casos os caminhos convergiram para o mesmo. Contudo, quando eles não coincidiam, outros caminhos, com diferentes formas, foram estabelecidos para a identificação dos outros dois algoritmos. O caminho vermelho tracejado representa o *Perceptron*

Figura 18 – Mapas de teste com os caminhos definidos pelo algoritmo A* com base no vetor de custos w.

Estruturado, mapas 3 e 6. O caminho amarelo com quadrados representa o *Perceptron Estruturado com Margem Incremental*, mapa 6. Todos eles mostraram que existe uma forte associação entre as saídas representadas pelos custos e a estratégia do especialista. Isto confirma que esta abordagem é uma eficiente alternativa quando comparada ao *MMP*.

5.6.3 Análise dos Resultados

Os algoritmos avaliados tem comportamento similares, mas os algoritmos estruturados baseados no *Perceptron* (COELHO; NETO; BORGES, 2012) requerem um menor gasto de memória e processamento, uma das causa é não rodar o algoritmo A* durante o processo de aprendizado. Os custos são efetivamente preditos para um pequeno número de mapas, permitindo aplicações ainda maiores e variadas. É importante notar que o algoritmo baseado no *Perceptron* obteve novamente um valor de margem melhor que a do *MMP*, demonstrando ser eficiente e bastante promissor.

5.7 Conclusão dos Experimentos para Problemas Linearmente Separáveis

A utilização de técnicas de aprendizado no problema de determinação do caminho mínimo entre dois pontos, aliado ao problema de predição de custos, permitiu que o problema fosse solucionado sem qualquer tipo de conhecimento prévio dos custos das características dos mapas. Os algoritmos se mostraram eficientes e de baixa demanda

computacional. Vê-se também que, de modo geral, quanto maior a margem, mais próximo das expectativas do especialista.

5.8 Resultados Experimentais em Mapas Artificiais Não-Linearmente Separáveis

Neste experimento, é usado o modelo *Perceptron Estruturado Dual com Kernel*, seção 4.7, também chamada de representação dependente dos dados, conforme equação 5.25, seção 5.4.

5.8.1 Resultados do Conjunto de Treinamento

A Figura 20 mostra os mapas usados no conjunto de treinamento para um problema não-linear, incluindo os caminhos traçados por um especialista. Para isso, foram inseridas três intensidades para cada característica, de acordo com a Figura 19. Desse modo, para cada célula, a soma de todas as intensidades deve ser igual a três. O zero representa a ausência de determinada característica.

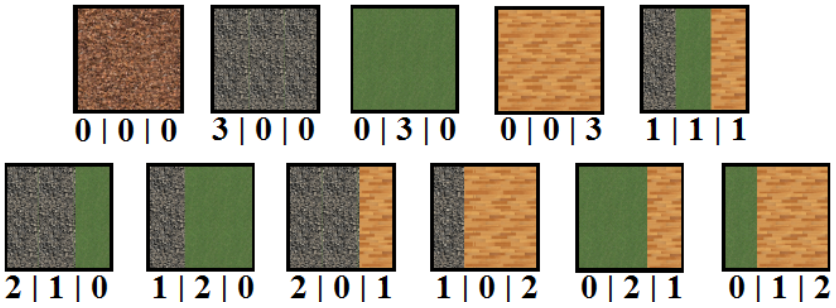

Figura 19 – Diferentes intensidades entre as três características. Os valores de intensidades são, nesta ordem: (Rocha | Vegetação | Caminho).

Como mostrado na Figura 20, todos os caminhos possuem a célula (1,1) do mapa como origem e a célula (4,2) do mapa como destino. Analisando a estratégia do especialista, pode-se concluir que o melhor é escolher um caminho que atravessa as células com duas ou três características distintas, não importando quais são os valores de intensidade. Neste caso, um algoritmo de separação linear não terá sucesso para os mapas apresentados na Figura 20. Isso foi comprovado empiricamente nessa experiência, pois no *Perceptron Estruturado* não houve convergência para uma margem positiva, mesmo com um grande número, na casa dos milhões, de iterações. Para resolver este problema só mesmo um algoritmo dual com *Kernel* não-linear. Apesar de ainda não ter sido desenvolvido um formalismo matemático para definir se um problema estruturado é ou não linearmente separável, o raciocínio intuitivo por trás deste experimento é o seguinte: a estratégia escolhida força uma relação não-linear entre duas características diferentes, tornando sua

Tabela 15 – *Valores de margens usando um Kernel quadrático.*

p	η	α_0	α_1	α_2	α_3	Margem
9	10^{-3}	0.0060	1.7910	1.4505	1.3414	**0.4624**
9	10^{-4}	0.0054	1.7882	1.4498	1.3407	**0.4627**
6	10^{-3}	0.0030	0.7958	0.6443	0.5967	**0.4620**
6	10^{-4}	0.0024	0.7946	0.6443	0.5959	**0.4626**

Tabela 16 – *Valores de margens usando um Kernel cúbico.*

p	η	α_0	α_1	α_2	α_3	Margem
9	10^{-3}	0.4210	1.1108	1.0614	0.7063	**2.9873**
9	10^{-4}	0.4229	1.1116	1.0610	0.7067	**3.0060**
6	10^{-3}	0.1198	0.3239	0.3147	0.2093	**2.7998**
6	10^{-4}	0.1253	0.3295	0.3144	0.2094	**3.0047**

resolução somente possível com um *kernel* polinomial quadrático ou de ordem superior. O *kernel* produto interno não consegue estabelecer esta relação.

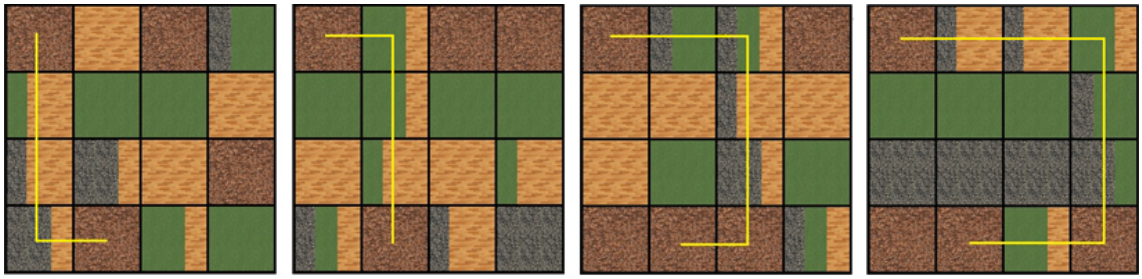

Figura 20 – Mapas com seus respectivos caminhos escolhidos por um especialista.

Todos os componentes dos vetor α foram inicializados com o valor 10^{-8}. Primeiramente, uma função polinomial quadrática foi usada como *Kernel*, ou seja, $J(\mu_i, \nu_j) = \langle \mu_i, \nu_j \rangle^2$ foi inserida na equação 5.25. A Tabela 15 mostra os valores de margem obtidos para diferentes taxas de aprendizagem η e diferentes valores do parâmetro p na função de perda.

A Tabela 16 apresenta os valores de margem obtidos para uma função de *Kernel* polinômio cúbico, $J(\mu_i, \nu_j) = \langle \mu_i, \nu_j \rangle^3$ utilizado na equação 5.25.

5.8.2 Resultados do conjunto de teste

Não é possível reconstruir e usar o vetor w no espaço direto. No entanto, pode-se avaliar o custo dos caminhos usando o truque *Kernel*. Seguindo o mesmo raciocínio da equação 4.63, percebe-se que é possível calcular o custo total de um caminho μy qualquer

sobre um mapa F_i de acordo com a equação 5.26:

$$\begin{aligned} &\sum_{j=1}^{m} \alpha_{\mu\nu j}.(\theta(F_j\nu^*) - \theta(F_j\nu_j)).\theta(F_i\mu y) \\ =&\sum_{j=1}^{m} \alpha_{\mu\nu j}.k(\langle F_j\nu_j, F_i\mu y\rangle) - k(\langle F_i\nu^*, F_i\mu y)\rangle) \\ =&\sum_{j=1}^{m} \alpha_{\mu\nu j}.J(F_j\nu_j, F_i y) - J(F_j\nu^*, F_i\mu y) \end{aligned} \tag{5.26}$$

Assim, pode-se realizar uma busca pelo melhor caminho através da comparação dos custos dos vários caminhos possíveis. Além disso, a análise do vetor α torna possível reconhecer os exemplos que têm maior influência na definição da estratégia. A Figura 21 apresenta os caminhos selecionados para o conjunto de teste e a Figura 22 apresenta em quais mapas esses caminhos foram analisados.

Como mostrado na Figura 22, os resultados satisfazem as expectativas do especialista. Os caminhos evitam células com apenas uma característica, escolhendo células com duas ou três características. Em todos os mapas, os caminhos obtidos com os *Kernels* quadrático e cúbico foram os mesmos, exceto para o último mapa, que apresentou uma pequena diferença. O caminho pontilhado branco representa esse desvio para o *Kernel* cúbico.

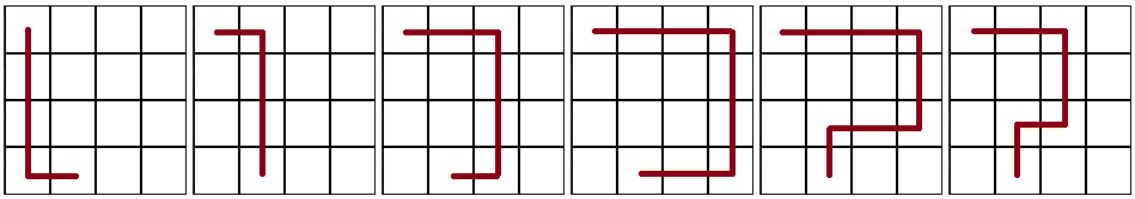

Figura 21 – Conjunto de caminhos possíveis μ cujo custo em F_i foi calculado de acordo com a equação 5.26.

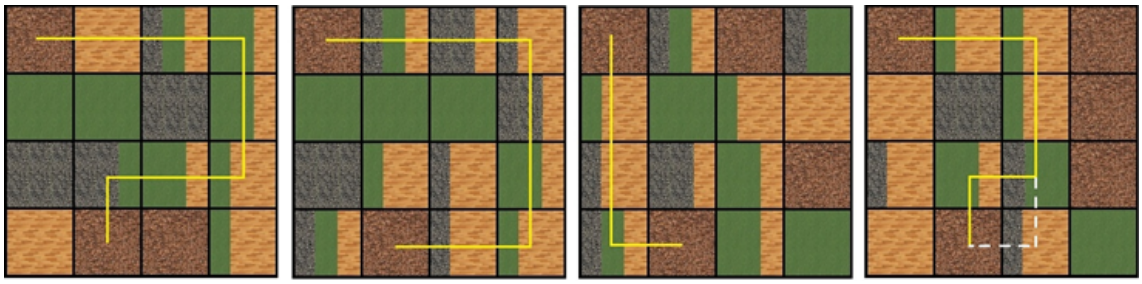

Figura 22 – Mapas de testes com seus respectivos melhores caminhos escolhidos entre os apresentados na Figura 21.

5.8.3 Análise dos Resultados

Utilizou-se um algoritmo de margem incremental para a predição estruturada não-linear. O estado da arte indica ser difícil a utilização de funções do *Kernel* para o tra-

tamento da não-linearidade no problema de previsão estruturado, pois não é possível reconstruir o vector w e utilizar no espaço direto. No entanto, é possível avaliar os valores do vetor de características utilizando o truque *Kernel*, que representa o custo de caminhos. Portanto, pode-se, assim, aprender estratégias não-lineares do especialista e avaliar os valores dos caminhos em novos mapas.

Os caminhos são obtidos por um processo de busca através da escolha do melhor caminho que representa a estratégia do especialista, mesmo não se tendo o custo de cada característica individual e, por conseguinte, a matriz de transição de custos. Além disso, a análise do vetor α, torna possível reconhecer os exemplos que têm a maior influência na definição de uma estratégia. O uso de um algoritmo gerador de caminho ótimo, tal como o A*, para resolver o problema de otimização em casos não-lineares não seria possível, porque requer o conhecimento das componentes do vector w no espaço direto.

O tratamento da não-linearidade em problemas de previsão estruturados nem sempre é fácil de resolver. Pode depender da estrutura do problema inverso e da solução do problema de optimização associado. No entanto, os resultados obtidos, em particular para o problema dos caminhos de planejamento, foram encorajadores, permitindo uma solução mais abrangente no tratamento de não-linearidade usando amostragem e funções *Kernel*.

6 Estratégia On-line para Predição de Dados Estruturados em Grafos de *Markov*

Redes Complexas tem sido alvo de importantes estudos (ALBERT; BARABáSI, 2002) (NEWMAN, 2001), possuindo um conjunto de diferentes aplicações: redes sociais (JIN; GIRVAN; NEWMAN, 2001), biológicas (GIRVAN; NEWMAN, 2002), de comunicação (SILVA et al., 2008), tecnológicas (Ramon Ferrer i Cancho; JANSSEN; SOLÉ, 2001) e de transportes (XU; HARRISS, 2008); retratando como exemplo o relacionamento entre pessoas (JONES; HANDCOCK, 2003), proteínas (EVLAMPIEV; ISAMBERT, 2007), redes *p2p* (GARCIA; SILVA; MEO, 2010), computadores (LLOYD; MAY, 2001) e aeroportos (LLOYD; MAY, 2001), respectivamente. As relações, entretanto, dependem da característica que se quer estudar e refletem propriedades intrínsecas dos elementos considerados. Por exemplo, pessoas podem estar ligadas por conexões de amizade ou devido ao compartilhamento de alguma opinião, enquanto aeroportos estarão ligados se possuem rotas que os conectam. Diferentes modelos matemáticos já foram propostos no intuito de produzir tais redes artificialmente, tais como: grafos aleatórios (SOLOMONOFF; RAPOPORT, 1951) (MOLLOY; REED, 1998), livres de escala (BARABASI; ALBERT, 1999) (BARABáSI; ALBERT; JEONG, 1999), mundo pequeno (WATTS, 1999) (WATTS; STROGATZ, 1998) e grafos de *Markov* (FIENBERG; WASSERMAN, 1981) (FRANK; STRAUSS, 1986). Todavia, em função do modelo selecionado, somente um certo conjunto de propriedades que retratam características reais de suas respectivas redes podem ser simuladas. Além disso, tais modelos necessitam da definição de parâmetros para adequar o modelo teórico as características que se deseja obter das redes reais. Neste sentido, torna-se importante o desenvolvimento de algoritmos de aprendizado que sejam capazes de realizarem a predição destes parâmetros. O problema de aprendizado em questão pode ser definido como um problema de predição estruturada, relacionado a teoria de problemas inversos, sendo formulado como uma simplificação do problema de predição estruturada utilizando o modelo *Perceptron* (COELHO; NETO; BORGES, 2012). Diversos testes relacionados à predição destes parâmetros em um modelo teórico foram realizados.

6.1 Introdução

O objetivo nesta aplicação é possibilitar a predição de parâmetros probabilísticos em problemas de redes complexas, em especial, as que se constroem utilizando o modelo de grafos de *Markov* (FRANK; STRAUSS, 1986). Este tipo de aprendizado tem como base a predição de um conjunto de parâmetros associados a um mapeamento funcional entre domínios estruturados e arbitrários de entrada e saída. Neste problema, tem-se como

domínios um ou mais objetos estruturados na forma de grafos, que modelam a evolução de redes complexas.

O algoritmo usado em questão é o *Perceptron Estruturado com Margem Zero*, por possibilitar o aprendizado mesmo para pares de entradas únicos.

É possível aqui também realizar uma correlação com a estrutura de problemas inversos através da formulação: $d = F(G(m))$, conforme visto em 0.1. Onde d é a característica (grau) desejada alcançar, F representa o modelo (*Markov*) a ser aplicada no grafo, G é o grafo em questão e m é o vetor de parâmetros do modelo utilizado. Porém, não tem-se a necessidade de que um único grau d e um grafo G sejam correlacionados por um vetor *vet*, tem-se na verdade a opção de que *vet* seja condizente para mais de um grafo G a fim de obter um mesmo d.

Este aprendizado torna possível a obtenção dos parâmetros de criação a partir da percepção das características das próprias redes, podendo ser de grande valia a sua utilização (COELHO et al., 2013). Frequentemente ocorre uma clara separação entre o processo de obtenção destes parâmetros e a geração de novas redes. Neste sentido, a geração somente se torna possível a partir do prévio conhecimento dos respectivos parâmetros, muitas vezes obtidos de forma imprecisa através unicamente da experiência do especialista em redes complexas, analisando testes e comparando com redes anteriores. Desta maneira, propõe-se um método para estimar esses parâmetros de forma mais precisa.

Com o intuito de comprovar a eficiência desta nova abordagem, bem como a corretude do algoritmo proposto, foram realizados diversos testes, com diferentes dados de entrada e refletindo a escolha de diferentes grafos. Os resultados obtidos corresponderam ao esperado, demonstrando sempre uma forte associação entre a saída obtida e a saída esperada.

6.2 Formulação Teórica

O algoritmo *Perceptron Estruturado com Margem Zero*, seção 4.2, será aplicado ao problema de geração de redes complexas dinâmicas, que preservam ou que evoluem para uma determinada característica desejada, obviamente, ligada ao contexto do sistema em estudo. A rede analisada é representada por um grafo $G = (V, E)$, sendo V o conjunto de nós e E o conjunto de arestas. Esta rede evolui, e, portanto, diferentes grafos orientados indexados no tempo são utilizados para representar a sua dinâmica, i.e, $G_t = (G_1, G_2, G_3, \ldots, G_n)$. Cada grafo $G_t = (V_t, E_t)$ representa uma instância da rede em um instante de tempo. Sem perda de generalidade, considera-se que o intervalo entre dois grafos gerados é igual a uma unidade de tempo. O objetivo é definir parâmetros para um dado modelo de dinâmica, de tal forma que esta família de grafos preserve ou possua determinada característica topológica (NEWMAN, 2001).

Neste trabalho, o modelo de dinâmica a ser parametrizado pelo algoritmo proposto é o modelo de *Markov* (FRANK; STRAUSS, 1986). Este modelo é simples e governa a existência de arestas ao longo do tempo. No processo estocástico *markoviano* modelado, dois estados são representados: no estado E, a aresta existe; no estado $!E$, a aresta não existe. De acordo com este modelo, uma aresta qualquer pode existir ou não em um dado intervalo de tempo. No tempo t, se a aresta existe, esta deixa de existir no tempo $t+1$, com probabilidade igual a $1-p$, e se mantém com probabilidade p. Da mesma maneira, se a aresta não existe no tempo t, passa a existir no tempo $t+1$, com probabilidade $1-q$ e permanece sem existir com probabilidade q. Neste sentido, no instante inicial $t=0$ existe um grafo G_0, que evolui ao longo do tempo seguindo o modelo descrito anteriormente.

Diversas características topológicas de uma rede podem ser modificadas. Neste estudo, considera-se o grau médio dos grafos, d_m, e consequentemente, o grau total, d_t. Esta característica é interessante, por exemplo, se o sistema real em análise somente suporta um número limitado de ligações entre os nós dos grafos.

Desta forma, a seguinte formulação pode ser desenvolvida: o objeto estruturado M_{x_i} é o grafo inicial G_0, o objeto estruturado M_{y_i} é o grafo final G_f. O vetor w é composto por p e q, que aplicados a M_{x_i} resultarão no grafo M_{y_i}, segundo a característica topológica desejada.

O objetivo consiste em aprender os parâmetros p e q, de modo que aplicados em M_{x_i} o leva a obter o mesmo grau médio do objeto M_{y_i}. Quando ocorre essa correspondência exata entre os mesmos, dá-se o nome de acerto. Enquanto não for obtido um número mínimo de acertos a ser definido, é preciso haver a correção de w.

Considerando a adequada representação dos objetos estruturados, a seguinte notação pode ser elaborada, a partir dos exemplos das matrizes M de adjacência que representam o conjunto de grafos $T = \{A, B, C, D, E, F\}$:

$$w = [p, q], \tag{6.1}$$

$$M_{x_i},\ i = A, B, C, D, E, F,$$

$$M_{y_i},\ i = A, B, C, D, E, F. \tag{6.2}$$

Seja φ uma função que retorna um vetor de duas posições: a quantidade de ligações existentes e a quantidade de ligações complementares. Por exemplo: para uma matriz de adjacência com 8 nós e 50 arestas, a função retornaria [50,14], pois o grafo completo possuíra no máximo 64 ligações. Aplicando esta função nas matrizes correspondentes, tem-se a seguinte definição para o mapeamento funcional entre os objetos apresentados:

$$f(x_i, y_i) = \varphi(M_{x_i}) - \varphi(M_{y_i}). \tag{6.3}$$

Considerando as matrizes que representam dois grafos A e B têm-se:

$$f(x_i, y_i) = \varphi(M_{x_A}) - \varphi(M_{y_B}). \tag{6.4}$$

Em função destas notações, a regra de correção apresentada na equação 4.20 pode então ser reescrito como segue. Se um erro ocorre, ou seja, se não existe a correspondência entre a saída real e a desejada: $(\varphi(M_{x_A}) - \varphi(M_{y_B})) - (\varphi(M_{x_A}) - \varphi(M_{y^*})) \neq 0$, então corrigi-se o vetor w:

$$[p, q] = [p, q] - \eta.(\varphi(M_{x_A}) - \varphi(M_{y_B}) - (\varphi(M_{x_A}) - \varphi(M_{y^*}))), \tag{6.5}$$

considerando o vetor $[p, q] > 0$.

Podendo ainda ser simplificado através de operações algébricas como segue. Se não existe a correspondência exata entre a saída desejada e a saída real, ou seja, se $\varphi(M_{y^*}) - \varphi(M_{y_B}) \neq 0$, então é necessária a correção:

$$[p, q] = [p, q] - \eta.(-\varphi(M_{y_B}) + \varphi(M_{y^*})),\ 0 < \eta \leq 1, \tag{6.6}$$

onde M_{y^*} representa a matriz de adjacência calculada pelo processo de aprendizado através da utilização do vetor w atual. Neste exemplo M_{y^*} é calculado aplicando-se o vetor $w = [p, q]$ em M_{x_A} através do processo de formação do grafo *markoviano*, ou seja, se uma aresta existe em M_{x_A}, esta deixa de existir com probabilidade igual a $1 - p$, e se mantém com probabilidade p. Da mesma maneira, se a aresta não existe, passa a existir com probabilidade $1 - q$ e permanece sem existir com probabilidade q. Para simplificar, esse processo de formação será representado como: $M_{y^*} = [p, q].M_{x_A}$.

6.3 Simulação do Grafo de *Markov*

No intuito de validar o algoritmo de treinamento proposto, formulou-se um problema de geração de uma sequência de grafos *markovianos*. Foram considerados grafos com $|V| = 8$ nós, possibilitando um total máximo de $|E| = 64$ arestas. Os parâmetros do *Perceptron Estruturado com Margem Zero* são descritos a seguir. A taxa de aprendizado n é igual a 0.2/64, sendo normalizado na mesma ordem de magnitude do número de nós para manter a probabilidade dos valores de p e q entre 0 e 1. O vetor w foi inicializado com [0.5,0.5], ou seja, a probabilidade inicial de uma aresta se manter ou ser retirada é a mesma.

Nesse contexto, pretende-se modificar M_{x_i} de modo a obter o mesmo grau médio que M_{y_i}, aprendendo os parâmetros p e q necessários para que isto ocorra.

Resumindo, p e q são atualizados conforme equação 6.6 sempre que $M_{y^*} = [p, q].M_{x_A}$ não corresponder a saída desejada M_{y_i}. Quando $M_{y^*} = M_{y_i}$, significa que obteve-se um

acerto e que o vetor w não precisará ser corrigido. Porém, como será exemplificado no experimento da seção 6.4.1, uma primeira ocorrência de acerto não significa necessariamente que os valores computados para p e q sejam precisos. Isto ocorre, devido ao próprio processo probabilístico de geração do grafo, onde para um $[p, q]$ ruim, a ocorrência de uma acerto é possível, mesmo com baixa probabilidade. Deste modo, é vantajoso continuar gerando mais grafos M_{y^*} e continuar corrigindo w, como será descrito a seguir.

Considere o modelo de *Markov* parametrizado com os valores de p e q. Seja uma família de grafos $G_t = (G_1, G_2, G_3, \ldots, G_n)$ geradas a partir de G_0. Uma questão a ser respondida é: quantos grafos α em média devem ser gerados através do processo $[p, q].M_{x_i}$, de modo que um total de β grafos possuam o valor de grau médio definido em M_{y_i}? Ou seja, em média quantas iterações α são necessárias para que o grau médio equivalente em G_t seja obtido em pelo menos β grafos. Em outras palavras, a estabilidade dos grafos gerados pode aqui ser definida como a fração $\vartheta = \alpha/\beta$, sendo a métrica β definida como o total de acertos, ou correspondências exatas de grau e a métrica α definida como o total de grafos gerados, necessários para que β atinja o valor previamente estipulado.

Nos cenários analisados, foi considerado $\beta = 50$, enquadrando-se estatisticamente numa distribuição normal de probabilidade. Segundo (BOLFARINE; BUSSAB, 2005), os modelos probabilísticos são conhecidos a partir dos dois seguintes resultados.

Primeiro, se $(X1, X2, \ldots\ldots, Xn)$ é uma amostra aleatória de uma população com distribuição normal de média $m\mu$ e desvio padrão $m\sigma$, então a média da amostra (X) terá uma distribuição também normal com a mesma média da população e com desvio padrão "$\sqrt{n}$ vezes menor"que o desvio padrão da população. Isto é: Se X é $N(m\mu, m\sigma)$ então $\bar{X}$ será $N(m\mu, \frac{m\sigma}{\sqrt{n}})$, onde $N()$ refere-se a uma distribuição normal e n, ao tamanho da amostra.

Segundo, através do teorema central do limite, se $(X1, X2, \ldots, Xn)$ é uma amostra aleatória extraída de uma população com qualquer distribuição de média $m\mu$ e desvio padrão $m\sigma$, então a média da amostra $(\bar{X})$ terá uma distribuição aproximadamente normal com a mesma média da população e com desvio padrão "$\sqrt{n}$ vezes menor"que o desvio padrão da população à medida que o tamanho da amostra aumenta. Para amostras de 30 ou mais valores, em geral, a aproximação já será suficiente boa, para se poder utilizar este resultado.

Resumindo, se X tem qualquer distribuição, então $\bar{X}$ terá uma distribuição aproximadamente $N(m\mu, \frac{m\sigma}{\sqrt{n}})$ para $n \geq 30$.

6.4 Experimentos e Resultados

Para melhor caracterização do processo de aprendizado foram considerados seis diferentes cenários de aplicação, como visto a seguir.

6.4.1 Aprendizado de um Grafo Menos Denso para um Grafo Mais Denso

Considere a matriz de adjacência M_{x_B}, com graus médio e total, respectivamente, $d_m = 1.25$ e $d_t = 10$, representando o grafo $G_0 = B$, com 8 nós. Seja também a matriz de adjacência M_{y_A} do grafo final $G_f = A$, com $d_m = 6.25$ e $d_t = 50$, conforme observado na Figura 23. Aplicando a equação 6.6, foram necessárias 6 iterações para que o grafo B igualasse o grau médio d_m do grafo A, onde cada iteração corresponde a uma geração de grafo $M_{y^*} = [p, q].M_{x_B}$ e a consequente correção de w quando $M_{y_A} \neq M_{y^*}$. Os valores dos parâmetros ao final desta etapa foram $p = 0.796$ e $q = 0.204$.

$$A = \begin{matrix} 01111100 \\ 10111110 \\ 11011111 \\ 11101111 \\ 11110111 \\ 11111011 \\ 01111101 \\ 00111110 \end{matrix} \qquad B = \begin{matrix} 01000000 \\ 10000000 \\ 00000011 \\ 00000011 \\ 00000000 \\ 00000000 \\ 00110000 \\ 00110000 \end{matrix}$$

Figura 23 – Matrizes de adjacência M_{x_B} e M_{y_A} correspondentes aos grafos B e A.

Utilizando os parâmetros $p = 0.796$ e $q = 0.204$ encontrados após esse primeiro acerto, ou seja, após a primeira equivalência de grau entre M_{y_A} e M_{y^*}, foram gerados a partir da aplicação do modelo *Markoviano* em M_{x_B}, ou seja, $[0.796, 0.204].M_{x_B}$, uma sequência de $\alpha = 233$ grafos, a fim de se obter $\beta = 50$ equivalências exatas de grau entre $[0.796, 0.204].M_{x_B}$ e M_{y_A}, sem qualquer correção adicional do vetor w no processo. Assim, em média $\vartheta = 4.66$ grafos devem ser gerados para que se obtenha o grau médio de B, segundo os valores anteriormente calculados para p e q. Como pode ser observado, $\vartheta = 4.66$ é um valor elevado, lembrando que quanto menor o valor de ϑ, melhor. Deste modo, essa abordagem de utilizar os parâmetros p e q logo no primeiro acerto não é a estratégia mais adequada. Nesta tese, valores considerados ótimos encontram-se entre $1 \leq \vartheta \leq 2$ e valores aceitáveis entre $2 < \vartheta \leq 4$.

Deste modo, uma nova abordagem foi então considerada. Testou-se também a correção contínua do vetor w, ou seja, mesmo após o primeiro acerto, a correção continuou ocorrendo cada vez que uma diferença entre M_{y_A} e M_{y^*} acontecia durante a geração de grafos. Repetiu-se a geração e a correção até a obtenção de $\beta = 50$ grafos. A média de iterações foi de $\vartheta = 8.02$, para um $\alpha = 401$. Considerando todos os valores de parâmetros encontrados durante o processo, o menor e o maior valor obtido para p foi representado

pelo intervalo $Ip_t = [0.760, 0.808]$, e para q, pelo intervalo: $Iq_t = [0.192, 0.240]$. A média final de todos os parâmetros obtidos foram $pm_t = 0.783220$ e $qm_t = 0.216781$. Analisando somente os parâmetros nos 50 casos considerados como acertos, tem-se como intervalos $Ip_a = [0.764, 0.802]$ e $Iq_a = [0.192, 0.236]$. Os valores médios para os 50 acertos de β, foram $pm_a = 0.781200$ e $qm_a = 0.218800$.

Note que a diferença entre a média dos parâmetros quando houve acerto e a média geral foi somente de: $d_p = |0.783220 - 0.781200| = 0.00202$ e $d_q = |0.216781 - 0.218800| = 0.002019$. Esta diferença mínima será obtida em todos os cenários posteriores tornando válidas as duas formas de avaliação dos parâmetros.

Utilizando as médias finais pm_a e qm_a como valores fixos para p e q no modelo *Markoviano*, necessitou-se de um $\alpha = 93$ grafos para se obter 50 acertos $M_{y^*} = M_{y_A}$, ou seja, para $\beta = 50$. Nota-se que neste caso não se está corrigindo o vetor w, mas somente estão sendo gerados novos grafos a partir de pm_a e qm_a, ou seja, $M_{y^*} = [pm_a, qm_a].M_{x_B}$, para pm_a e qm_a fixos. Uma média ϑ de 1.86 iterações foram necessárias para cada correspondência exata de grau médio.

Ou seja, existem duas etapas principais no processo. A primeira refere-se à obtenção de um vetor w estável, através de sua contínua correção, até se atingir 50 acertos, para então ser calculada a média do vetor w. A segunda refere-se ao processo de geração de novos 50 grafos, utilizando-se o vetor $w = [pm_a, qm_a]$ fixo, calculado anteriormente, de modo a verificar sua qualidade.

Outra importante medida é o grau do erro absoluto ε para cada grafo gerado. Este erro é definido como o módulo da diferença entre o grau total obtido em algum grafo gerado G_t e o grau do grafo desejado G_f. No exemplo de aprendizado apresentado anteriormente obteve-se $\varepsilon < 3$ em 95% dos grafos.

A seguir são apresentados resultados com o objetivo de verificar a variação dos valores dos parâmetros p e q e sua eficácia no processo de geração. Neste sentido, observou-se que os valores médios pm_a e qm_a tendem a convergir para os mesmos resultados, independentemente dos valores iniciais.

Considere novamente os mesmos grafos B e A e o mesmo processo experimental descrito acima executado desde o início. Neste segundo treinamento, os valores de p e q obtidos foram, respectivamente, 0.659375 e 0.340625 para a primeira convergência. Considerando $\beta = 50$, necessitou-se de um $\alpha = 1204$ para estes valores de p e q, com $\vartheta = 20.48$. Em 95% dos casos tem-se $\varepsilon < 11$ e somente em 45% dos grafos tem-se $\varepsilon < 3$. Observe que estes valores estão aquém da eficiência desejada.

Verificando-se agora novamente a contínua correção de w durante a geração dos grafos, foram necessárias 370 iterações, ou seja, $\alpha = 370$, resultando em um $\vartheta = 3.40$. Ao final desse novo processo se obteve $Ip_t = [0.659375, 0.83125]$ e $Iq = [0.16875, 0.340625]$. A

média final de todos os parâmetros obtidos foram $pm_t = 0.809228$ e $qm_t = 0.190072$. Se forem analisados somente os parâmetros das vezes que ocorreram acertos, $Ip_a = [0.659375, 0.83000]$ e $Iq_a = [0.192, 0.340625]$. As novas médias obtidas entre os 50 acertos foram $pm_a = 0.797750$ e $qm_a = 0.202250$.

Para testar a eficiência de p e q para essa nova simulação, uma nova sequência $\beta = 50$ grafos foi gerada, utilizando as médias finais pm_a e qm_a como parâmetros de entrada. O total de grafos necessários reduziu-se para $\alpha = 289$, gerando um $\vartheta = 5.78$, uma diminuição considerável se comparado com o ϑ anterior de 20.48. Analisando o erro têm-se: $\varepsilon < 6$ em 95% das vezes e $\varepsilon < 3$ em 80% das vezes.

Considerando as duas simulações apresentadas, notou-se que os valores médios pm_a e qm_a convergem para resultados próximos. Esta convergência se torna mais precisa quando se aumenta o número de correções, atribuindo-se um valor superior para β. Como exemplo, para $\beta = 100$ a diferença de valores de pm_a e qm_a se reduziu para menos de 0.009. E no final, ambos obtiveram $\varepsilon < 3$ em 95% dos casos.

Estes resultados validam a utilização dos valores médios das variáveis para a parametrização do modelo *markoviano*. A explicação para este fato se dá através da Lei dos Grandes Números, onde se a probabilidade de um certo evento é ρ e se n tentativas independentes são feitas com k sucessos, então $k/n \to \rho$ se $n \to \infty$. Assim, se o custo computacional não for uma restrição, pode-se optar pela obtenção dos valores médios de p e q para valores superiores de β acertos, e posterior utilização desses valores para a geração de sequências de grafos que obedeçam a dinâmica do modelo de *Markov*, descrito na seção 6.2.

6.4.2 Aprendizado de um Grafo Mais Denso para um Menos Denso

Considere os mesmos grafos descritos no primeiro cenário (seção 6.4.1). Entretanto, neste segundo cenário, o grafo inicial G_0 é o representado pelo grafo A e o final G_f, representado pelo grafo B.

Observa-se que não mais será analisado os valores p e q do primeiro acerto para fazer-se a verificação de eficiência, pois como demonstrado no cenário anterior, isto pode não ser adequado. Portanto, serão trabalhados somente os valores médios, neste cenário e nos posteriores.

Um total de $\alpha = 444$ grafos foram gerados durante o processo de aprendizado, para se alcançar o total de $\beta = 50$ acertos, resultando em $\vartheta = 8.88$. Considerando todos os valores de p e q, tem-se $Ip_t = [0.109375, 0.44375]$ e $Iq_t = [0.55625, 0.890625]$. Os valores médios são: $pm_t = 0.164894$ e $qm_t = 0.831858$.

Considerando somente os acertos obtidos, tem-se $Ip_t = [0.121875, 0.184375]$ e $Iq_t = [0.815625, 0.878125]$, os valores médios alcançados foram $pm_a = 0.159188$ e $qm_a =$

0.820812.

Seguindo a metodologia apresentada no primeiro cenário, a qualidade da predição foi avaliada com a utilização dos valores das médias finais pm_a e qm_a como parâmetros de entrada. Para $\beta = 50$ necessitou-se de $\alpha = 174$ grafos gerados, resultando em um fator de correspondência $\vartheta = 3.48$.

Analisando o erro têm-se: $\varepsilon < 5$ em 100% das vezes e $\varepsilon < 2$ em 80% das vezes.

6.4.3 Aprendizado para um Grafo Manter-se Estável Utilizando o Processo de Formação *Markoviano*

Neste cenário é realizado o treinamento cujo objetivo é manter o valor do grau médio. Considere a Figura 24, onde $G_0 = G_f = E$. Por conseguinte: $M_{x_E} = M_{y_E}$, com graus $d_m = 3.75$ e $d_t = 30$.

$$E = \begin{matrix} 01101000 \\ 10111000 \\ 10001011 \\ 00001011 \\ 10110111 \\ 11001001 \\ 00111000 \\ 00111000 \end{matrix}$$

Figura 24 – Matriz de adjacência M_{x_E} e M_{y_E} correspondente ao grafo E.

Durante a correção obteve-se $\alpha = 395$ grafos gerados para $\beta = 50$ acertos, resultando em um fator $\vartheta = 7.90$. Considerando todos os grafos gerados, tem-se $Ip_t = [0.400000, 0.521875]$ e $Iq_t = [0.478125, 0.600000]$. Os valores médios alcançados foram: $pm_t = 0.451733$ e $qm_t = 0.548267$.

Considerando somente os acertos obtidos têm-se: $Ip_a = [0.421875, 0.490625]$ e $Iq_a = [0.509375, 0.578125]$. Os valores médios alcançados foram $pm_a = 0.453187$ e $qm_a = 0.546813$.

Em relação a qualidade da predição, utilizando-se as médias finais pm_a e qm_a, foram necessários $\alpha = 99$ grafos gerados para $\beta = 50$ acertos, resultando em um fator $\vartheta = 1,98$. Em 90% dos grafos gerados $\varepsilon < 3$.

6.4.4 Estipulando uma Topologia Fixa para o Aprendizado

Com o método proposto, é possível estipular uma topologia específica no grafo inicial G_0 que deve ser preservada durante o processo de aprendizado. A ideia é que esta topologia específica, ao final do treinamento, garanta o valor de grau médio esperado.

Neste cenário, como pode ser observado na Figura 25, $G_0 = F$, com matriz correspondente M_{x_F}, possuindo graus $d_t = 35$ e $d_m = 4.375$. O grafo final $G_f = A$, possuindo graus $d_t = 50$ e $d_m = 6.25$, com matriz correspondente M_{y_A}, conforme Figura 23. A seguinte topologia deve ser mantida: os nós 0, 1 e 2 devem permanecer com as mesmas ligações (arestas) durante o processo de aprendizado.

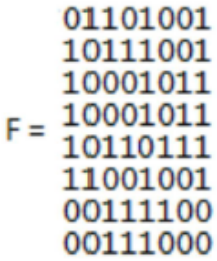

Figura 25 – Matriz de adjacência M_{x_F} correspondente ao grafo F.

Neste cenário necessitou-se de $\alpha = 168$ graus gerados para um total de $\beta = 50$ acertos, resultando em um fator $\vartheta = 3.36$. Tem-se: $Ip_t = [0.909375, 0.95000.]$, $Iq_t = [0.05000, 0.090625]$, $pm_t = 0.931409$ e $qm_t = 0.068591$.

Considerando somente os acertos obtidos com grau médio $d_m = 1.25$ obteve-se: $Ip_a = [0.909375, 0.956875]$. e $Iq_a = [0.05000, 0.090625]$. Os valores médios alcançados foram $pm_a = 0.929289$ e $qm_a = 0.70711$.

Observa-se, na Figura 26, que foi necessário o preenchimento de quase todas as ligações restantes, refletido no alto valor de p, para que os graus fossem igualados, uma vez que não era possível acrescentar arestas nos três primeiros nós.

Figura 26 – Exemplo de matriz resultante do processo de aprendizado de F em A.

Utilizando as médias finais pm_a e qm_a como parâmetros de entrada da geração de grafo *markoviano*, a quantidade de grafos gerados α reduziu de 168 para 78. Obteve-se um fator $\vartheta = 1.56$, resultando em $\varepsilon < 3$ graus em 95% das vezes.

6.4.5 Testando a Escalabilidade do Algoritmo

Para testar a escalabilidade do algoritmo aumentou-se a ordem de grandeza do grafo para $|V| = 10^2$ nós. Considere um grafo inicial G_0 possuindo 2182 arestas, e um

grafo final G_f com 7818 arestas. Assim, 5636 arestas precisam ser inseridas, para que o grau médio torne-se equivalente. Em G_0, tem-se d_m =21.82 e em G_f tem-se d_m =78.18. Após 17 iterações, o grafo gerado a partir de G_0 obteve um total de 7916 arestas, uma diferença de somente 98 arestas se comparado com o grafo G_f. Os valores parciais de probabilidades foram p =0.777820 e q =0.22218.

Nas trintas iterações seguintes o erro absoluto ε ficou entre 70 e 300 graus, com média igual a 129. Os valores médios alcançados foram $pm_t = 0.778920$ e $qm_t = 0.221080$.

Nas cinquenta iterações seguintes o erro absoluto ε estabilizou-se entre 30 e 110, com média igual a 45. Ao final desta simulação, os valores médios alcançados foram $pm_t = 0.779620$ e $qm_t = 0.220380$.

Para um total de 275 iterações, o erro tornou-se zero, sendo obtidos os valores $p = 0.778900$ e $q = 0.221100$. Observa-se que estes valores apresentam pouca variação em relação aos valores obtidos anteriormente, com um número inferior de iterações. Uma possível alternativa para esta situação de erro inerentemente probabilístico seria limitar o número de iterações, adotando-se uma margem de erro aceitável para o problema. Por exemplo, se na definição do conceito de β fosse relaxada a condição de correspondência exata, admitindo-se um erro relativo $\varepsilon_r < 1\%$, obter-se-ia uma convergência em um número reduzido de iterações.

Numa nova instância do processo de aprendizado, considerando o erro relativo $\varepsilon_r < 1\%$, foram necessários $\alpha = 199$ grafos gerados para $\beta = 50$ acertos, resultando em um fator $\vartheta = 3.98$. Os valores médios alcançados foram $pm_a = 0.779000$ e $qm_a = 0.221000$.

Aplicando-se estes valores na geração de novos grafos, necessitou-se de 144 iterações para a obtenção de 50 acertos, resultando em um fator $\vartheta = 2.28$.

6.4.6 Abordagem Estruturada Mista

É possível, também, utilizar no processo de aprendizado mais de um par de grafos como objetos de entrada.

Seja $S = \{(G_{01}, G_{f1}), (G_{02}, G_{f2}), (G_{03}, G_{f3}), \ldots, (G_{0m}, G_{fm})\}$. Considerou-se como grafo de saída um único grafo objetivo G_{fi}, com $i = 1, \ldots, m$.

Num cenário com essas características, utilizou-se como grafos iniciais as correspondentes matrizes de adjacência M_{x_C} (Figura 27), com grau $d_t = 32$; M_{x_D}, com grau $d_t = 26$; e M_{y_B}, com grau $d_t = 10$. Para $\beta = 50$, necessitou-se $\alpha = 544$ iterações, resultando em um fator $\vartheta = 10.88$. Os intervalos obtidos foram $Ip_t = [0.112500, 0.446875]$ e $Iq_t = [0.553125, 0.887500]$. As médias finais alcançadas foram: $pm_t = 0.146369$ e $qm_t = 0.849954$. Considerando somente os acertos, obteve-se $Ip_a = [0.115625, 0.181250]$ e $Iq_a = [0.818750, 0.884375]$. As médias finais alcançadas foram: $pm_a = 0.141750$ e

$qm_a = 0.838250$.

$$C = \begin{matrix} 01111100 \\ 10111100 \\ 11000111 \\ 11000111 \\ 11000000 \\ 11110010 \\ 00110100 \\ 00110000 \end{matrix}$$

Figura 27 – Matriz de adjacência correspondente ao grafo C.

Utilizando-se estes valores, necessitou-se de um total de $\alpha = 59$ iterações para $\beta = 50$ acertos. Obteve-se um fator $\vartheta = 1.18$ resultando em $\varepsilon < 3$ graus em 99% das vezes.

6.5 Considerações

A aplicação em redes complexas dinâmicas foi formulada de acordo com o novo algoritmo denominado *Perceptron Estruturado com Margem Zero*, seção 4.2, derivado diretamente do *Perceptron Estruturado*, seção 4.1. O mesmo foi aplicado a solução de um problema relacionado a predição de parâmetros probabilísticos em possíveis redes complexas varmarkovianas.

Como visto na seção 6.4.6, a partir de diferentes estruturas de grafos, é possível que sejam estimados parâmetros que os façam convergir para um mesmo grafo objetivo. Isto decorre em função do próprio processo de aprendizado, cujo fator de mudança dos parâmetros se dá através do grafo final. Consequentemente, a medida que se aumenta o número de iterações ou o tamanho do aprendizado, menos peso os grafos iniciais terão no processo de convergência. Se o número de iterações estender-se indefinidamente, sua influência torna-se nula.

Trata-se de estudo ainda em desenvolvimento, no entanto, inovador ao possibilitar novas formas de aprendizado a partir da extração dos parâmetros utilizados para se criar uma rede complexa, real ou não, com determinadas características. Podendo-se futuramente inserir outros tipos de processo de formação de redes que não o *markoviano*. O estudo se encaminha para a extração de características além unicamente do grau dos nós. Em primeiro lugar está sendo analisado o parâmetro clusterização, pois por ter tanto o caráter local quanto o global facilita a utilização do algoritmo de aprendizado proposto. Posteriormente, espera-se que seja possível expandir a proposta para qualquer característica relevante a um grafo e aplicá-las a redes reais conhecidas da literatura, consequentemente aumentando-se ainda mais a dimensão das redes, verificando o grau de escalabilidade. Espera-se também estender a formulação teórica, já estando em desen-

volvimento o *Perceptron Estruturado Regressor*, de modo a englobar entradas e saídas variadas de modo a minimizar um determinado erro ε. Calculando, desse modo, um vetor w ótimo que minimize a diferença entre as saídas.

Apesar de ser ainda uma aplicação em sua fase inicial, outro resultado relevante obtido neste estudo foi a descoberta de possíveis parâmetros de entrada para manter a rede com determinadas características constantes, ao mesmo tempo que permite que outras sejam modificadas com o tempo, como apresentado no cenário da seção 6.4.3. Neste caso, o grafo E que continua a gerar indefinidamente novos grafos com a característica determinada inicialmente, neste caso, o grau.

Os resultados apresentados indicam que esta nova abordagem sugerida para resolver o problema de predição de parâmetros probabilísticos na geração de redes complexas não somente é viável, como também é eficiente, tanto em termos de esforço computacional quanto da qualidade da predição.

7 Conclusões e Trabalhos Futuros

Este trabalho concentrou-se no desenvolvimento de estratégias baseadas no modelo *Perceptron* visando a aplicação em predição estruturada. Todas as técnicas de solução apresentadas foram testadas em diferentes exemplos, mostrando a aplicabilidade e potencial de cada estratégia.

A construção de modelos baseados nas variáveis primais foi o enfoque inicial do desenvolvimento. Neste quesito, contribuições de grande interesse foram obtidas. Desde o modelo do *Perceptron Estruturado com Margem Fixa* até o modelo baseado em *Margem Incremental*, que mostrou resultados de melhor qualidade. É interessante ressaltar, também, o modelo de *Margem Zero*, importante para aplicações com apresentação de entradas únicas. Posteriormente, partiu-se para desenvolvimentos relativos a formulação dual do problema e a inclusão de funções *Kernel*, utilizadas quando uma hipótese baseada em uma classe de funções lineares não é capaz de resolver o problema no espaço de entrada.

Esta etapa é considerada o grande desafio do trabalho, visto que modelos estruturais duais estão ainda em nível incipiente de pesquisa. A consistência do modelo apresentado indica um direcionamento adequado para o embasamento do *Perceptron Estruturado Dual*.

Os modelos duais foram bastante importantes para o entendimento do escopo dos problemas analisados. A própria constatação que um problema estruturado é ou não linearmente separável já representa certa dificuldade. Talvez, uma maneira prática de se visualizar isto seja executando o problema na sua forma primal. Se o mesmo não apresentar margem positiva num tempo viável, provavelmente o problema não é linearmente separável. Caso este mesmo problema seja resolvido na forma dual para algum *Kernel*, confirma-se, pelo menos empiricamente, esta suposição.

Em relação ao problema de predição de custos em planejamento de caminhos, este foi extensamente pesquisado, conforme pode ser visto neste trabalho, tanto em mapas reais quanto em mapas artificiais. A maior dificuldade encontrada foi no que se refere a obtenção de mapas reais, sua discretização e posterior análise das características presentes nas células. Este processo foi em parte manual, exigindo a extração e análise da paleta de cores de cada mapa. Certamente, seria muito mais eficiente a utilização de um programa de segmentação de imagens. No entanto, até o momento, não foi encontrado nenhum código aberto adaptável, visto que apesar de existirem muitas pesquisas de pós-graduação relativos à elaboração deste programa, nenhuma se encontrou disponível para utilização.

Cada exemplo apresentou seus próprios resultados e conclusões, no entanto, de uma forma geral observou-se que o aprendizado efetivamente ocorreu e que em cada um

dos testes correspondeu às expectativas do especialista de forma eficiente.

Outro problema tratado, com grande potencial de desenvolvimento, é o referente aos grafos de *Markov*. Foi abordado até o momento a predição baseada somente em uma medida de centralidade ou característica: o grau médio do grafo. Porém, existe a possibilidade de estender este trabalho considerando a utilização de outra característica: o grau de clusterização global. Ou seja, calcula-se os parâmetros p e q de uma rede de *Markov* para que seja possível obter algum grau de clusterização desejado. Isto poderá ser feito seguindo a mesma lógica desenvolvida na seção 6.2, alterando-se basicamente a função φ, que retorna um vetor de duas posições: a quantidade de ligações existentes e a quantidade de ligações complementares. Esta nova abordagem irá retornar o grau de clusterização global e seu valor complementar, visto que o grau máximo de clusterização é 1. Em ambos os casos, p e q atuam localmente, modificando a presença ou a ausência de uma aresta. Resumindo, a mudança local, proveniente de p e q, reflete na alteração do valor global relacionado.

Conforme explicado, segundo o próprio aspecto da função estruturada, é necessário encontrar parâmetros locais que reflitam parâmetros globais, tais como o custo das células cujo somatório reflete o custo de um caminho ou a medida de grau dos nós que reflete o grau médio total do grafo. Tem-se este facilitador em alguns casos, mas para outros casos, não. É necessário então modificar o processo de aprendizado ou buscar outro método de geração, onde seja possível, mesmo implicitamente, levar novas considerações durante o cálculo da saída ótima. Deste modo, viabiliza-se como trabalho futuro o estudo relacionado a predição conjunta de características globais que possuem respectivos parâmetros locais.

Finalizando, considera-se também, como possibilidade de trabalhos futuros, a extensão dessa abordagem ao problema de regressão estruturada.

Referências

AIZERMAN, M. A.; BRAVERMAN, E. A.; ROZONOER, L. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, v. 25, p. 821–837, June 1964. 27

ALBERT, R.; BARABáSI, A. L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, v. 74, n. 1, p. 47–97, 2002. 85

ARONSZAJN, N. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, v. 68, n. 3, p. 337–404, 1950. Disponível em: <http://dx.doi.org/10.2307/1990404>. 27

BAKIR, G. et al. *Predicting Structured Data*. 1. ed. Massachusetts: MIT Press, 2007. ISBN 9780262026178. Disponível em: <http://books.google.com.br/books?id=b1EFKUoFF8IC>. 14, 29

BARABASI, A. L.; ALBERT, R. Emergence of scaling in random networks. *Science*, v. 286, p. 509–512, 1999. 85

BARABáSI, A.-L.; ALBERT, R.; JEONG, H. Mean-field theory for scale-free random networks. jul. 1999. Disponível em: <http://arxiv.org/abs/cond-mat/9907068>. 85

BELLMAN, R. E. *Dynamic Programming*. 1. ed. [S.l.]: Princeton University Press, 1957. ISBN 0486428095. 116

BERTSEKAS, D. *Minimization Methods for Non-differentiable Functions*. 1. ed. [S.l.]: Shor, Naum Z., 1985. ISBN 0-387-12763-1. 130, 131

BERTSEKAS, D. *Convex Analysis and Optimization*. 1. ed. [S.l.]: Athena Scientific, 2003. ISBN 1886529450. 39

BOLFARINE, H.; BUSSAB, W. de O. *Elementos de amostragem*. Edgard Blücher, 2005. ISBN 9788521203674. Disponível em: <http://books.google.com.br/books?id=a_fqPwAACAAJ>. 89

BOSER, B. E.; GUYON, I.; VAPNIK, V. A training algorithm for optimal margin classifiers. In: HAUSSLER, D. (Ed.). *COLT*. ACM, 1992. p. 144–152. ISBN 0-89791-497-X. Disponível em: <http://dblp.uni-trier.de/db/conf/colt/colt1992.html>. 27

BOYD, B.; L., X.; MUTAPCIC, A. Subgradient methods, stanford university, autumn. *Notes for EE392o*, 2003. 131

BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. 1. ed. Cambridge: Cambridge University Press, 2004. ISBN 0521833787. 123

BRINKHUIS, J.; TIKHOMIROV, V. *Optimization: Insights and Applications*. 1. ed. Princeton: Princeton University Press, 2005. ISBN 0691102872. 123

COELHO, M. A. N.; NETO, R. F.; BORGES, C. C. H. Predição de dados estruturados utilizando a formulação de máxima margem com aplicação em planejamento de caminhos. *Congresso Ibero-Latino Americano de Métodos Computacionais em Engenharia - 30º CILAMCE*, Novembro 2009. 39, 42

COELHO, M. A. N.; NETO, R. F.; BORGES, C. C. H. Perceptron models for online structured prediction. *International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2012*, p. 320–327, Agosto 2012. 39, 77, 80, 85

COELHO, M. A. N. et al. Estratégia online para predição estruturada em redes complexas. *Brazilian Congress on Computational Intelligence, CBIC 2013*, p. 1, Setembro 2013. 86

CRAMMER, K.; SINGER, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research 2*, p. 265–292, 2001. 34, 36, 37

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2. ed. New York: Wiley-Interscience, 2001. ISBN 0471056693. 20

ENGL, H. W.; HANKE, M.; NEUBAUER, A. *Regularization of inverse problems*. Dordrecht, Boston: Kluwer Academic Publishers, 1996. L'édition brochée porte la date de copyright 2000. ISBN 0-7923-6140-7. Disponível em: <http://opac.inria.fr/record=b1092144>. 133

EVLAMPIEV, K.; ISAMBERT, H. Modeling protein network evolution under genome duplication and domain shuffling. *BMC Systems Biology*, v. 1, n. 1, p. 49, 2007. ISSN 1752-0509. Disponível em: <http://www.biomedcentral.com/1752-0509/1/49>. 85

FIENBERG, S.; WASSERMAN, S. An exponential family of probability distributions for directed graphs: Comment. *Journal of the American Statistical Association*, jan. 1981. Disponível em: <http://links.jstor.org/sici?sici=0162-1459(198103)76%253A373%253C54%253AAEFOPD%253E2.0.CO%253B2-J>. 85

FRANK, O.; STRAUSS, D. Markov graphs. *Journal of the American Statistical Association*, American Statistical Association, v. 81, n. 395, p. 832–842, set. 1986. ISSN 0162-1459. Disponível em: <http://links.jstor.org/sici?sici=0162-1459%28198609%2981%3A395%3C832%3AMG%3E2.0.CO%3B2-C>. 85, 87

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, v. 29, p. 1189–1232, 2000. 50

GARCIA, M. A.; SILVA, A. P. C. da; MEO, M. Using hidden markov chains for modeling p2p-tv traffic. In: *GLOBECOM*. IEEE, 2010. p. 1–6. ISBN 978-1-4244-5638-3. Disponível em: <http://dblp.uni-trier.de/db/conf/globecom/globecom2010.html>. 85

GIRVAN, M.; NEWMAN, M. E. J. Community structure in social and biological networks. *PNAS*, v. 99, n. 12, p. 7821–7826, June 2002. 85

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização Combinatória e Programação Linear*. 2. ed. [S.l.]: Campus / Elsevier, 2005. ISBN 8535215204. 106

HALMOS, P. *Introduction to Hilbert Space: And the Theory of Spectral Multiplicity*. AMS Chelsea Publishing, American Mathematical Society, 1957. ISBN 9780821813782. Disponível em: <http://books.google.com.br/books?id=4hHEv4nNDokC>. 27

HAYKIN, S. *Redes Neurais Princípios e Práticas*. 2. ed. Porto Alegre: Bookman Companhia ED, 2001. ISBN 9788573077186. 19

JIN, E.; GIRVAN, M.; NEWMAN, M. E. J. Structure of growing social networks. *Physical Review E*, jan. 2001. Disponível em: <http://link.aps.org/doi/10.1103/PhysRevE.64.046132>. 85

JONES, J. H. H.; HANDCOCK, M. S. An assessment of preferential attachment as a mechanism for human sexual network formation. *Proceedings. Biological sciences / The Royal Society*, v. 270, n. 1520, p. 1123–1128, jun. 2003. ISSN 0962-8452. Disponível em: <http://dx.doi.org/10.1098/rspb.2003.2369>. 85

KIVINEN, J.; SMOLA, A. J.; WILLIAMSON, R. C. Online learning with kernels. *IEEE Transactions on Signal Processing*, n. 52, p. 2165–2176, Agosto 2002. 21

LEITE, S. C.; NETO, R. F. Incremental margin algorithm for large margin classifiers. *Neurocomputing*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, v. 71, n. 7-9, p. 1550–1560, 2007. ISSN 0925-2312. 4, 7, 20, 21, 22, 23, 33, 46, 53

LEMARECHAL, C. Nondifferentiable optimization. *Elsevier Science Publishers, North-Holland*, p. 529–572, 1989. 131

LIMA, A. Uma Estratégia de Decomposição por Relaxação Lagrangeana para a Otimização da Programação Diária da Operação de Sistemas Hidrotérmicos com Modelagem Detalhada da Rede Elétrica. In: . [S.l.]: Aplicação ao Sistema Brasileiro, Tese (Doutorado), COPPE/UFRJ, 2007. 131

LLOYD, A.; MAY, R. How viruses spread among computers and people. *Science*, v. 292, n. 5520, p. 1316–1317, 2001. 85

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, p. 115–133, 1943. 17

MERCER, J. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, v. 209, p. 415–446, 1909. 26, 27

MüLLER, K.-R. et al. An introduction to kernel-based learning algorithms. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, v. 12, n. 2, p. 181–201, 2001. 25, 26, 27

MOLLOY, M.; REED, B. The size of the giant component of a random graph with a given degree sequence. *Combin. Probab. Comput.*, v. 7, p. 295, 1998. 85

NEMHAUSER, G.; WOLSEY, L. *Integer and Combinatorial Optimization*. 1. ed. [S.l.]: Wiley-Interscience Series in Discrete Mathematics and Optimization, 1999. ISBN 978-0471359432. 131

NEWMAN, M. E. J. The structure and function of networks. *Computer Physics Communications*, v. 147, p. 40–45, 2001. 85, 86

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2. ed. Cambridge: Cambridge University Press, 2006. ISBN 978-0-387-30303-1. 126

NOVIKOFF, A. B. On convergence proofs on perceptrons. In: *Proceedings of the Symposium on the Mathematical Theory of Automata*. New York, NY, USA: Polytechnic Institute of Brooklyn, 1962. v. 12, p. 615–622. 19

PLATT, J. C. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: *Advances in Large Margin Classifiers*. [S.l.]: MIT Press, 1999. p. 61–74. 50, 134

Ramon Ferrer i Cancho; JANSSEN, C.; SOLÉ, R. V. *Topology of Technology Graphs: Small World Patterns in Electronic Circuits*. 2001. Physical Review E 64 046119. 85

RATLIFF, N. Learning to search: Structured prediction techniques for imitation learning. In: . [S.l.]: Dissertação (Doctor of Philosophy in Robotics), Carnegie Mellon University, 2009. 14, 50, 134

RATLIFF, N.; BAGNELL, J. A.; ZINKEVICH, M. Maximum margin planning. *Twenty Second International Conference on Machine Learning. ICML06*, p. 729–736, 2006. 7, 48, 49, 60

RATLIFF, N. et al. *Boosting Structured Prediction for Imitation Learning, Paper 54*. 2007. 1-11 p. Disponível em: <http://repository.cmu.edu/robotics/54>. 50, 134

ROCKAFELLAR, R. T. *Convex analysis*. 1. ed. Princeton: Princeton University Press, 1970. ISBN 0691015864. 129

RODRIGUES, S. Relaxação Lagrangeana e Subgradientes com Dilatação de Espaço Aplicados a um Problema de Grande Porte. In: . [S.l.]: Tese, COPPE/UFRJ), 1994. 131

ROSENBLATT, F. *The perceptron: A theory of statistical separability incognitive systems (Project PARA)*. U.S. Department of Commerce, Office of Technical Services, 1958. Disponível em: <http://books.google.com.br/books?id=gfmJcAAACAAJ>. 17

RUSSEL, S. J.; NORVING, P. *Inteligência Artificial*. 2. ed. [S.l.]: Editora Campus, 2004. ISBN 8535211772. 59, 106, 109, 112, 113

SCHoeLKOPF, B.; SMOLA, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. Cambridge, MA: MIT Press, 2002. 27

SCHRIJVER, A. *Combinatorial Optimization: Polyhedra and efficiency*. 1. ed. [S.l.]: Springer Verlag NY, 2003. ISBN 3540443894. 30, 35

SILVA, A. P. C. da et al. Quality assessment of interactive voice applications. *Computer Networks*, v. 52, n. 6, p. 1179–1192, 2008. Disponível em: <http://dblp.uni-trier.de/db/journals/cn/cn52.html>. 85

SOLOMONOFF, R.; RAPOPORT, A. Connectivity of random nets. v. 13, p. 107–117, 1951. 85

SOUZA, T. C. A. d. Métodos subgradientes em otimização convexa não diferenciável. In: . [S.l.]: Universidade Federal de Juiz de Fora, Dissertação (Mestrado em Modelagem Computacional), 2008. 130, 131

TASKAR, B. Learning Structures Prediction Models: A Large Margin Approach. In: . [S.l.]: Dissertação (Doctor of Philosophy), Stanford University, 2004. 14, 28, 29, 33, 35

TASKAR, B. et al. Learning structures prediction models: A large margin approach. *Twenty Second International Conference on Machine Learning. ICML05*, p. 896–903, 2005. 15, 29, 36, 37, 50, 134

TASKAR, B.; GUESTRIN, C.; KOLLER, D. Max-margin markov networks. *Neural Information Processing Systems*, 2003. 15

TSOCHANTARIDIS, I. et al. Support vector machine learning for interdependent and structured output spaces. In: ACM. *Proceedings of the twenty-first international conference on Machine learning*. [S.l.], 2004. p. 104. 54

TSOCHANTARIDIS, I. et al. Large margin methods for structured and interpendent output variables. *Journal of Machine Learning*, p. 1453–1484, 2005. 15, 34, 37, 135

URRUTY, J. B. H.; LEMARECHAL, C. *Fundamentals of Convex Analysis*. 1. ed. [S.l.]: Springer, 2001. ISBN 3-540-42205-6. 129

VAPNIK, V. *Statistical Learning Theory*. 1. ed. [S.l.]: Wiley and Sons Inc, 1998. ISBN 0471030031. 22, 33, 37

WATTS, D.; STROGATZ, S. Collective dynamics of 'small-world' networks. *Nature*, n. 393, p. 440–442, 1998. 85

WATTS, D. J. *Small worlds: the dynamics of networks between order and randomness*. Princeton, NJ, USA: Princeton University Press, 1999. ISBN 0-691-00541-9. 85

WESTON, J.; SCHÖLKOPF, B.; BOUSQUET, O. Joint kernel maps. In: *Computational Intelligence and Bioinspired Systems*. [S.l.]: Springer, 2005. p. 176–191. 54

WESTON, J.; WATKINS, C. Multi-class support vector machines. *URL citeseer.ist.psu.edu/article/weston98multiclass.html*, 1998. 34, 36, 54, 55, 58

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. Defense Technical Information Center, 1960. 40, 121

WINK, O.; NIESSEN, W.; VIERGEVER, M. Minimum cost path determination using a simple heuristic function. In: SANFELIU, A. et al. (Ed.). *Image, speech and signal processing*. Los Alamitos: IEEE computer society press, 2000. v. 3, p. 1010–1013. 110

XU, Z.; HARRISS, R. Exploring the structure of the U.S. intercity passenger air transportation network: a weighted complex network approach. *GeoJournal*, v. 73, n. 2, p. 87–102, out. 2008. Disponível em: <http://dx.doi.org/10.1007/s10708-008-9173-5>. 85

YANN, L. et al. Energy-based models: Structured learning beyond likelihoods. *Neural Information Processing systens Foundation*, 2006. 15

Apêndices

APÊNDICE A – Busca de Caminhos

A.1 Planejamento de Caminhos

"A tarefa de apresentar uma sequência de ações que alcançarão um objetivo é chamada planejamento" (RUSSEL; NORVING, 2004).

O planejamento de caminhos começa com a percepção dos dados relevantes à resolução do problema, em seguida tem-se a análise desses dados e a determinação da função de avaliação com seus custos ou recompensas como base para a escolha de um caminho, e por último a determinação de quais ações devem ser empregadas para que tal caminho possa ser percorrido.

A.2 Otimização Combinatória e Problema do Caminho Mínimo

Problemas de otimização objetivam maximizar ou minimizar uma função definida sobre um domínio. A teoria clássica de otimização trata do caso em que o domínio é infinito. No caso dos problemas de otimização combinatória, o domínio é tipicamente finito. Em geral é fácil listar os seus elementos e testar se um dado elemento pertence a esse domínio. Porém, testar todos os elementos deste domínio na busca pelo melhor mostra-se inviável na prática para a maioria dos problemas (GOLDBARG; LUNA, 2005).

Como exemplos tem-se o problema da mochila, o problema do caixeiro viajante e o problema da satisfabilidade máxima. Eles possuem várias aplicações práticas: projeto de redes de telecomunicação, o empacotamento de objetos em *containers*, a localização de centros distribuidores, análise de dados, na economia (matrizes de entrada/saída), na física (estados de energia mínima), entre outras.

O caminho de custo mínimo é a sequência de ligações que se deve seguir do nó inicial até o nó final num grafo, cujo custo total é mínimo. É um dos problemas mais estudados em Otimização Combinatória. Existem quatro tipos de situações no problema do caminho mínimo, porém é abordado com ênfase só a primeira:

- caminho mínimo entre origem e destino;
- caminho entre a origem e os demais vértices;
- caminho mínimo entre todos os pares de vértices;
- k-ésimos caminhos mínimos entre um par de vértices: contingenciamento.

Na Figura 28 o caminho mínimo (ótimo) entre a e e é o caminho que passa pelos vértices a, b, c, d, e.

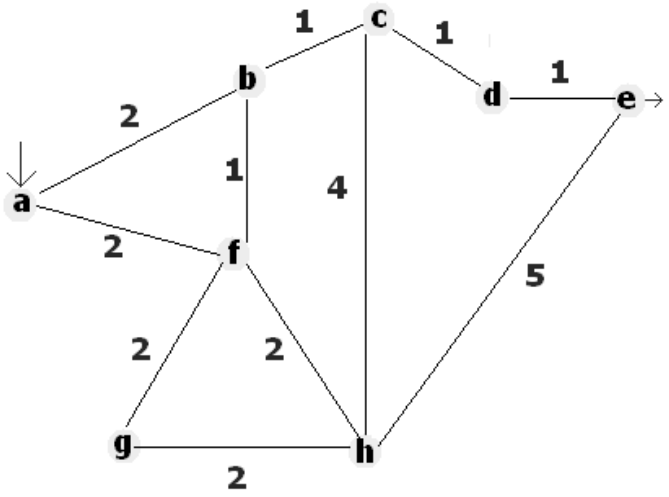

Figura 28 – Exemplo de um grafo com custos nas arestas

O problema do caminho mínimo entre dois vértices fixos pode ser modelado como um problema de fluxo compatível a custo mínimo. A quantidade de fluxo que passa no arco (i, j) é representado como $x_{i,j}$. Neste trabalho considera-se a passagem de um fluxo unitário na rede, deste modo o valor da função objetivo retrata o custo do caminho, ou seja, $x_{i,j} \in \{0, 1\}$ então se $x_{i,j} = 0$ o fluxo não passa pelo arco (i, j) e se $x_{i,j} = 1$ o fluxo passa pelo arco.

Não há restrições de capacidade no arco quando o fluxo é unitário, simplesmente passa ou não passa, ou seja, o fluxo é sempre compatível.

A modelagem matemática da forma primal do problema do caminho mínimo se apresenta da seguinte forma. Tomando $X = [x_{i,j}]$, com $i, j = 1, 2, ..., n$, como o vetor de variáveis, representando a quantidade de fluxo que transita no arco (i, j) e $C = [c_{i,j}]$ como o vetor de custos, representando o custo unitário desta transição, pode-se definir o problema na sua forma primal como:

$$Minimizar \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j}.x_{i,j} \tag{A.1}$$

Sujeito às restrições de balanceamento:

$$\sum_{j=1}^{n} x_{i,j} - \sum_{k=1}^{n} x_{k,i} = \begin{cases} 1 & \text{se } x_i \text{ for origem} \\ -1 & \text{se } x_i \text{ for destino} \\ 0 & \text{caso contrário} \end{cases} \tag{A.2}$$

e acrescido com as restrições de não-negatividade: $x_{i,j} \geq 0$.

Algumas considerações sobre a modelagem matemática: A variável $x_{i,j}$ representa a quantidade de fluxo que deve passar no arco (i, j). Observe que, pelo fato de ser imposta uma restrição de não-negatividade no valor dessas variáveis, os seus valores serão iguais a 0 ou 1. Como a quantidade de fluxo é unitária ($f = 1$) não há necessidade de representar as restrições de capacidade dos arcos no modelo. Se existir um arco de retorno, ligando o objetivo à origem, tem-se um problema de fluxo circulatório que simplifica as restrições de balanceamento, que fica na forma:

$$\sum_{j=1}^{n} x_{i,j} - \sum_{k=1}^{n} x_{k,i} = 0 \tag{A.3}$$

Nesse caso, a matriz de incidência recebe um arco adicional de capacidade infinita, ligando o destino à origem, e a variável $x_{n,1}$ recebe o valor 1. As restrições de balanceamento, também chamadas de equilíbrio, asseguram a conservação dos fluxos em cada vértice. As restrições de integridade (valores inteiros) são garantidas, pela propriedade de uni modularidade da matriz de incidência, ou seja, qualquer submatriz possuirá determinante com valores 0, 1 ou -1, garantindo, dessa forma, a obtenção de soluções inteiras nos sistemas algébricos que fazem a sua utilização.

Na forma compacta, ou seja, na representação matricial, pode-se definir o problema primal através da seguinte formulação:

Min $C.X$

Sujeito à:

$A.X = 0$,

$X \geq 0$

A utilização de uma técnica de programação linear, como exemplo, o método simplex, para solucionar esse problema, não se mostra eficiente na prática. Ao longo do trabalho serão vistas outras estratégias para a determinação do caminho de custo mínimo entre dois vértices. Entretanto, a formulação primal do problema é importante na obtenção de heurísticas, na formulação dual e, consequentemente, na derivação de diversos algoritmos.

Este problema se adapta a diversas situações práticas. Em roteamento, por exemplo, pode-se modelar os vértices do grafo como cruzamentos, os arcos como vias, e os custos associados aos arcos como o tempo de trajeto ou à distância percorrida, e a solução seria o caminho mais curto, ou o caminho mais rápido, entre os vértices. Em redes de computadores, os vértices podem representar equipamentos diversos, os arcos correspondem a trechos de cabeamento, e os custos poderão estar associados às taxas máximas de transmissão de dados. Neste caso, a solução seria a rota de transmissão mais rápida. Além disso, pode-se considerar o problema do caminho mínimo como um subproblema

presente na solução de diversos problemas de fluxo em redes, como exemplo, no processo de geração de colunas para a solução de problemas de distribuição de fluxos de diferentes mercadorias, modelado segundo a concepção de formulação de caminhos.

Outras possibilidades de aplicação do problema do caminho mínimo incluem quaisquer problemas envolvendo redes ou grafos em que se tenham grandezas (distâncias, tempos, perdas, ganhos, despesas, etc.) que se acumulam linearmente ao longo do percurso da rede e que alguém deseje minimizar. Neste trabalho o problema do caminho mínimo é aplicado ao problema de determinação de caminhos para um agente móvel, ou robô, em um ambiente caracterizado por um grid de células (ou estados) associados a um conjunto de características, para determinado estado objetivo ou vértice de destino.

A.3 Resolução de Problemas por meio da Busca

A formulação de problemas é um processo para decidir quais ações e estados devem ser considerados. Depois segue-se a busca: um processo de procurar uma sequência de ações e estados que o leve a seu objetivo.

Um algoritmo de busca recebe um problema como entrada, o ambiente do problema é representado por um espaço de estados. O algoritmo retorna uma solução sob uma forma de sequência de ações, com isso tem-se um projeto que consiste em formular, buscar e executar. Um problema pode ser definido formalmente em quatro componentes (RUSSEL; NORVING, 2004):

- O estado inicial em que o agente começa;
- Uma descrição das ações possíveis que estão disponíveis para o agente;
- O teste de objetivo, que determina se um estado é o objetivo;
- E uma função de custo de caminho, que atribui um valor numérico a cada caminho refletindo sua própria medida de desempenho.

Conceitos básicos para se entender um problema de busca: ao invés de se usar a notação $C_{x,y}$ para representar o custo do caminho, pode-se usar também $c(x,a,y)$, e levar em conta qual ação a foi executada para ir do estado x ao y. Uma solução para um problema de busca é um caminho desde o nó inicial até o final, e uma solução ótima tem o menor custo de caminho entre todas a outras. O espaço de estados é dividido em três conjuntos: os já visitados: **conjunto Fechados**, os candidatos: **conjunto Abertos** e os não-visitados: **conjunto Desconhecidos**. E o custo estimado do caminho mais econômico de um estado x até um estado objetivo y é conhecido como **componente heurística**. São apresentadas agora as definições de alguns termos que medem o desempenho da busca.

A.4 Medição de Desempenho da Busca

Para medir o desempenho durante uma busca te-se quatro fatores a considerar:

- Completeza: O algoritmo encontra uma solução se a mesma existir;
- Otimalidade: A solução retornada é ótima;
- Complexidade de tempo: quanto tempo leva para retornar uma solução;
- Complexidade de espaço: quanta memória é necessária pra efetuar a busca.

A Complexidade depende do fator de ramificação no espaço de estados, representado por b, e pela profundidade da solução, representado por d.

A determinação de um caminho ótimo entre dois nós em uma rede é um problema fundamental que recebeu atenção considerável de várias comunidades de pesquisa nos últimos quarenta anos, pois suas complexidades de tempo e espaço tornam a resolução de certos problemas impraticável. (WINK; NIESSEN; VIERGEVER, 2000).

O capítulo seguinte tem as explicações de dois tipos diferentes de formulações para se determinar um caminho mínimo: as abordagens *forward* e *backward*.

APÊNDICE B – Determinação de Caminhos

São discutidos agora os dois tipos gerais de determinação de caminhos, a busca *forward* e a busca *backward*, cada um com seus respectivos algoritmos.

B.1 Busca *Forward*

O problema de determinação de caminhos pode ser resolvido através de um processo de busca *forward*, ou, de expansão no espaço de estados do problema. A busca *forward*, ou direta, é aquela que determina um processo de exploração que caminha do estado inicial ou raiz do problema em direção aos estados finais que representam a solução do mesmo. Apesar deste processo caracterizar, normalmente, uma forma de planejamento, onde o conhecimento a priori do espaço de estados e das possíveis transições e respectivos custos sejam necessários, é possível a sua adaptação para a solução de problemas *online*, relacionados à determinação de caminhos em tempo real e em ambientes dinâmicos. Para isso, é importante descrever os algoritmos de busca.

Inseridos na busca *Forward* existem dois grandes conjuntos de tipos de buscas: as buscas sem informações, que não são abordadas por serem demasiadamente simplistas, e as buscas com informação, abordadas por possuírem melhor desempenho.

B.1.1 Algoritmo de Dijkstra

Dentro da área de otimização combinatória, o método mais usado para encontrar o caminho mínimo entre uma origem pré-fixada e os demais nós do grafo, quando não há arcos de custo negativo, é o **Algoritmo de Dijkstra**. Ou seja, o algoritmo de Dijkstra identifica, a partir de um nó do grafo, qual é o custo mínimo entre esse nó e todos os outros nós do grafo. A cada iteração m o algoritmo determina o caminho mínimo de um nó origem i até um nó k qualquer. Esse algoritmo segue o princípio de uma busca ordenada, portanto os custos dos caminhos tem valores crescentes, por esse motivo que não pode haver arcos com custo negativo. Porém isso não chega a ser um grande problema, pois os custos dos arcos são geralmente grandezas físicas mensuráveis. O algoritmo de Dijkstra é de ordem máxima $O(n^2)$, ou seja, de complexidade quadrática.

O algoritmo de Dijkstra considera um grafo $G = (V, E)$, onde os nós pertencentes a V são divididos em três conjuntos: os já visitados(conjunto fechados), os candidatos(conjunto abertos) e os não-visitados(conjunto desconhecidos).

Seja $D^m_{i,k}$ a soma dos custos dos arcos para de i se chagar a k passando por um

caminho qualquer, e m a m-ésima iteração. Tem-se que $D_{i,k}^{m} = Min\{D_{i,p}^{m-1}, D_{i,p}^{m-1} + C_{p,k}\}$. Onde p é um nó fechado na última iteração. O nó cujo $D_{i,k}^{m}$ foi calculado é colocado no conjunto fechados e seus arcos apontam para os nós que serão incluídos no conjunto abertos. O algoritmo de Dijkstra foi desenvolvido para resolver problemas em rede genéricas.

Os passos principais do algoritmo de Dijkstra, a cada iteração, envolvendo a escolha de um vértice para fechar e a atualização do vetor de distâncias, podem ser implementados por um algoritmo de busca ordenada conhecido na literatura de Inteligência Artificial como algoritmo *Best-First*.

B.1.2 Busca A*

Para tornar mais eficiente a determinação do caminho mínimo entre a origem e um vértice de destino fixo é possível adicionar uma componente heurística. Sua utilização produz um novo algoritmo de busca ordenada: o A*.

Dessa forma, o algoritmo A* se apresenta como a solução ótima mais apropriada ao problema de Busca de Caminhos, pois encontra o caminho de menor custo de um vértice a outro examinando apenas os vizinhos mais promissores do vértice atual da busca.

O algoritmo A* é otimamente eficiente para qualquer função heurística dada, ou seja, nenhum outro algoritmo ótimo tem a garantia de expandir um número de nós menor que ele usando a mesma heurística. Os nós são avaliados de acordo com a equação:

$$f(n) = g(n) + h(n) \tag{B.1}$$

Onde *g(n)* corresponde ao custo exato do caminho desde o nó inicial até o nó n e *h(n)* o custo estimado do caminho de menor custo para ir do nó n até o objetivo. Então pode-se afirmar que *f(n)* é o custo estimado da solução de custo mais baixo passando por n (RUSSEL; NORVING, 2004).

Como em todo processo de busca ordenada tem-se o nó com menor valor $f(n)$ para expandir, (Figura 29), sendo este valor armazenado numa estrutura de nós já pesquisados: a lista de nós fechados; e seus filhos numa estrutura de nós a pesquisar: a lista de nós abertos.

Para preservar a otimalidade do A*, a heurística deve ser admissível. A admissibilidade de uma heurística pode ser comprovada pela propriedade da consistência, sendo a mesma suficiente para a admissibilidade, porém não necessária.

A consistência se baseia no seguinte fato: para um nó n e os seus sucessores n' gerados por uma ação a, o custo estimado de atingir o objetivo a partir de n não é maior que o custo de chegar a n' somado ao custo estimado de n' para o objetivo $h(n) \leq c(n, a, n') + h(n')$ (Figura 30).

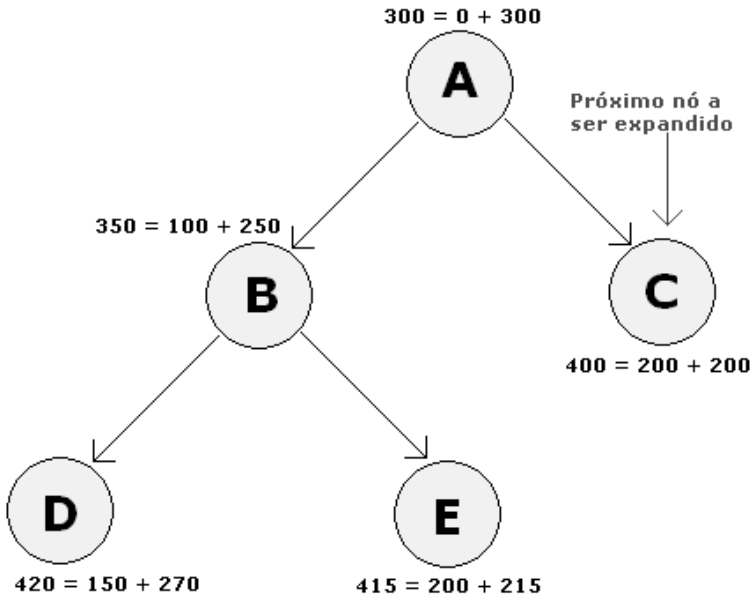

Figura 29 – Esquema de uma expansão de nó em uma busca A*

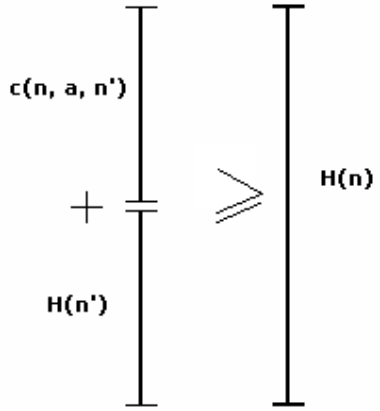

Figura 30 – Condição para uma heurística consistente

Mesmo com todas as vantagens da busca A*, o crescimento exponencial ocorrerá, a menos que o erro na função heurística não cresça com maior rapidez que o logaritmo do custo do caminho real. A condição para crescimento sub exponencial em notação matemática é:

$$|h(n) - h^*(n)| \leq O(log\ h^*(n)) \tag{B.2}$$

Porém, a maioria das heurísticas consistentes, não atende a condição para crescimento subexponencial. Por essa razão, com frequência é impraticável insistir em uma solução ótima (RUSSEL; NORVING, 2004). É possível usar variantes da busca A* que encontrem rapidamente soluções não-ótimas, ou projetar heurísticas mais precisas, embora não estritamente admissíveis.

B.2 Solução *Backward*

A abordagem adotada pelo algoritmo para a solução do problema é do tipo inversa. O algoritmo inicia a partir do vértice objetivo, explorando todo o fecho transitivo inverso do mesmo, até que o vértice de origem seja alcançado.

Inicialmente, tem-se a formulação dual do problema do caminho mínimo, que apresentará uma relação direta com o equacionamento de Bellman e a sua solução por um processo de programação recursiva, estabelecendo uma relação de recorrência característica das técnicas de programação dinâmica e de aprendizado por reforço. A teoria da programação dinâmica e sua relação com o aprendizado por reforço pode ser encontrada no Apêndice A.

B.2.1 Conversão Primal-Dual

O problema do caminho mínimo pode ser analisado sob uma ótica diferente através de sua formulação dual obtida da formulação primal, apresentada na seção anterior. A conversão primal para dual apresenta os seguintes aspectos:

A cada restrição do problema primal relacionado à equação de equilíbrio de um vértice associa-se uma variável dual, denominada μ_i, indexada pelo respectivo vértice. Tem-se então um vetor de variáveis duais com n componentes. Como será visto posteriormente, cada variável dual representará o valor de um caminho do vértice de origem até o vértice associado.

Como o problema primal do caminho mínimo é de minimização, o problema dual será um problema de maximização, sendo o vetor de custos correspondente ao vetor de demanda do problema primal. Convém lembrar que este vetor possui tamanho n.

Como visto no capítulo 2, seção 2.4, equações (2.1) e (2.2), as restrições do problema primal são de igualdade„ fazendo com que o vetor de variáveis duais, conhecido também como vetor de multiplicadores, tenha valores irrestritos. O vetor que representa o lado direito do sistema de restrições do problema dual será correspondente ao vetor de custos do problema primal, tendo portanto tamanho m.

O problema dual apresenta um sistema de m restrições na forma de inequações, considerando que as variáveis associadas do problema primal são limitadas inferiormente.

Considerando o produto da matriz de incidência $A_{n,m}$ pelo vetor de variáveis duais μ pelo lado esquerdo, pode-se descrever a formulação dual do problema de caminho mínimo entre dois vértices fixos x_1 e x_n na forma matricial:

Maximizar $\mu.[1, 0, 0, ..., 0, 0, -1]^T$

Sujeito à:

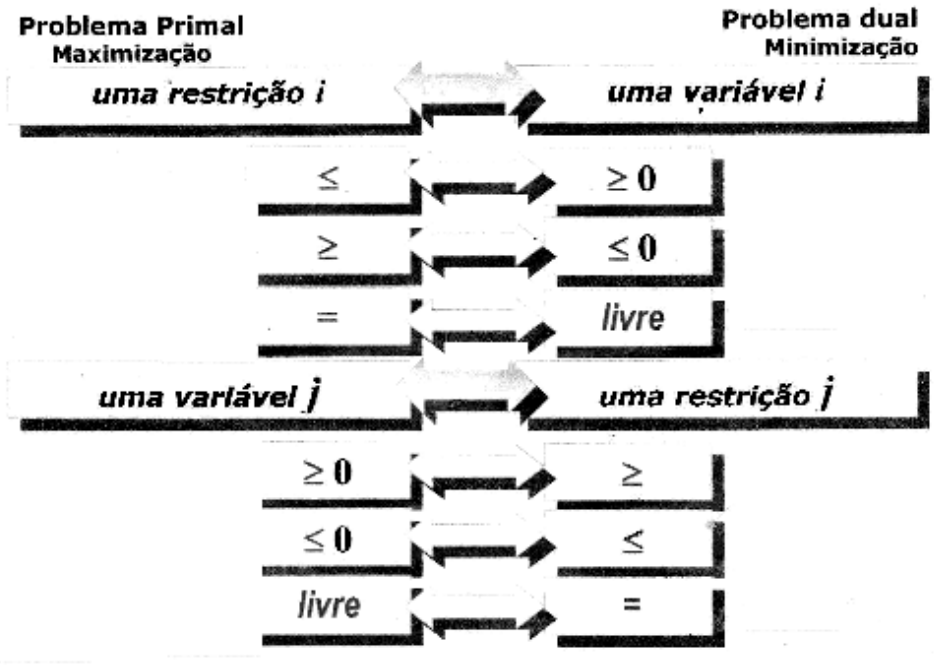

Figura 31 – Relações primal-dual

$\mu.A \leq c$

μ irrestrito e $\mu_1 = 0$

Na forma algébrica, pode-se reescrever a formulação dual como:

Maximizar $\mu_1 - \mu_n$

Sujeito à:

$\mu_k - \mu_j \leq c_{k,j}$, para todo arco (k, j) do grafo.

μ_i irrestrito, para $i = 1, ..., n$

$\mu_1 = 0$

Aplicando a mesma transformação para outros vértices de destino, permanecendo o vértice de origem x_1 fixo, tem-se a seguinte formulação do problema dual para o problema de caminho mínimo entre uma origem fixa e os demais vértices do grafo:

Maximizar $(\mu_1 - \mu_n) + (\mu_1 - \mu_{n-1}) + ... + (\mu_1 - \mu_2)$

Sujeito à:

$\mu_k - \mu_j \leq c_{k,j}$, para todo arco (k, j) do grafo.

μ_i irrestrito, para $i = 1, ..., n$

$\mu_1 = 0$

Invertendo o sinal de μ_i, ou seja, fazendo $\mu_i = -\mu_i$, e, tomando $\mu_1 = 0$, tem-se a

seguinte formulação final para o problema dual:

Maximizar $\mu_n + \mu_{n-1} + ... + \mu_2$

Sujeito à:

$\mu_j - \mu_k \leq c_{k,j}$, para todo arco (k, j) do grafo.

μ_i irrestrito, para $i = 1, ..., n$

$\mu_1 = 0$

Se tanto o primal quanto o dual admitem soluções factíveis, então ambos têm soluções ótimas iguais.

B.2.2 Solução do Problema Dual

O algoritmo proposto para a solução do problema dual é baseado em uma técnica que mantém sempre uma solução primal viável equivalente a uma árvore geradora, e procura a cada iteração satisfazer as restrições do problema dual segundo uma técnica de relaxação. Dessa forma, quando for alcançada uma solução que seja ao mesmo tempo primal viável e dual viável fica garantido a obtenção de uma solução ótima.

B.2.3 Equações de *Bellman*

Princípio da otimalidade de Richard Bellman: uma sequência ótima de decisões tem a propriedade de que quaisquer que sejam o estado e a decisão inicial, as decisões remanescentes constituem uma sequência ótima de decisões com relação ao estado decorrente da primeira decisão, ou seja, toda sub caminho do caminho ótimo é ótima com relação a suas extremidades inicial e final (BELLMAN, 1957).

Seja $C_{k,j}$ o custo do arco (k, j), $P_{i,j}$ o caminho mínimo de um nó de origem i até um nó destino j que passa por algum nó k em uma rede, e $M_{i,j}$ o somatório dos custos dos arcos do caminho $P_{i,j}$. Podemos dizer pelo princípio da otimalidade que para cada $k \neq j$ existe um arco (k, j), tal que o caminho $M_{i,j} = M_{i,k} + C_{k,j}$ é o menor possível, para todas as possibilidades do nó k. Então podemos dizer que $M_{i,j} = Min(M_{i,k} + C_{k,j})$, $k \neq j$. Procura-se então minimizar a função objetivo, que neste caso é o somatório dos custos dos arcos entre uma origem e um destino: $\sum_i^j M_{i,k} + C_{k,j}$ para cada k. **O problema está em determinar o nó** k de maneira eficiente de modo que o caminho seja mínimo.

As equações de Bellman são descritas para grafos que podem apresentar arcos de peso negativo. Se baseiam no princípio da otimalidade de Richard Bellman que norteou o desenvolvimento da programação dinâmica.

Considerando a possibilidade de decompor um caminho mínimo μ_j em um sub caminho, também mínimo, μ_k, seguido de um arco (k, j), pode-se expressar o valor do

caminho μ_j na forma: $\mu_j = \mu_k + c_{k,j}$, para todo vértice x_j diferente do vértice de origem x_1.

Deve-se achar qual vértice x_k deve pertencer à equação acima de modo a definir o valor do caminho μ_j.

Claramente, x_k deve ser escolhido de tal forma que o caminho μ_j seja o menor possível entre todas as possibilidades. Portanto, as equações de *Bellman* podem ser descritas na forma:

$\mu_1 = 0$

$\mu_j = Min_{k \neq j}\{\mu_k + c_{k,j}\}$, $j = 2, ..., n$

Vale ressaltar que a solução desse sistema é equivalente à solução do problema dual. Observando que a obtenção do valor mínimo de um conjunto de valores pode ser obtida por um problema de maximização parametrizado, associado ao fato de que o parâmetro deve ser menor ou igual a cada valor do conjunto, pode-se reescrever o sistema acima na forma:

Maximizar μ_j

Sujeito à:

$\mu_j \leq \mu_k + c_{k,j}$, para todo $k \neq j$

$\mu_1 = 0$

Generalizando a equação de *Bellman* para todo vértice x_j tem-se:

Maximizar $\mu_n + \mu_{n-1} + ... + \mu_2$

Sujeito à:

$\mu_1 = 0$

$\mu_j - \mu_k \leq c_{k,j}$, para todo arco (k, j) do grafo.

Tem-se que a solução das equações de *Bellman* para um grafo sem circuitos pode ser vista como um sistema triangular segundo a abordagem direta. Para um grafo acíclico é possível estabelecer uma enumeração no conjunto de vértices de tal forma que só existirá arco (k, j) se $k < j$. Assim, o sistema de inequações apresenta uma forma triangular superior, podendo ser resolvido diretamente pelo processo de eliminação e substituição de variáveis. Esse processo requer a realização de $(n-1) * n/2$ adições e $(n-1) * (n-2)/2$ comparações, sendo portanto de ordem máxima $O(n^2)$.

De outra maneira, pode-se analisar a solução das equações de *Bellman* por um processo recursivo, segundo a abordagem inversa, implementando um algoritmo baseado na técnica de divisão e conquista. Basicamente, esse algoritmo se divide em duas fases. A primeira é uma fase de decomposição no estilo *top-down*, na qual os sub caminhos são

decompostos até que o sub caminho relacionado ao vértice de origem seja chamado recursivamente. Em seguida, o algoritmo apresenta uma fase de conquista no estilo *bottom-up*, onde os valores dos sub caminhos são conquistados sucessivamente até a determinação do caminho relacionado ao vértice objetivo. Caso o grafo apresente a existência de circuitos positivos a forma de solução do sistema se baseará na utilização de um método de aproximações sucessivas. A descrição do algoritmo em alto nível pode ser feita na forma:

Algoritmo DivisãoConquista;
Caminho(x_k: vértice);
Início
    Se $x_k = x_1$ Então
        $\mu_1 \leftarrow 0$; marcar x_k;
    Senão
        Para todo $v \in L^-[x_k]$ Faça
            Se v não marcado Então
                Caminho(v);
                $\mu_k \leftarrow$ Min $\{\mu_k, \mu_v + c_{v,k}\}$;
            Senão
                $\mu_k \leftarrow$ Min $\{\mu_k, \mu_v + c_{v,k}\}$;
            FimSe;
        FimPara;
        marcar x_k;
    FimSe;
Fim;
Início
    Para $i = 2, ..., n$ Faça
        $\mu_i \leftarrow \infty$;
    FimPara;
    Defina x_1 raiz;
    Defina x_n objetivo;
    Caminho(x_n);
Fim.

B.2.4 Algoritmo de *Bellman-Ford*

Este algoritmo propõe a solução das equações de *Bellman* quando o grafo apresenta circuitos de peso positivo, através de um método de aproximações sucessivas. Dessa forma tem-se, inicialmente, para a primeira iteração:

$$\mu_1^{(1)} = 0$$

$\mu_j^{(1)} = c_{1,j}$, para todo $j \neq 1$

Para as iterações seguintes, computa-se o valor de um caminho μ_j da iteração $m+1$ em função das aproximações obtidas até a iteração m. Portanto:

$\mu_1^{(m+1)} = 0$

$\mu_j^{(m+1)} = \text{Min } \{\mu_j^{(m)}, Min\{\mu_k^{(m)} + c_{k,j}\}\}$

Para cada vértice de x_j é necessário provar que as sucessivas aproximações são monotonicamente decrescentes, ou seja: $\mu_j^{(1)} \geq \mu_j^{(2)} \geq .... \geq \mu_j^{(n-1)}$, a fim de assegurar a convergência do algoritmo.

No passo inicial considera-se a hipótese válida para μ_1, pois seu valor é sempre zero. Na hipótese indutiva para um vértice x_j, $j \neq 1$, considera-se que $\mu_j^{(m)}$ seja o caminho de comprimento mínimo não contendo mais que m arcos.

No passo geral, admitindo que algum caminho mais curto, da origem para o vértice j, contenha mais que m arcos, então, o mesmo conteria $m+1$ arcos, equivalentes à soma de um arco (k, j) ao caminho $\mu_k^{(m)}$, considerando $\mu_k^{(m)}$ um caminho mínimo, pela hipótese indutiva. Dessa forma, minimizando $\mu_k^{(m)} + c_{k,j}$ para todas as escolhas possíveis, tem-se necessariamente: $\mu_j^{(m+1)} \leq \mu_j^{(m)}$.

É importante observar que existem no máximo $n-1$ melhorias para cada caminho do grafo, determinando uma cota superior para o algoritmo de $O(n^3)$.

Para a implementação do algoritmo de *Bellman-Ford*, foi reescrito o método de aproximações sucessivas na forma:

$\mu_1 = 0$

$\mu_j^{(m+1)} = \text{Min } \{\mu_j^{(m)}, (\mu_1^{(m)} + c_{1,j}), ..., (\mu_{k-1}^{(m)} + c_{k-1,j}), (\mu_{k+1}^{(m)} + c_{k+1,j}), ..., (\mu_n^{(m)} + c_{n,j})\}$

Como o processo de convergência é assegurado, o algoritmo pode ser implementado utilizando-se o aninhamento de três *loops* iterativos, na forma:

```
Algoritmo BellmanFord;
Início
    μ1 ← 0;
    μj ← c1,j, para todo j ≠ 1;
    melhora ← VERDADEIRO;
    Enquanto melhora Faça
            melhora ← FALSO;
            Para i ← 2 até n Faça
                Se Atualiza(μi, xi) Então
                    melhora ← VERDADEIRO;
                FimSe;
```

```
                FimPara;
        FimEnquanto;
Final.
Função Atualiza(var μ_i: real; x_i: vértice): lógico;
Início
        Atualiza ← FALSO;
        Para j ∈ L⁻[x_i] Faça
                Se μ_i < μ_j + c_{j,i} Então
                        Atualiza ← VERDADEIRO;
                        μ_i ← μ_j + c_{j,i};
                FimSe;
        FimPara;
Final.
```

APÊNDICE C – Aprendizado por Correção de Erros

No intuito de ajustar os pesos das correções pode ser calculada a diferença entre a saída real gerada pela rede e a saída desejada, fornecida em um aprendizado supervisionado, obtendo assim o erro atual de uma rede neural.

Durante o aprendizado supervisionado, os erros vão sendo calculados sucessivamente, até que cheguem a um valor satisfatório definido a priori, geralmente um valor algumas dezenas de grandeza menor quando comparados com os valores atuantes na diferença. Sendo assim, surge uma curva de erros, a qual está diretamente relacionada à natureza do modelo de neurônio utilizado.

Este processo utiliza algoritmos para caminhar sobre a curva de erros, com o intuito de alcançar um erro menor do que o definido a priori. Devido a essa abordagem muitas vezes, o algoritmo não alcança este mínimo global, atingindo o mínimo local, porém esse fato não é de grande importância se o algoritmo conseguiu alcançar um mínimo local menor que o erro máximo estipulado. Tem-se então que:

$$e_k = sd_k - y_k$$

Sendo e o erro; sd a saída desejada apresentada durante o treinamento; y a saída real da rede; e k o estímulo em questão.

Para a correção do erro, os pesos da rede devem ser ajustados, de forma a aproximar a saída real à desejada. Uma das regras de aprendizado para RNA, bem conhecida é a regra *delta* (WIDROW; HOFF, 1960), também conhecida como $LeastMeanSquare$ (LMS), que minimiza o erro médio quadrático. Esta é apresentada a seguir, e seu ajuste dependerá dos seguintes fatores: do próprio erro calculado, do valor do estímulo de entrada que é transmitido pelo peso a ser ajustado, e também da taxa de aprendizado, a qual relaciona-se à cautela com que a curva de erros é percorrida. Para um dado estímulo k, no passo de treinamento n:

$$\Delta w_{i,j} = \eta.e_k(n).x_j(n)$$

Onde $\Delta w_{i,j}$ é valor de ajuste a ser acrescido ao peso $w_{i,j}$; η é a taxa de aprendizado; $e_k(n)$ é o valor do erro; e $x_j(n)$ é o valor do estímulo.

O valor atualizado do peso será:

$$w(n+1) = w(n) + \Delta w_{i,j}(n)$$

Logo, é possível minimizar a função de erro, também conhecida como função de

custo, utilizando a regra *delta* para corrigir os valores dos pesos:

$$\epsilon(n) = \frac{1}{2}.e^2(n)$$

Onde $\epsilon(n)$ é o erro da rede no passo n do treinamento; e $e(n)$ é o valor da função de custo no passo n do treinamento.

Na regra *delta* generalizada ou algoritmo de retro propagação, a aprendizagem supervisionada ocorre através de exemplos, em tempo discreto e auxiliada por um método de gradiente descendente, para corrigir os erros.

Os pesos sinápticos são ajustados de acordo com o erro quadrático para todos os padrões do conjunto de treinamento. O processo de redução gradativa do erro tende à convergência, onde o erro é estável. A evolução do processo de aprendizagem ocorre, até que algum critério seja satisfeito, como um valor mínimo de erro global ou uma diferença sucessiva mínima entre erros.

APÊNDICE D – Otimização

Aqui é apresentado o ferramental necessário para a solução do problema de maximização, sob o ponto de vista da otimização. Sendo abordado desde a área geral de otimização não-linear, e aos poucos aumentando a especificação, sendo visto a otimização convexa e a programação quadrática.

D.1 Otimização Não-Linear

A Otimização é a área da Programação Matemática que trata de problemas cujo interesse consiste em encontrar pontos de máximo ou de mínimo de funções. Programação ou otimização não-linear é o processo de resolver um sistema de igualdades e desigualdades, juntamente com suas restrições, sobre um conjunto de variáveis reais desconhecido, juntamente com uma função objetivo a ser maximizada ou minimizada, onde algumas das restrições ou a função objetivo é não-linear. Matematicamente o problema pode ser indicado como (BRINKHUIS; TIKHOMIROV, 2005):

$$\max_{x \in X} f(x) \text{ ou } \min_{x \in X} f(x).$$

Onde $f : \mathbb{R}^n \to R$ e $X \subseteq \mathbb{R}^n$.

A seguir será visto a otimização convexa, necessária para o entendimento posterior das formulações da predição de dados estruturados.

D.2 Otimização Convexa

Otimização convexa é um subcampo da otimização matemática e estuda o problema de minimizar ou maximizar funções convexas.

O convexidade de $\mathcal{X}$ e f tornam aplicáveis poderosas ferramentas de análise convexa. O teorema de Hahn-Banach e a teoria de subgradientes conduzem a teoria de condições necessárias e suficientes para otimalidade, a teoria dual generaliza isso para a programação linear e efetivos métodos computacionais (BOYD; VANDENBERGHE, 2004).

A minimização convexa tem aplicações em uma vasta gama de disciplinas, como o controle automático de sistemas, processamento de sinais e desenhos de circuitos eletrônicos.

Uma característica importante da função convexa utilizada aqui é a seguinte: o problema de maximizar uma função convexa pode ser reformulado equivalentemente como

um problema de minimização convexa. E tem-se também a seguinte propriedade das funções convexas: se existe um mínimo local, então ele é o mínimo global.

D.3 Multiplicadores de Lagrange

Achar pontos extremos de funções é muito importante quanto se deseja otimizar algo. Existem muitos métodos, tanto determinísticos como interativos para resolver estes problemas. Um desses importantes métodos é o de Lagrange, ele é específico para problemas que se conhece o domínio de trabalho, ou seja, que tenha suas restrições bem definidas.

Uma maneira para achar os pontos de máximo é igualar a zero todas as derivadas parciais. Se não houvesse vínculos, isto seria o mesmo que impor $df = 0$, onde df, o diferencial da função f, é dado por:

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz$$

Uma vez eliminado z por meio do vínculo, tem-se, em lugar desta última, a equação:

$$dF = \frac{\partial F}{\partial x}dx + \frac{\partial F}{\partial y}dy = 0$$

Ou seja, não aparece mais o diferencial dz, indicando que a função F não depende de z. O método de Lagrange oferece uma técnica mais eficiente e simétrica para eliminar a dependência em z, ou seja, para se livrar do termo em dz na expressão do diferencial da função cujos máximos se procura.

Considere o diferencial da função f:

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz$$

E, como $g(x, y, z) = 0$, tem-se:

$$dg = \frac{\partial g}{\partial x}dx + \frac{\partial g}{\partial y}dy + \frac{\partial g}{\partial z}dz = 0$$

Seja λ um número qualquer, de valor a ser determinado posteriormente. Adiciona-se a df a quantidade λdg, que é zero. Logo:

$$df = df + \lambda dg$$

Portanto, pode-se escrever:

$$df = \left(\frac{\partial f}{\partial x} + \lambda\frac{\partial g}{\partial x}\right)dx + \left(\frac{\partial f}{\partial y} + \lambda\frac{\partial g}{\partial y}\right)dy + \left(\frac{\partial f}{\partial z} + \lambda\frac{\partial g}{\partial z}\right)dz$$

Mas, como λ é indeterminado, pode-se determiná-lo agora impondo que o coeficiente de dz na expressão anterior seja nulo, ou seja:

$$\frac{\partial f}{\partial z} + \lambda \frac{\partial g}{\partial z} = 0$$

Com isso, tem-se agora um df independente de z, e pode-se localizar seus pontos de máximo impondo que $df = 0$, ou, mais precisamente, que $df + \lambda dg = 0$. Mas isso dá as condições:

$$\frac{\partial f}{\partial x} + \lambda \frac{\partial g}{\partial x} = 0$$
$$\frac{\partial f}{\partial y} + \lambda \frac{\partial g}{\partial y} = 0$$

Como, adicionalmente, tem-se a condição dada pela Equação $\frac{\partial f}{\partial z} + \lambda \frac{\partial g}{\partial z} = 0$, nota-se que o conjunto das equações que determinam os pontos de máximo (bem como o valor de λ) é obtido da seguinte maneira: igualem-se a zero as derivadas parciais da função:

$$f + \lambda g$$

A generalização é imediata. Seja $f(x, y, z, u, v)$ a função cujos pontos de máximo deseja-se localizar, e sejam $g(x, y, z, u, v) = 0$ e $h(x, y, z, u, v) = 0$ condições subsidiárias. Então igualam-se a zero as derivadas parciais da função:

$$f + \lambda_1 g + \lambda_2 h$$

onde λ_1 e λ_2 são coeficientes a determinar. Se houver n condições subsidiárias $g_i = 0$, igualem-se a zero as derivadas parciais da função:

$$f + \sum_i \lambda_i g_i$$

Os λ_i são denominados multiplicadores de Lagrange.

Os multiplicadores de Lagrange possuem a seguinte definição. Considere f com suas m restrições g. Sejam elas deriváveis em primeira ordem, contínuas, e que $\nabla g \neq 0$ em qualquer circunstância. Se f tiver um extremo relativo dentro de suas restrições, este ponto ocorre em um ponto $P(x_1^*, x_2^*, ..., x_n^*)$, tal que P pertença a uma superfície de restrição de f na qual os gradientes $\nabla f(x_1^*, x_2^*, ..., x_n^*)$ e $\nabla g(x_1^*, x_2^*, ..., x_n^*)$ são paralelos, ou seja, existe λ tal que a seguinte condição seja satisfeita:

$$\nabla f(x_1^*, x_2^*, ..., x_n^*) = \lambda \nabla g(x_1^*, x_2^*, ..., x_n^*)$$

Apesar de sua solução ser bem simples, para que se ache a sua exata solução nesta forma é necessário que as restrições sejam estritamente na forma de equações, e não

inequações como se apresenta em nosso problema de predição de dados estruturados. Porém pode-se utilizar um método conhecido como Relaxação Lagrangeana nas inequações, neste caso usa-se o subgradiente com os multiplicadores de Lagrange, ao invés do gradiente, para achar uma solução bem aproximada. Tal abordagem é devidamente explicada neste mesmo capítulo nas próximas seções.

D.4 Programação Quadrática

A programação quadrática é um tipo especial de problema de otimização matemática. Representa o problema de otimizar uma função quadrática de diversas variáveis estando sujeita a restrições lineares sobre essas variáveis.

O problema de programação quadrática pode ser formulado como (NOCEDAL; WRIGHT, 2006):

Assuma que $x \in \mathbb{R}^n$. A matriz $Q_{n\times n}$ é simétrica, e c é um vetor $(n \times 1)$.

Minimizar (em relação a x)

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$$

Sujeita a uma ou mais restrições na forma:

$$A\mathbf{x} \leq \mathbf{b} \text{ (restrição de desigualdade)}$$

$$E\mathbf{x} = \mathbf{d} \text{ (restrição de igualdade)}$$

Onde $\mathbf{x}^T$ indica o vetor transposto de $\mathbf{x}$. A notação $Ax \leq b$ significa que todas as entradas do vetor Ax são menores ou iguais que a entrada correspondente do vetor $\mathbf{b}$.

Se Q é uma matriz positiva semi definida, então $f(\mathbf{x})$ é uma função convexa. Tem-se então que a programação quadrática convexa é um caso especial do problema geral de otimização convexa, e tem um mínimo global se existe pelo menos um vetor x que satisfaça as restrições e $f(\mathbf{x})$ está limitada em baixo na região viável. É condição suficiente para ter um ponto $\mathbf{x}$ como um mínimo global a função $f(\mathbf{x})$ ser convexa. Esse mínimo global é único. Se $Q = 0$ então tem-se um problema de programação linear.

O dual de um problema de programação quadrática também é um problema de programação quadrática. Para demonstrar isso considere o caso onde $c = 0$ e Q é positivo definido. Pode-se escrever o Lagrangiano (NOCEDAL; WRIGHT, 2006):

$$L(x, \lambda) = \frac{1}{2}x^T Qx + \lambda^T(Ax - b)$$

Para calcular a função $g(\lambda)$, definida como $g(\lambda) = \inf_x L(x, \lambda)$, calcula-se o ínfimo de L, com $\nabla_x L(x, \lambda) = 0$:

$$x^* = -Q - 1A^T\lambda$$

Portanto, a função dual é:

$$g(\lambda) = -\frac{1}{2}\lambda^T A Q^{-1} A^T \lambda - b^T \lambda$$

Então a função do problema dual de programação quadrática é:

$$\text{Max: } -\frac{1}{2}\lambda^T A Q^{-1} A^T \lambda - b^T \lambda$$
$$\text{Sujeito a: } \lambda \geqslant 0$$

APÊNDICE E – Subdiferenciais e Subgradientes

E.1 Função Convexa

Em matemática, os conceitos de subderivada, subgradiente, e subdiferencial surgem em análise convexa, e estão frequentemente relacionados à otimização convexa.

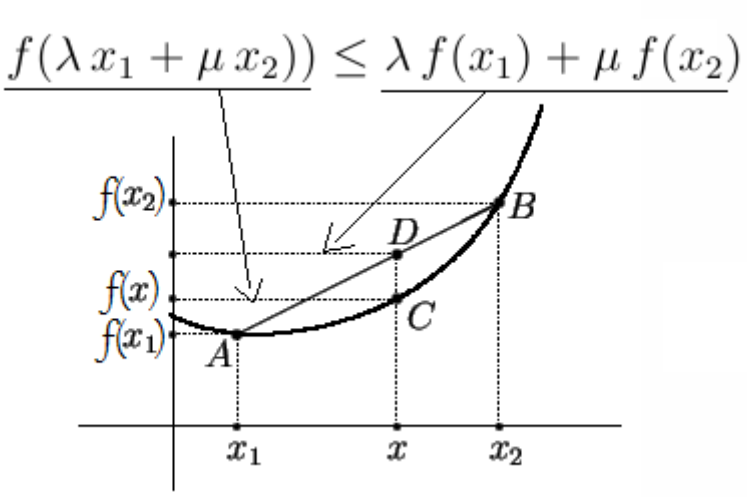

Figura 32 – Gráfico de uma função convexa

Seja $I \subseteq \mathbb{R}$ um intervalo (limitado ou não). Uma função $f : I \longrightarrow \mathbb{R}$ é dita convexa se tiver a seguinte propriedade: Dados dois pontos A e B no gráfico de f, a corda que une estes dois pontos está sempre acima do gráfico de f. Dados $x_1 < x < x_2$ em I, como na (Figura 32), chamando de $\mu = \dfrac{x - x_1}{x_2 - x_1}$, tem-se:

$$0 \leq \mu \leq 1 \quad , \quad x = x_1 + (x - x_1) = x_1 + \mu\,(x_2 - x_1) = (1 - \mu)\,x_1 + \mu\,x_2 \ ,$$

ou ainda, chamando $\lambda = 1 - \mu$,

$$x = \lambda\,x_1 + \mu\,x_2 \quad , \quad \lambda \geq 0 \ , \ \mu \geq 0 \text{ e } \quad \lambda + \mu = 1 \ .$$

Os pontos C e D da Figura têm coordenadas:

$$C = (\lambda\,x_1 + \mu\,x_2, f(\lambda\,x_1 + \mu\,x_2)) \text{ e } \qquad D = (\lambda\,x_1 + \mu\,x_2, \lambda\,f(x_1) + \mu\,f(x_2)).$$

A função f é convexa quando o ponto D está sempre acima de C. Isto se expressa como:

$$\forall x_1, x_2 \in I \quad , \quad \forall \lambda, \mu \geq 0 \text{ com } \quad \lambda + \mu = 1 \quad ,$$

resultando em:

$$f(\lambda\, x_1 + \mu\, x_2)) \leq \lambda\, f(x_1) + \mu\, f(x_2)\ .$$

Uma Figura $\mathcal{A}$ é convexa quando para quaisquer dois pontos A e B de $\mathcal{A}$, o segmento de reta que une A e B está totalmente contido em $\mathcal{A}$. A função f é convexa se e somente se o seu epigráfico, isto é, o conjunto que está acima de seu gráfico é convexo:

$$\mathcal{A} = \{(x, y) \in \mathbb{R}^2 \mid\ x \in I\ ,\ y \geq f(x)\}.$$

Esta é a justificativa para o nome função convexa. Duas propriedades importantes de uma função convexa: se uma função convexa possui um mínimo local, ele também será um mínimo global; o máximo de funções convexas também é uma função convexa (ROCKAFELLAR, 1970).

E.2 Subderivada e Subdiferencial

Uma função continuamente diferenciável de uma variável é convexa num intervalo, se e só se para $f(y) \geq f(x) + f'(x)(y - x)$, para todos x e y no intervalo. Porém, fazendo-se $f : I \to R$, onde I é um intervalo real, ser uma função convexa definida sobre um intervalo aberto na reta dos reais, tem-se que tal função pode não ser necessariamente diferenciável em todos os pontos, como por exemplo, o valor absoluto, $f(x) = |x|$. Entretanto, para qualquer x_0 no domínio da função pode-se traçar uma linha a qual cruza o ponto $(x_0, f(x_0))$ e em qualquer lugar toca ou passa abaixo do gráfico de f (Figura 33). O coeficiente angular desta linha é chamado de **subderivada**, e ao contrário da derivada em um ponto, esta pode ter mais de um valor (URRUTY; LEMARECHAL, 2001).

Rigorosamente, uma **subderivada** de uma função convexa $f : I \to R$ em um ponto x_0 em um intervalo aberto I é um número real c onde $f(x) - f(x_0) \geq c(x - x_0)$, $\forall x \in I$. O conjunto $[a, b]$ de todas subderivadas é chamada de **subdiferencial** da função f em x_0.

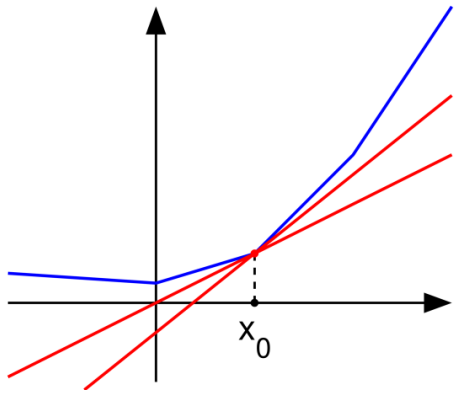

Figura 33 – Subderivadas de uma função convexa

Como exemplo de alguns valores de **subdiferencial** pode-se considerar a função

$f(x) = |x|$ a qual é convexa. Então, um subdiferencial na origem pode ser o intervalo $[-1, 1]$, em qualquer ponto que $x_0 < 0$ é o valor -1, e se $x_0 > 0$ é o valor 1, na origem existem outros, na verdade, infinitos. Já se o subdiferencial tiver um valor único em um ponto, a função é diferenciável neste ponto, ou seja, o subdiferencial é igual a derivada.

E.3 Subgradiente

Os conceitos de **subderivadas** e **subdiferenciais** podem ser generalizados para funções de muitas variáveis. Se $f : U \to R$ é um valor real de uma uma função convexa definida em conjunto aberto convexo no espaço Euclidiano $\mathbb{R}^n$, um vetor g neste espaço, (Figura 34), é chamado de **subgradiente** em um ponto $x^0 \in \mathbb{R}^n$ se $\forall x \in \mathbb{R}^n$:

$$f(x) \geq f(x^0) + g^T(x - x^0); \forall x \in \mathbb{R}^n$$

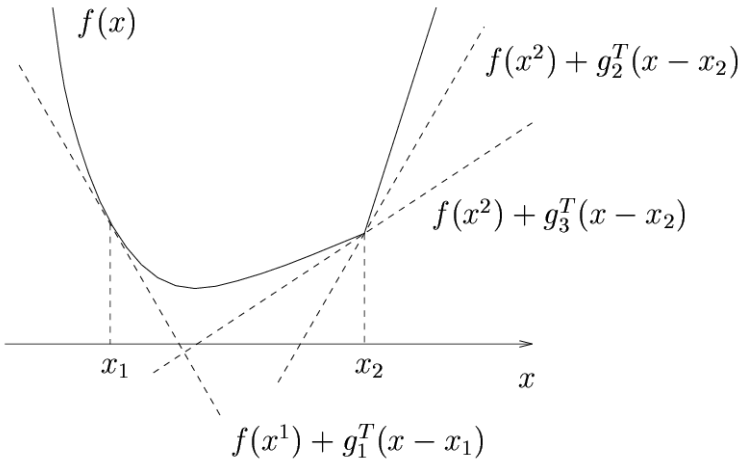

Figura 34 – Exemplos de alguns subgradietes

O conjunto de todos os subgradientes de f em x^0 é chamado de subdiferencial de f em x^0 e é denotado como $\partial f(x^0)$. Na prática, o conhecimento de qualquer elemento de $\partial f(x)$ nos pontos necessários é suficiente para a implementação de inúmeros métodos de otimização.

Os Métodos Subgradientes são pioneiros em otimização não diferenciável. Foram originalmente desenvolvidos por (BERTSEKAS, 1985), na União Soviética, nas décadas de 60 e 70. Esses métodos, também chamados de Métodos Gradientes Generalizados, são uma generalização dos Métodos Gradientes no qual o gradiente da função é substituído por um subgradiente para obter uma nova direção de busca. Possuem uma estrutura muito simples que não utiliza busca linear. O tamanho do passo pode ser fixado ou pode mudar com as iterações, porém dependendo do passo escolhido não se tem a garantia de convergência global (SOUZA, 2008).

Considere o problema de minimizar a função $f(w)$ com $w \in \mathbb{R}$. O processo de atualização básica do subgradiente segundo (BERTSEKAS, 1985) é a seguinte:

Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função convexa com domínio $\mathbb{R}^n$. O método usado na iteração do subgradiente é:

$$w^{(k+1)} = w^{(k)} - \alpha_k d^{(k)}$$

Onde $d^{(k)}$ denota o subgradiente de uma função f em $w^{(k)}$. Se f é diferenciável, o único subgradiente é o próprio gradiente do vetor ∇f. Pode acontecer de $d^{(k)}$ não ser um sentido de descida para f em $w^{(k)}$, ou seja, o sentido oposto ao subgradiente pode não ser de descida, o que é um grande problema. Por isso, (BOYD; L.; MUTAPCIC, 2003) apresenta a proposta de manter uma lista f_{melhor} que controla o menor valor da função objetivo encontrado até agora, mantendo somente os melhores valores da função a cada iteração, ou seja, mantendo o melhor $w^{(k)}$ encontrado:

$$\text{Descida: } f_{melhor}^{(k)} = Min\{f_{melhor}^{(k-1)}, f(w^{(k)})\}$$

É fundamental uma escolha adequada do tamanho do passo $\alpha_k > 0$ em cada iteração. Uma das maiores dificuldades está na escolha do tamanho do passo α_k (NEMHAUSER; WOLSEY, 1999). Se os passos forem muito pequenos, w^k se aproximará muito lentamente do ponto ótimo. Por outro lado, segundo (LIMA, 2007), se forem excessivamente largos, o método poderá oscilar desnecessariamente em torno da solução.

Assim, para garantir a convergência global do método, pode-se escolher o tamanho do passo α_k , chamado de Passo da Série Divergente (RODRIGUES, 1994), de forma que $\lim_{k\to\infty} \alpha_k = 0$ e $\sum_{k=1}^{\infty} \alpha_k = +\infty$, sua demonstração pode ser encontrada em (LEMARECHAL, 1989).

Outra dificuldade dos Métodos Subgradientes é estabelecer um critério de parada uma vez que os subgradientes são encontrados arbitrariamente e por isso, não contem informação sobre a condição de otimalidade. Testes práticos devem ser aplicados observando a especificidade do problema. Um critério de parada para os Métodos Subgradientes pode ser dado pelo número máximo de iterações atingido ou pela condição $w^{k+1} \approx w^k$, ou ainda se $f(w^{k+1}) \approx f(w^k)$ (SOUZA, 2008).

O vetor $d^{(k)}$ é a direção do subgradiente de $f(w)$ para w^k. A cada iteração k é dado um passo do ponto corrente w^k em sentido oposto ao subgradiente.

O algoritmo básico do método do subgradiente funciona da seguinte maneira:

$w = w^0$

$k = 1$

Enquanto critério de parada não for satisfeito faça

Resolver (d^k)

$w^{k+1} = w^k - \alpha_k(d^k)$

$k \leftarrow k + 1$

Fim-enquanto

APÊNDICE F – Problema Inverso

O campo dos problemas inversos foi primeiramente descoberto e apresentado pelo físico soviético-armênio Viktor Ambartsumian. Uma definição abrangente é apresentada no livro de (ENGL; HANKE; NEUBAUER, 1996): "Resolver um problema inverso é determinar causas desconhecidas a partir de efeitos desejados ou observados".

A formulação básica padrão de um problema inverso pode ser dada segundo (ENGL; HANKE; NEUBAUER, 1996):

$$d = G(m), \tag{F.1}$$

onde G é um operador que descreve a relação entre os dados observados d e os parâmetros a serem determinados m.

APÊNDICE G – MMP Boost e SMO Estruturado

Resumidamente, o procedimento associado ao *MMP Boost*, (RATLIFF et al., 2007) e (RATLIFF, 2009), consiste iterativamente em:

1 - Utilizando o vetor atual de características determinar o conjunto de custos. Em seguida obter a matriz atualizada de custos e os melhores caminhos para cada amostra do conjunto de treinamento. Caso esta solução reflita a solução do especialista, então pare, o vetor atual de custos é ótimo. Caso contrário vá para o passo 2.

2 - Considerando todos mapas, forme um conjunto de treinamento para um problema de classificação binária da seguinte forma: Crie um conjunto de exemplos positivos representado pelos vetores de características observados nas células ou estados que possuem interseção entre cada caminho planejado pelo especialista e cada caminho planejado pelo algoritmo associando o rótulo de valor +1. Da mesma forma, crie também um conjunto de exemplos negativos representado pelos vetores de características associados aos estados de cada caminho planejado pelo algoritmo que não possuem interseção com o caminho planejado pelo especialista, associando o rótulo de valor -1.

3 - Com o conjunto de treinamento obtido, treine um classificador que seja capaz de generalizar o valor do rótulo para os demais estados de cada mapa. Considere a criação de uma nova característica expressa por uma combinação ou associação das características atuais. Esta associação pode ser linear ou não-linear dependendo da natureza do classificador. Esta característica reforça a presença (custo) dos estados que refletem a política do especialista e inibe a presença dos demais. O valor desta nova característica para cada célula e para cada mapa estará associado aos valores dos rótulos preditos pelo classificador. Neste caso os valores são binários representando a ocorrência ou não da mesma.

Uma outra abordagem é a obtenção da formulação dual do problema de predição estruturada e a posterior utilização de funções *Kernel* (*trick Kernel*), proposta por (TASKAR et al., 2005), o qual utiliza para solução uma adaptação do método *SMO* (*Sequential Minimal Optmization*), de (PLATT, 1999).

Seja o problema em sua forma primal mais simples:

$$
\begin{aligned}
&Min \frac{1}{2}\|w\|^2 \\
&Sujeito\ a: \\
&\forall y \in Y_i : w^T.f(y_i) \geq w^T.f(y);\ i = 1,\ \ldots,\ m.
\end{aligned}
\tag{G.1}
$$

lembrando que m é a quantidade de pares em $S = \{(x_i, y_i),\ \forall i\}$ (conjunto de treinamento).

Considerando a definição do vetor diferença:

$$\Delta f_i(y) = f_i(y_i) - f_i(y),\ y \in Y_i. \tag{G.2}$$

Tem-se:

$$\begin{aligned} &Min \frac{1}{2}\|w\|^2 \\ &Sujeito\ a: \\ &\forall y \in Y_i : w^T.\Delta f_i(y) \geq 0;\ i = 1,\ \ldots,\ m.. \end{aligned} \tag{G.3}$$

Associando a cada restrição do primal uma variável dual positiva $\varrho_i(y)$, tem-se a seguinte forma final de *Wolfe*, seguindo o mesmo procedimento de relaxação lagrangeana utilizado para a obtenção da forma dual do modelo SVM:

$$\begin{aligned} &Max \sum_{i,y} \varrho_i(y) - \frac{1}{2}.||\sum_{i,y} \varrho_i(y).\Delta f_i(y)||^2 \\ &Sujeito\ a: \\ &\sum_{y} \varrho_i(y) = 0;\ \varrho_i(y) \geq 0,\ \forall y \in Y_i. \end{aligned} \tag{G.4}$$

O vetor final w^* é obtido a partir da expansão em função da solução dual ótima ϱ^*, ou seja:

$$w^* = \sum_{i,y} \varrho_i^*(y).\Delta f_i(y). \tag{G.5}$$

O termo quadrático pode ser reescrito como:

$$||\sum_{i,y} \varrho_i(y).\Delta f_i(y)||^2 = \sum_{i,y}\sum_{j,v} \varrho_i(y).\varrho_j(v).\Delta f_i(y).\Delta f_j(v). \tag{G.6}$$

O produto interno dos vetores diferenças pode ser definido na forma de uma função *Kernel*:

$$K_{i,j}(y, v) = \langle \Delta f_i(y), \Delta f_j(v) \rangle,\ \forall y \in Y_i, \forall v \in Y_j;\ i = 1,\ \ldots,\ m\ e\ j = 1,\ ...,\ m. \tag{G.7}$$

O maior problema relacionado a solução da formulação dual se refere ao grande número de variáveis (restrições da formulação primal) geralmente em número exponencial. Para evitar este problema emprega-se um algoritmo de planos de cortes, (TSOCHANTARIDIS et al., 2005), a exemplo da técnica de geração de restrições empregada para a solução da formulação primal.