Análise Automática de Desempenho
de Classificadores e Técnicas de Classificação
na Plataforma R

Ney Barros Luz

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração: Ciência da Computação

Orientadora: Profa. Dra. Nina S. T. Hirata

O autor contou com o apoio da Câmara Legislativa
do Distrito Federal para o desenvolvimento deste trabalho

São Paulo

2005

Análise Automática de Desempenho de Classificadores e Técnicas de Classificação na Plataforma R

Este exemplar corresponde à redação da dissertação
de mestrado a ser apresentada por Ney Barros Luz
à Comissão Julgadora.

Resumo

Tecnologias relacionadas a classificação possuem grande potencial de utilização prática. Notadamente, a geração de classificadores, a análise de seus desempenhos e a seleção do melhor classificador dentre os que foram gerados é feita muitas vezes de maneira não automática. Consequentemente, dado um problema de classificação, é de grande importância e interesse uma solução computacional que ofereça diversas técnicas de geração de classificadores e critérios para análise de desempenho, selecionando automaticamente os melhores classificadores. Ao oferecer várias técnicas de geração de classificadores, essa solução pode ser útil na comparação de diversas abordagens. Além disso, para um especialista em projeto de classificadores, tal solução possibilita sua concentração nas características que sejam mais relevantes e dependentes de contexto como, por exemplo, entendimento e modelagem do problema. Para os não especialistas interessados ou envolvidos num processo de classificação, a solução pode compor uma camada de abstração dos aspectos de geração de classificadores e análise de desempenho. Dentre as diversas técnicas para geração de classificadores existentes, algumas são baseadas em indução a partir de exemplos. Tomando como ponto de partida três dessas técnicas e uma abordagem estatística para análise de desempenho, investigou-se de forma geral alguns de seus princípios teóricos e práticos. Com base nessa investigação, estudou-se a plataforma R para viabilizar a solução computacional desejada. O fruto desse estudo e dessa investigação é um arcabouço contendo técnicas de geração de classificadores e seleção automática dos melhores classificadores por análise de desempenho. Experimentos utilizando o ambiente R, técnicas nele implementadas e um pacote especialmente desenvolvido ilustram a viabilidade da solução proposta.

Sumário

1	Introdução	1
1.1	Motivação	1
1.2	O problema de classificação	4
1.3	Objetivos	5
1.4	Contribuições	6
1.5	Organização do texto	7
2	Conceitos básicos	8
2.1	Conceitos relacionados à representação	8
2.2	Processo	9
3	Geração e análise de desempenho de classificadores	21
3.1	Vizinhos mais próximos	21
3.1.1	Exemplo	22
3.1.2	Variações da técnica	25
3.2	Árvores de decisão	27
3.2.1	Exemplo	31
3.2.2	Variações da técnica	42
3.3	Redes Neurais Artificiais	43
3.3.1	Variações da técnica	49
3.4	Abordagem para análise de desempenho	50
3.4.1	Análise de desempenho	50
3.4.2	Estratégias para partição dos dados	53
4	Aspectos de implementação	63

4.1	A plataforma R	63
4.2	O pacote class	67
4.3	O pacote rpart	70
4.4	O pacote nnet	75
4.5	O pacote linnaeus	78
5	Experimentos e Resultados	83
5.1	Classificadores	83
5.2	Experimento 1	90
5.3	Experimento 2	92
5.4	Experimento 3	94
6	Conclusão	96
6.1	Um apanhado geral	96
6.2	Considerações finais	96
A	Exemplos de dados dos experimentos	98

Lista de Figuras

2.1	Projeto de Classificador	13
2.2	Validação do Classificador	15
2.3	Projeto e Validação de Classificador	16
2.4	Classes Relacionadas a Projeto de Classificadores	19
2.5	Classificador Modelado por Orientação à Objetos	20
3.1	Exemplo de Árvore - Título V da Constituição Brasileira	27
3.2	Função Logarítmica Binária	29
3.3	Entropia para uma Variável Aleatória Discreta X com 2 valores distintos	30
3.4	Construção de uma Árvore de Decisão (passo 1)	35
3.5	Construção de uma Árvore de Decisão (passo 2)	38
3.6	Construção de uma Árvore de Decisão (passo 3)	41
3.7	Exemplo de Neurônio Artificial	43
3.8	Neurônio Artificial com Valor de Limiar θ	55
3.9	Função Logística	55
3.10	Função Logística ($T = 2$)	56
3.11	Função Logística ($T = 1.5$)	56
3.12	Função Logística ($T = 1$)	57
3.13	Função Logística ($T = 1/2$)	57
3.14	Função de Ativação Linear	58
3.15	Função de Ativação Degrau	58
3.16	Função de Ativação Rampa	59
3.17	Função de Ativação Tangente Hiperbólica	59
3.18	Função de Siebert para $k = 8$, na faixa $0 \leq r \leq 1$, com $\alpha = 1/2$	60
3.19	Exemplo de Rede Neural Artificial	60
3.20	Mínimo Global e Mínimo Local de uma Função	61
3.21	Máximo da função $p(1 - p) = 1/4$ quando $0 \leq p \leq 1$	62
4.1	Pacotes Padrões de uma Distribuição R	65

4.2	Pacotes Recomendados da CRAN	66
4.3	Funcionalidades de Vizinhos Mais Próximos do Pacote <i>class</i>	67
4.4	Funcionalidades de Árvores de Decisão do Pacote <i>rpart</i>	71
4.5	Funcionalidades de Redes Neurais do Pacote <i>nnet</i>	75
4.6	Funcionalidades do Pacote <i>linnaeus</i>	79
A.1	Experimento 2 - Plotagens e Histogramas dos Atributos de <i>íris</i>	100
A.2	Experimento 2 - Plotagens e Histogramas dos Atributos de <i>íris</i> (cont.)	101
A.3	Experimento 2 - Diagrama em Caixas dos Atributos de <i>íris</i>	102
A.4	Experimento 2 - Atributos da Base de Dados <i>íris</i>	103
A.5	Experimento 3 - Plotagens e Histogramas dos Atributos	104
A.6	Experimento 3 - Plotagens e Histogramas dos Atributos (cont.)	106
A.7	Experimento 3 - Diagrama em Caixas dos Atributos	107
A.8	Experimento 3 - Relação entre os Atributos	108
A.9	Experimento 1 - Diagrama em Caixas de VMP versus RNA sobre <i>synth</i>	109
A.10	Experimento 2 - Diagrama em Caixas de VMP versus RNA sobre <i>íris</i>	110

Lista de Tabelas

1.1	Proposições Legislativas, Ementas e Tipos de Proposições	4
1.2	Ementa de uma Proposição Legislativa a ser Classificada	5
2.1	Descrição de Estados Brasileiros	11
2.2	Abordagens para Geração de Classificadores (adaptado de [JDM00])	12
2.3	Técnicas de Extração e Projeção de Atributos (adaptado de [JDM00])	17
2.4	Técnicas de Seleção de Atributos (adaptado de [JDM00])	18
3.1	Parlamentos e Portes Institucionais	22
3.2	Parlamentos, Distâncias e Portes Institucionais	24
3.3	Aplicação da Técnica Vizinhos Mais Próximos para $k = 3$	25
3.4	Proposições Legislativas ($\mathcal{T}$)	32
3.5	Proposições Legislativas ($\mathcal{T}'$)	35
3.6	Proposições Legislativas ($\mathcal{T}''$)	37
3.7	Algoritmo Backpropagation (adaptado de [Mit97])	49
4.1	Liberdades de um Software Livre	63
4.2	Descrição das Funções de Vizinho Mais Próximo do Pacote <i>class</i>	68
4.3	Descrição das Funções de Vizinho Mais Próximo do Pacote <i>class</i> (cont.)	69
4.4	Descrição das Funções de Vizinho Mais Próximo do Pacote <i>class</i> (cont.)	70
4.5	Descrição das Funções de Árvores de Decisão do Pacote <i>rpart</i>	72
4.6	Descrição das Funções de Árvores de Decisão do Pacote <i>rpart</i> (cont.)	73
4.7	Descrição das Funções de Árvores de Decisão do Pacote <i>rpart</i> (cont.)	74
4.8	Descrição das Funções de Redes Neurais do Pacote <i>nnet</i>	76
4.9	Descrição das Funções de Redes Neurais do Pacote <i>nnet</i> (cont.)	77
4.10	Descrição das Funções do Pacote <i>linnaeus</i>	80
4.11	Descrição das Funções do Pacote <i>linnaeus</i> (cont.)	81
4.12	Descrição das Funções do Pacote <i>linnaeus</i> (cont.)	82

5.1	Especificação dos Parâmetros para VMP_1	83
5.2	Especificação dos Parâmetros para VMP_2	84
5.3	Especificação dos Parâmetros para VMP_3	84
5.4	Especificação dos Parâmetros para VMP_4	84
5.5	Especificação dos Parâmetros para VMP_5	84
5.6	Especificação dos Parâmetros para VMP_6	84
5.7	Especificação dos Parâmetros para VMP_7	85
5.8	Descrição da Rede Neural Artificial RNA_1	85
5.9	Descrição da Rede Neural Artificial RNA_2	85
5.10	Descrição da Rede Neural Artificial RNA_3	86
5.11	Descrição da Rede Neural Artificial RNA_4	86
5.12	Descrição da Rede Neural Artificial RNA_5	86
5.13	Descrição da Rede Neural Artificial RNA_6	87
5.14	Descrição da Rede Neural Artificial RNA_7	87
5.15	Descrição da Rede Neural Artificial RNA_8	87
5.16	Descrição da Rede Neural Artificial RNA_9	88
5.17	Descrição da Rede Neural Artificial RNA_{10}	88
5.18	Parâmetros para a Árvore de Decisão AD_1	89
5.19	Parâmetros para a Árvore de Decisão AD_2	89
5.20	Experimento 1 - Descrição da Base de Dados <i>synth</i>	90
5.21	Experimento 1 - Análise de Desempenho de Classificadores para <i>synth</i>	91
5.22	Experimento 2 - Descrição da Base de Dados <i>iris</i>	92
5.23	Experimento 2 - Análise de Desempenho de Classificadores para <i>iris</i>	93
5.24	Experimento 3 - Descrição da Base de Dados <i>imunopediatria</i>	94
5.25	Experimento 3 - Análise de Desempenho de Classificadores para <i>imunopediatria</i>	95
A.1	Experimento 1 - Alguns Exemplos da Base de Dados <i>synth</i>	98
A.2	Experimento 2 - Alguns Exemplos da Base de Dados <i>iris</i>	99
A.3	Experimento 3 - Alguns Exemplos da Base de Dados <i>imunopediatria</i>	105

Notação Utilizada

$\mathbb{N}$	Conjunto dos números naturais
$\mathbb{Z}$	Conjunto dos números inteiros
$\mathbb{R}$	Conjunto dos números reais
$A_i, i = 1, 2, \dots, n$	Atributo, característica, <i>feature</i> , variável, medida, conjunto, componente
$a_i \in A_i, i = 1, 2, \dots, n$	Valor de atributo, elemento
$A_1 \times A_2 \times \dots \times A_n$	Espaço de atributos, produto cartesiano
n	Dimensão do espaço de atributos
$[A_1, A_2, \dots, A_n]$	Esquema de atributos, tipo da instância, n -upla ordenada, sequência
$\vec{x} = [a_1, a_2, \dots, a_n]$	Instância, exemplo, vetor, vetor (de valores) de atributos
k	Número de categorias
$\{C_1, C_2, \dots, C_k\}$	Categorias, classes
$\{1, 2, \dots, k\}$	Categorias, classes mapeadas como um subconjunto finito dos inteiros positivos
Σ	Alfabeto
Σ^*	Conjunto de todos os exemplos possíveis sobre Σ
X	Espaço de exemplos
C	Conceito
f	Função alvo para um conceito
$\vec{w}$	Vetor de pesos
η	Taxa de aprendizado
$\mathcal{T}$	Amostra
$\mathcal{T}_a$	Amostra de aprendizado, amostra de treinamento, conjunto de treinamento
$\mathcal{T}_v$	Amostra de validação, amostra de teste, conjunto de teste, conjunto de validação
$m = \mathcal{T} $	Tamanho de uma amostra
$\mathcal{H}$	Espaço de hipóteses

Notação Utilizada (cont.)

$\hat{f}$	Hipótese de uma função alvo, classificador, agente computacional, estimador, modelo, regra
A	Algoritmo de aprendizado, operador, esquema de aprendizado
g	Função de ativação
VMP	Hipótese gerada por vizinhos mais próximos
RNA	Hipótese gerada por rede neural artificial
AD	Hipótese gerada por árvore de decisão
$erro_T(\hat{f})$	Erro observado de uma hipótese, estimador do erro verdadeiro
$erro_D(\hat{f})$	Erro verdadeiro de uma hipótese
Z_n	Nível de significância
μ	Média populacional
σ	desvio padrão populacional
DP_o	Desvio padrão otimista
IC_o	Intervalo de confiança otimista
DP_c	Desvio padrão conservativo
IC_c	Intervalo de confiança conservativo
δ	Função de Kronecker
$X \sim b(n, p)$	X segue o modelo de distribuição binomial com parâmetros n e p
$X \sim N(\mu, \sigma^2)$	X segue o modelo de distribuição normal com parâmetros μ e σ^2
$P(X = x_i) = p(x_i) = p_i$	Função de probabilidade
$F(x) = P(X \leq x)$	Função de distribuição

Capítulo 1

Introdução

Considerando que dado algo a ser classificado:

1. em geral, é possível projetar agentes computacionais capazes de fazer a classificação;
2. há diversas maneiras para projetá-los;
3. é possível avaliar quão bom é cada agente.

A partir dessas considerações, perguntas talvez naturais sejam: como projetar os agentes computacionais? Como avaliar, sob um ponto de vista formal, esses agentes? Como selecionar o melhor?

Respostas à estas perguntas são o assunto deste trabalho. O tema tratado é classificação. Específico a esse tema, é realizado um estudo sobre geração, análise de desempenho e seleção automática de classificadores.

1.1 Motivação

Tecnologias relacionadas à classificação possuem grande potencial de utilização prática. Por exemplo:

- na medicina, a partir de dados clínicos e laboratoriais, pode-se por meio de classificação diagnosticar prováveis imunodeficiências, auxiliando o trabalho médico;
- no processamento de imagens, a partir de imagens obtidas por satélites, pode-se classificar áreas de desmatamento em meio às florestas;

- na bioinformática, a classificação pode ser utilizada para analisar seqüências de DNA ou expressões gênicas;
- na engenharia de software, a partir de componentes desenvolvidos, testados e classificados, pode-se aumentar a produtividade de construção de software;
- na política, a partir de análises do processo legislativo, pode-se detectar padrões que sugiram pontos de melhoria e aumento da qualidade da atuação parlamentar.

Um processo de classificação envolve diversos aspectos. Dentre eles, um é a geração automática de classificadores. No contexto de imagens, esse aspecto é conhecido por projeto automático ou programação automática de operadores. Um grupo de cientistas e estudantes que atua na área de processamento de imagens [Vis] tem trabalhado com ele ao longo dos últimos anos. Um dos objetivos do grupo é desenvolver uma metodologia para projeto automático de operadores baseada em técnicas de indução a partir de exemplos [Tom96, BDT97, BTHH00, Has00, Hir00, Jr01, Bru03]. A metodologia tem como fundamentação conceitual diversas teorias, como Morfologia Matemática [Ser82, BB91, Hei94], Teoria das Probabilidades [Mey83, Dan00, ML04], Álgebra Booliana [PY73, Men77, Dag95] e Aprendizado Computacional [AB92, Hau92], dentre outras. Ela insere o problema de programação automática dentro do contexto de aprendizado computacional e oferece o ISI (*Incremental Splitting of Intervals*), um algoritmo que pode ser utilizado tanto para treinamento de operadores quanto para minimização de funções booleanas. Tomita elaborou o algoritmo [Tom96] e, ao analisar diversos trabalhos, observou que algumas técnicas para automatizar o projeto de operadores eram dependentes de contexto enquanto outras eram independentes. Além disso, notou que as técnicas dependentes de contexto exploravam ou levavam em consideração na sua formulação características específicas do problema a ser resolvido, enquanto as independentes de contexto não. Percebeu também que cada técnica apresentava diferente grau de restrição em função do grau de dependência de contexto e do tipo de restrição imposta sobre o espaço de classificadores [Hir00]. Depois dessas observações e a partir de estudos de problemas de outras áreas de conhecimento, como a biologia computacional, ela reforçou a constatação de que o processo de projetar automaticamente operadores de imagens poderia ser generalizado para um processo de projetar automaticamente classificadores de dados. Assim, a metodologia para projeto automático poderia ser generalizada como uma metodologia para geração de classificadores para análise de dados em geral [CM98, Vap98]. Um exemplo de aplicação dos classificadores gerados por essa metodologia seria sobre dados gênicos, cuja análise vem demandando especial atenção [AFV94, BO98].

Adicionalmente ao ISI, outros algoritmos para geração automática de classificadores foram elaborados. Esse fato e as constatações anteriores sugerem a construção de um sistema computacional para a análise de diversas categorias de dados que possa oferecer a implementação de diversos algoritmos. Nesse caso, o sistema separaria os procedimentos independentes de contexto dos dependentes do domínio do problema. Os aspectos independentes de contexto

passíveis de automação podem ser automatizados. Além disso, ao oferecer várias técnicas de geração de classificadores, o sistema pode ser útil na comparação das diversas abordagens. Há várias abordagens para gerar classificadores, daí surgindo uma questão: dado um problema de classificação, qual a abordagem mais adequada para a geração do classificador? O sistema facilitaria a escolha ao indicar a abordagem mais apropriada ou mesmo mesclar abordagens. Nessa linha, o sistema pode também ser um facilitador nas pesquisas relacionadas a combinação de classificadores [AGK98, Alp98, Die00, KHD98, KHW97].

A construção do sistema computacional requer algumas considerações. Para tratar dos requisitos de dependência e independência de contexto, a arquitetura do sistema deve oferecer facilidade de expansão e abrangência. O sistema deve oferecer ainda características importantes e desejáveis como facilidade de uso, eficiência, precisão e robustez.

Um sistema computacional assim traria algumas vantagens. Para um especialista em projeto de classificadores, possibilitaria que este se concentrasse nas características mais relevantes do problema ou que fossem dependentes do contexto, como por exemplo, na aquisição de características relevantes ou filtragem de dados. Para os não especialistas interessados ou envolvidos num processo de classificação, o sistema seria uma camada de abstração do aspecto de geração automática de classificadores.

Na seção a seguir é introduzido o problema de classificação.

1.2 O problema de classificação

A tabela 1.1 apresenta proposições legislativas, suas respectivas ementas e classificações quanto ao tipo de proposição. Dada a ementa apresentada na tabela 1.2 na próxima página, qual o tipo de sua proposição?

Proposição Legislativa	Ementa de Proposição Legislativa	Tipo de Proposição Legislativa
P1	Dispõe sobre a Lei Orgânica do Tribunal de Contas do Distrito Federal e dá outras providências.	Projeto de Lei Complementar
P2	Dispõe sobre o Programa de Melhoria do Atendimento ao Contribuinte, e dá outras providências.	Projeto de Lei Ordinária
P3	Concede o Título de Cidadão Honorário de Brasília ao atleta olímpico Vanderlei Cordeiro de Lima.	Projeto de Decreto Legislativo
P4	Cria o Parlamento Jovem Brasiliense, no âmbito da Câmara Legislativa do Distrito Federal.	Projeto de Resolução
P5	Dá nova redação ao art. 96, §2o., da Lei Orgânica do Distrito Federal.	Proposta de Emenda à Lei Orgânica
P6	Dispõe sobre a Campanha de Prevenção de Acidentes Domésticos no Distrito Federal e dá outras providências.	Projeto de Lei Ordinária
P7	Concede o Título de Cidadão Benemérito de Brasília à carateca Lucélia de Carvalho Ribeiro.	Projeto de Decreto Legislativo

Tabela 1.1: Proposições Legislativas, Ementas e Tipos de Proposições

De uma maneira geral, o problema de classificação pode ser compreendido a partir de cinco elementos: um conjunto de objetos, um conjunto de características, um conjunto de classes¹, um classificador e um conjunto de critérios de classificação.

Na situação descrita, o conjunto de objetos é o conjunto de proposições legislativas. O conjunto de atributos é aquele que caracteriza o objeto, nesse caso, com dois elementos:

¹que serão denominadas *categorias* para não haver confusão com o conceito de classes de orientação à objetos

*Altera a denominação e a estrutura organizacional
da Secretaria de Estado de Desenvolvimento
Tecnológico do Distrito Federal.*

Tabela 1.2: Ementa de uma Proposição Legislativa a ser Classificada

identificador de proposição e ementa. O conjunto {Projeto de Lei Complementar, Projeto de Lei, Projeto de Decreto Legislativo, Projeto de Resolução, Proposta de Emenda à Lei Orgânica} de tipos de proposições constitui o conjunto de categorias. O elemento classificador é qualquer pessoa que observe a ementa e classifique a proposição em uma das categorias.

E quanto aos critérios de classificação?

Em geral, estes nem sempre estão explícitos ou são bem conhecidos. Na situação dada, pode ser razoável supor que a pessoa que fará a classificação não conheça os critérios que definem os diversos tipos de proposição. Nesse caso, como ela poderia proceder?

Uma heurística poderia utilizar as proposições classificadas da tabela 1.1 na página anterior e consistir em fazer uma suposição sobre os critérios de classificação a partir de semelhanças entre os vocábulos das ementas apresentadas e a ementa da proposição com classificação desconhecida. A opção por essa heurística insere o problema de classificação num contexto de aprendizado denominado *aprendizado supervisionado*, onde hipóteses são feitas a partir de objetos com classificação já conhecida.

Nesse texto, trata-se o problema de classificação sob o ponto de vista do classificador e, em particular, sua geração e análise de desempenho. Os classificadores gerados são agentes computacionais que, assim como uma pessoa na situação descrita, fará a classificação de objetos a partir de heurísticas que se utilizam de exemplos já classificados. Isto é, o classificador será um agente computacional que, dado um conjunto de objetos descritos por um conjunto de atributos, um conjunto de categorias e uma hipótese dos critérios de classificação, classificará os objetos nas categorias por aprendizado supervisionado.

1.3 Objetivos

O objetivo deste trabalho é a seleção automática de classificadores por análise de desempenho. Para cumprir esse objetivo, metas intermediárias tornaram-se necessárias, particularmente o estudo de técnicas de geração e de uma abordagem formal para análise de desempenho de classificadores.

Especificamente no que diz respeito às técnicas de geração de classificadores, o estudo se concentra em três técnicas de aprendizado supervisionado. As técnicas de geração de classificadores escolhidas foram: vizinhos mais próximos, árvores de decisão e redes neurais

artificiais. Embora existam outras técnicas de geração de classificadores, estas não foram abordadas. Todavia, variantes de cada uma das técnicas escolhidas foram consideradas.

A abordagem para análise de desempenho de classificadores e de técnicas de geração de classificadores tem como base fundamentos teóricos de probabilidade e estatística. Nela, os erros dos classificadores são vistos como variáveis aleatórias cuja distribuição segue o modelo binomial.

Com vistas à construção de uma solução computacional para geração e análise de desempenho de classificadores, adotou-se um paradigma de reutilização. Nessa linha, traçou-se também como meta o estudo de uma plataforma de software livre que servisse como parte dessa solução. R [R, R D03b, R D03d, R D03a, R D03e, R D03f, R D03c] foi a opção escolhida.

1.4 Contribuições

Neste trabalho procurou-se contribuir da seguinte forma:

- descrevendo conceitos e técnicas de geração de classificadores;
- descrevendo uma abordagem para análise de desempenho de classificadores;
- analisando a plataforma R como base computacional para implementação de diversas técnicas de classificação, incluindo geração e análise de desempenho. Em particular, foi feita uma catalogação dos pacotes existentes voltados para classificação;
- implementando o pacote *linnaeus*, que contribui com funções de análise de desempenho de classificadores, além de oferecer bases de dados para testes, nos moldes de outros pacotes da plataforma R;
- contribuindo com a aplicação de elementos de linguagem gráfica voltados para engenharia de software. Estes elementos, assim como gráficos matemáticos, visam a facilitar o entendimento e visualização de conceitos. Não se encontrou a utilização desses elementos na literatura da plataforma R;
- comparando empiricamente classificadores;
- contextualizando conceitos de classificação no âmbito do poder legislativo brasileiro. Como parece não existir trabalhos com esse enfoque, o estudo apresentado é uma contribuição nesse sentido.

1.5 Organização do texto

O capítulo 2 apresenta conceitos básicos, em particular os voltados à representação de um objeto bem como um processo para geração e análise de desempenho de classificadores.

Uma apresentação das técnicas escolhidas para geração de classificadores é feita no capítulo 3, envolvendo as técnicas de vizinhos mais próximos, árvores de decisão e redes neurais artificiais. Nesse capítulo também é descrita uma abordagem para análise de desempenho de classificadores. Conceitos de erro observado e erro verdadeiro de um classificador são apresentados e modelados sob enfoque estatístico clássico.

Já o capítulo 4 apresenta aspectos de implementação, com enfoque na plataforma R e em seus pacotes relacionados à classificação. O pacote *linnaeus* é descrito nesse capítulo.

Experimentos de geração de classificadores e análise de desempenho utilizando os conceitos teóricos dos capítulos 2 e 3 e a solução computacional do capítulo 4 são o assunto do capítulo 5.

O capítulo 6 apresenta as conclusões do trabalho. Nele há um apanhado geral e a descrição de possíveis trabalhos futuros que podem ser desenvolvidos como continuação dessa pesquisa.

Por fim, além do texto principal, há um apêndice contendo alguns dados e informações utilizadas nos experimentos.

Capítulo 2

Conceitos básicos

O propósito deste capítulo é descrever conceitos subjacentes a esse trabalho e apresentar um processo para geração e análise de desempenho de classificadores. São apresentados alguns conceitos básicos, em particular os voltados à representação.

2.1 Conceitos relacionados à representação

Seja $n \in \mathbb{N}$. Sejam n conjuntos, $A_1, A_2, \dots, A_n$. Seja $\vec{x}$ um elemento do produto cartesiano $A_1 \times A_2 \times \dots \times A_n$. Isto é, $\vec{x} = [a_1, a_2, \dots, a_n]$, $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$ é uma n -upla ordenada. No contexto deste trabalho:

- cada conjunto A_i , $i = 1, 2, \dots, n$, será denominado **atributo**;
- cada elemento $a_i \in A_i$, $i = 1, 2, \dots, n$, será denominado **valor do atributo A_i** ;
- $A_1 \times A_2 \times \dots \times A_n$ será denominado **espaço de atributos**;
- $(A_1, A_2, \dots, A_n)$, será denominado **esquema de atributos**;
- $\vec{x} = [a_1, a_2, \dots, a_n]$ será denominada **instância**.

Exemplificando, seja $n = 4$ e os seguintes conjuntos: A_1 o conjunto sexo biológico, {feminino, masculino}; A_2 o conjunto idade (em anos), um subconjunto de $\mathbb{N}$; A_3 o conjunto altura (em metros), um subconjunto de $\mathbb{R}$; A_4 o conjunto peso (em quilos), um subconjunto de $\mathbb{R}$.

Tem-se que:

- sexo biológico, idade, altura e peso são atributos;
- feminino é um valor do atributo sexo biológico;
- o espaço de atributos é $\text{sexo biológico} \times \text{idade} \times \text{altura} \times \text{peso}$;
- $(\text{sexo biológico}, \text{idade}, \text{altura}, \text{peso})$ é o esquema de atributos;
- $\vec{x} = [\text{masculino}, 21, 1.70, 65.5]$ é uma instância.

O conjunto de todos os valores possíveis de um atributo é denominado *domínio* do atributo.

Uma instância é uma n -upla ordenada de valores de atributos. Sua concepção conceitual é chamada esquema de atributos.

Pode-se fazer uma analogia com os conceitos em linguagens de programação de tipos de dados, variáveis e valores das variáveis.

Um atributo corresponde à noção de um tipo de dados, como por exemplo, inteiro ou real. O conceito de um *esquema de atributos* corresponde à noção de um tipo de dados composto, definido pelo usuário. Cada valor particular corresponde a um valor de atributo.

Uma variável do tipo de dados composto possui valores particulares num certo instante de tempo. Cada um desses valores particulares corresponde a um valor de atributo.

O conceito de uma variável de tipo de dados composto numa linguagem de programação corresponde ao conceito de *instância*.

Valores de atributos de instâncias são diferentes para instâncias diferentes. Dessa forma, cada atributo A_i pode ser visto com uma variável aleatória, denotada por A_i para distinguir do atributo enquanto conjunto.

Cada instância definida como um ponto no espaço de atributos, pode também ser vista como um vetor em relação a um sistema de coordenadas cartesianas, tendo seu ponto inicial em algum ponto de referência fixo, normalmente a origem. Uma vez que os valores de seus atributos são vistos como variáveis aleatórias, uma instância $\vec{x}$ passa a ser entendida como um vetor aleatório, denotada por $\mathbf{X}$ para distinguir de sua natureza enquanto um ponto no espaço de atributos. Assim, cada instância é expressa por um vetor num espaço n -dimensional e diversas instâncias formam uma distribuição de $\mathbf{X}$ num espaço n -dimensional.

2.2 Processo

Para gerar um classificador e analisar seu desempenho, torna-se necessário um processo que contemple pelo menos os seguintes aspectos: definição dos valores possíveis da função alvo; representação de uma instância; levantamento dos exemplos ou sua captação a partir

do ambiente; seleção de exemplos de treinamento e de teste; geração do classificador, possivelmente com extração e/ou seleção de atributos; análise de desempenho do classificador.

Em termos computacionais, é preciso considerar de que maneira são representados os objetos a serem classificados e também o classificador em si. Muitos objetos podem ser descritos através de valores de atributos. Por exemplo, suponha que um estado brasileiro possa ser representado pelo seguinte vetor de atributos:

[Área (em km^2), Ponto mais Elevado (em metros), População, Densidade Populacional (hab./ km^2), Crescimento Demográfico (% ao ano), População Urbana (%), Domicílios, Carência Habitacional, Acesso à Água (%), Acesso à Rede de Esgoto (%), IDH, Mortalidade Infantil (%), Médicos (por 10000 hab.), Leitos Hospitalares (por 1000 hab.), Educação Infantil (matrículas), Rede Pública (%), Ensino Fundamental (matrículas), Rede Pública (%), Ensino Médio (matrículas), Rede Pública (%), Ensino Superior (matrículas), Rede Pública (%), Analfabetismo (%), Analfabetismo funcional (%), Senadores, Deputados Federais, Deputados Distritais/Estaduais, Eleitores, Participação PIB Nacional (%), PIB Agropecuário, PIB Indústria, PIB Serviços, PIB per capita (em R\$), Exportação (em milhões de dólares), Importação (em milhões de dólares), Geração de Energia (em GWh), Consumo (em GWh), Linhas Telefonia Fixa (em milhões), Linhas Telefonia Celular (em milhões), População (em milhões), Automóveis, Jornais Diários].

A partir desse vetor de atributos, os estados de São Paulo e Ceará, assim como o Distrito Federal, por exemplo, podem ser descritos de acordo com os valores dos atributos na tabela 2.1 na próxima página. Para propósitos de classificação, nem todos os atributos listados são necessariamente relevantes. Similarmente, o conjunto de atributos pode não ser suficiente para uma classificação correta.

No processo, os casos em que os objetos a serem classificados são representados por n -uplas ordenadas $(a_1, a_2, \dots, a_n)$, onde a_i , $i = 1..n$ são números de ponto fixo ou flutuante. Atributos categóricos ou qualitativos podem em geral ser quantizados ou representados também numericamente.

As categorias são representadas como números no conjunto $\{0, 1, \dots, k\}$, onde k representa o número de categorias e 0 tipicamente representa a categoria “desconhecida”.

A amostra de treinamento

$$\mathcal{T} = \{(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_m, f(x_m))\}$$

onde $x_i = (a_1, a_2, \dots, a_n)$ e $f(x_i) \in \{0, 1, \dots, k\}$, é considerada uma amostra de uma função desconhecida $y = f(x)$.

O objetivo do treinamento é tentar determinar y , ou seja, gerar um classificador que dado um novo valor x prediga o mais corretamente possível o valor de f correspondente.

No processo de geração de um classificador, a amostra utilizada é em geral dividida em dois subconjuntos disjuntos: uma parte denominada *amostra de treinamento* a ser utilizada

Estado	CE	DF	SP
Área (em km^2)	148.825,6	5.801,9	248.209,4
Ponto mais elevado (em m)	1.154	1.341	2.797
População	7.758.441	2.189.789	38.709.320
Densidade ($hab./km^2$)	50,9	364,1	152,2
Crescimento demográfico (% ao ano)	1,7	2,8	1,8
População urbana (%)	71,5	95,6	93,4
Domicílios	1.757.888	547.656	10.364.152
Carência habitacional	408.021	109.895	1.161.757
Acesso à água (%)	60,8	88,7	93,5
Acesso à rede de esgoto	33,9	89,7	88,2
IDH	0,700	0,844	0,820
Mortalidade infantil (%)	38,1	18,5	18
Médicos (por 10 mil habitantes)	8,8	33,9	22,1
Leitos hospitalares (por mil habitantes)	2,1	2,1	2,3
Educação infantil (matrículas)	309.467	89.485	1.675.808
Educação infantil - rede pública (% matrículas)	67,2	51	75,3
Ensino fundamental (matrículas)	1.835.064	370.511	5.898.603
Ensino fundamental - rede pública (% matrículas)	89,1	80,1	86,8
Ensino médio (matrículas)	380.834	127.358	2.100.823
Ensino médio - rede pública (% matrículas)	85,3	80,8	87
Ensino superior (matrículas)	63.429	79.329	898.643
Ensino superior - rede pública (% matrículas)	67,4	22,2	14,6
Analfabetismo (%)	24,7	5,2	6,1
Analfabetismo funcional (%)	44,3	16,1	20,2
Senadores	3	3	3
Deputados federais	22	8	70
Deputados distritais/estaduais	46	24	94
Eleitores	4.623.794	1.288.501	24.263.612
Eleitores (% do eleitorado brasileiro)	4,2	1,2	22,1
Participação no PIB nacional (%)	1,8	2,8	33,3
PIB agropecuária (%)	6,1	0,5	4,1
PIB indústria (%)	38,1	6,9	42,7
PIB serviços (%)	55,8	92,6	53,2
PIB per capita (em R\$)	2.858	15.725	10.642
Exportação (em milhões de dólares)	543,9	27,2	20.000
Importação (em milhões de dólares)	631,2	555,7	19.900
Geração de energia elétrica (em GWh)	1	98	61.603
Consumo de energia elétrica (em GWh)	5.500	3.465	87.065
Telefonia fixa (em milhões de linhas)	1,3	1	16,1
Telefonia celular (em milhões de linhas)	1,3	1,1	14,6
População da capital	2.256.233	2.189.789	10.677.019
Automóveis na capital	263.000	589.383	4.013.286
Jornais diários na capital	4	2	16

Tabela 2.1: Descrição de Estados Brasileiros

para gerar o classificador, e a outra parte denominada *amostra de teste* utilizada para avaliar o desempenho do classificador obtido.

Há diversas abordagens para a geração de classificadores. Algumas delas estão na tabela 2.2. Dentre essas, a estatística e redes neurais são bastante utilizadas.

As abordagens para geração de classificadores são conhecidas genericamente por aprendizado e podem ser divididas em supervisionada e não-supervisionada. No caso do aprendizado supervisionado, é dada uma amostra de objetos para os quais suas respectivas categorias são conhecidas e deseja-se que o classificador tenha a propriedade de generalização. Um classificador com essa propriedade é aquele capaz de classificar o mais corretamente possível não apenas os objetos da amostra, mas também a maior parte dos objetos do mesmo conjunto de objetos. No caso de aprendizado não-supervisionado, não são conhecidas as categorias às quais os objetos pertencem. A expectativa principal com relação às abordagens não-supervisionadas é que elas sejam capazes de separar os objetos em agrupamentos naturais intrínsecos aos objetos. As abordagens existentes utilizam em geral alguma medida de (dis)similaridade para agrupar os objetos em categorias. Outras vezes nem se sabe quais são os critérios, mas são dadas apenas os objetos e pergunta-se como classificá-los em dois conjuntos.

Um classificador tipicamente particiona o espaço de atributos de forma que cada elemento deste espaço seja associado a uma única categoria. A forma em que estas partições são representadas depende do algoritmo envolvido no treinamento. Por exemplo, árvores de decisão representam através de uma série de hiperplanos hierárquicos; já a técnica dos k -vizinhos mais próximos não usa nenhuma representação explícita, embora os hiperplanos que definem as fronteiras das diversas regiões possam ser determinados.

Abordagem	Representação	Função de reconhecimento	Critério típico de validação
Casamento de padrões	Exemplos, pixels curvas	Correlação, medida de distância	Erro de classificação
Estatística	Características	Função discriminante	Erro de classificação
Sintática ou estrutural	Primitivas	Regras, gramática	Erro de aceitação
Redes neurais	Exemplos, pixels, características	Função rede	Erro médio quadrático

Tabela 2.2: Abordagens para Geração de Classificadores (adaptado de [JDM00])

Figura 2.1: Projeto de Classificador

Projetar um classificador por aprendizado supervisionado envolve gerar o classificador a partir de exemplos de treinamento e validá-lo. A figura 2.1 na página precedente apresenta as atividades de um processo simplificado de projeto de classificador e a figura 2.2 na página seguinte apresenta as atividades de um processo simplificado de validação. O pré-processamento corresponde ao conjunto de operações que preparará o padrão para sua posterior representação. Essas operações podem ser remoção de ruídos, normalização etc. No treinamento, a extração/seleção de características consiste em reduzir o número de atributos, mantendo apenas aqueles mais relevantes à classificação, ou transformar o conjunto de características de maneira a obter um conjunto relevante à classificação. Por fim, ocorre o treinamento em si que gera um classificador para particionar o espaço de características. Na validação, as atividades até a extração/seleção de características devem ser idênticas às aplicadas no treinamento. Em seguida ocorre a classificação, isto é, a aplicação do classificador e a análise do seu desempenho.

Dependendo do desempenho, o classificador pode ser aceitável ou não. Caso não seja, é comum repetir o processo de treinamento alterando-se alguns parâmetros do processo.

Além das etapas de coleta de dados, pré-processamento e extração/seleção de características, as fases de treinamento e validação têm em comum o fato de utilizarem as amostras disponíveis. Existem várias possíveis formas para se dividir os exemplos em exemplos de treinamento e validação. Em geral, utiliza-se uma parcela para treinamento e outra para validação. Em função destes aspectos comuns aos dois processos podemos reuni-los em um único diagrama conforme a figura 2.3 na página 16. Se o desempenho do classificador não for satisfatório, todo ou parte do processo poderá ser repetido. Pode-se coletar mais dados, mudar o conjunto de atributos, particionar os exemplos de diferentes formas e, por fim, pode-se alterar parâmetros do algoritmo de treinamento ou até mesmo o próprio algoritmo.

Adicionalmente às técnicas para geração de classificadores, outras podem ser utilizadas para minimizar o erro de classificação ou permitir que a classificação seja mais eficiente computacionalmente. Exemplos são as técnicas de extração e projeção de características na tabela 2.3 na página 17 e as técnicas de seleção de características na tabela 2.4 na página 18.

O projeto de classificadores pode ser visualizado sob uma perspectiva orientada à objetos. A figura 2.4 na página 19 apresenta um diagrama com as classes relevantes por essa abordagem. No diagrama é possível fazer as seguintes observações. A primeira é que cada instância está associada a uma categoria. Segundo, cada categoria se relaciona com nenhuma, uma ou várias instâncias. Terceiro, há uma relação de agregação entre um padrão e seus atributos, caracterizando a natureza da instância como um conjunto de atributos. Quarto, vê-se que um exemplo pode ser tanto de treinamento quanto de teste.

Figura 2.2: Validação do Classificador

Figura 2.3: Projeto e Validação de Classificador

Abordagem	Propriedade
Análise de Componentes Principais (PCA)	Mapeamento linear; rápido; baseado em autovetor
Análise Discriminante Linear	Mapeamento linear supervisionado; rápido; baseado em autovetor
<i>Projection Pursuit</i>	Mapeamento linear; interativo; não-Gaussiano
Análise de Componentes Independentes (ICA)	Mapeamento linear; interativo; não-Gaussiano
<i>Kernel PCA</i>	Mapeamento não-linear; baseado em autovetor
Rede PCA	Mapeamento linear; interativo
PCA não-linear	Mapeamento linear; critério não-Gaussiano usualmente interativo
Rede auto-associativa não-linear	Mapeamento não-linear; critério não-Gaussiano; interativo
<i>Multidimensional scaling (MDS), and Sammon's projection</i>	Mapeamento não-linear; interativo
<i>Self-Organizing Map (SOM)</i>	Não-linear; interativo

Tabela 2.3: Técnicas de Extração e Projeção de Atributos (adaptado de [JDM00])

Abordagem	Propriedade
Busca exaustiva	Avalia todos os $\binom{d}{m}$ subconjuntos possíveis (d-número de características, m-número de exemplos)
<i>Branch-and-Bound Search</i>	Utiliza o método de pesquisa <i>branch-and-bound</i> ; apenas uma porção de todos os subconjuntos de características possíveis necessitam ser enumerados para se encontrar o subconjunto ótimo
<i>Best Individual Features</i>	Avalia todas as m características individualmente; seleciona a melhor característica individual;
<i>Sequential Forward Selection (SFS)</i>	Seleciona a característica mais simples e então adiciona uma característica por vez de forma que a combinação com as características selecionadas maximize a função critério
<i>Sequential Backward Selection (SBS)</i>	Inicia com todas as d características e remove sucessivamente uma característica por vez
<i>“Plus l-take away r” Selection</i>	Primeiro incrementa o subconjunto de características por l utilizando <i>forward selection</i> e então remove r características utilizando <i>backward selection</i>
Método <i>Sequential Forward Floating Search (SFFS) and Sequential Backward Floating Search (SBFS)</i>	Uma generalização do método “ <i>plus-l take away-r</i> ”; os valores de l e r são determinados automaticamente e atualizados dinamicamente

Tabela 2.4: Técnicas de Seleção de Atributos (adaptado de [JDN00])

Figura 2.4: Classes Relacionadas a Projeto de Classificadores

Ainda sob o enfoque orientado à objetos, os classificadores podem ser vistos como objetos. A figura 2.5 mostra um diagrama de classes. Nele, pode-se perceber a representação e o algoritmo como dois elementos importantes para definir um classificador. Como objetos, os classificadores podem compor uma classe. A caracterização dessa classe de classificadores é dada pela técnica de geração do classificador.

Figura 2.5: Classificador Modelado por Orientação à Objetos

Capítulo 3

Geração e análise de desempenho de classificadores

Há diversas abordagens para a geração de classificadores. Cada abordagem envolve uma ou mais técnicas. Neste capítulo serão apresentadas três técnicas básicas. Adicionalmente, há diversos enfoques para análise de desempenho de classificadores. Uma forma baseada em erro observado e intervalos de confiança será considerada.

Segue a abordagem de geração de classificadores por vizinhos mais próximos.

3.1 Vizinhos mais próximos

Seja $\vec{x}_q = (a_1, \dots, a_n)$ uma instância a ser classificada.

Seja $V = \{v_1, v_2, \dots, v_p\}$ o conjunto dos valores possíveis de uma função alvo f .

Seja $\mathcal{T} = \{(\vec{x}_1, f(\vec{x}_1)), (\vec{x}_2, f(\vec{x}_2)), \dots, (\vec{x}_m, f(\vec{x}_m))\}$ uma amostra de treinamento e $\vec{x}_i$ um exemplo genérico dessa amostra, $i = 1..m$.

Seja

$$d(\vec{x}_q, \vec{x}_i) = \sqrt{\sum_{r=1}^n (a_r^{(\vec{x}_i)} - a_r^{(\vec{x}_q)})^2}$$

a distância euclidiana, que mede a distância de $\vec{x}_q$ a cada exemplo de $\mathcal{T}$ a partir dos valores de seus atributos.

Seja k um número, $1 \leq k \leq m$, utilizado para selecionar os k exemplos da amostra com as menores medidas de distância a $\vec{x}_q$.

Pela técnica dos k vizinhos mais próximos, a hipótese $\hat{f}$ que classifica a instância $\vec{x}_q$ é dada por:

$$\hat{f}(\vec{x}_q) = \underset{v \in V}{\operatorname{argmax}} \left(\sum_{i=1}^k \delta(v, f(\vec{x}_i)) \right) \quad (3.1)$$

onde δ denota o símbolo delta de Kronecker, definido por $\delta(i, j) = 1$ se $i = j$ e $\delta(i, j) = 0$, caso contrário. A função $\operatorname{argmax}_{v \in V}$ retorna como argumento v com maior valor na soma dos símbolos de Kronecker.

3.1.1 Exemplo

Considere os parlamentos da tabela 3.1, definidos pelo seguinte vetor de atributos:

$P = [\text{no. parlamentares}, \text{volume orçamentário}, \text{no. comissões permanentes}]$

onde:

no. parlamentares corresponde ao número de parlamentares que atuam no parlamento;

volume orçamentário corresponde ao volume de orçamento gerido pelo parlamento, em milhões de unidades de valor;

no. comissões permanentes corresponde ao número de comissões permanentes existentes no parlamento.

A função alvo *porte institucional* corresponde ao porte do parlamento e o conjunto de seus possíveis valores é {Câmara, Assembléia}.

<i>Parlamento</i>	<i>No. Parlamentares</i>	<i>Volume Orçamentário</i>	<i>No. Comissões Permanentes</i>	<i>Porte Institucional</i>
P1	94	426	22	Assembléia
P2	46	23	14	Assembléia
P3	55	118	15	Câmara
P4	41	7	10	Câmara
P5	24	35	9	???

Tabela 3.1: Parlamentos e Portes Institucionais

Tomando os parlamentos P1, P2, P3 e P4 como exemplos, qual seria o porte institucional do parlamento P5?

O procedimento para obtenção da hipótese consiste em calcular as distâncias do parlamento P5 ao parlamentos P1, P2, P3 e P4 e, a partir de um valor de k , aplicar $\hat{f}$ definida na equação 3.1.

1. Cálculo das distâncias de P5

1.1. Cálculo da distância de P5 a P1

$$\begin{aligned}
d(P5, P1) &= \sqrt{\sum_{r=1}^3 (a_r^{(P1)} - a_r^{(P5)})^2} = \\
&= \sqrt{(a_1^{(P1)} - a_1^{(P5)})^2 + (a_2^{(P1)} - a_2^{(P5)})^2 + (a_3^{(P1)} - a_3^{(P5)})^2} = \\
&= \sqrt{(94 - 24)^2 + (426 - 35)^2 + (22 - 9)^2} = \sqrt{(70)^2 + (391)^2 + (13)^2} = \\
&= \sqrt{4900 + 152881 + 169} = \sqrt{157950} = 397,429 \Rightarrow \\
d(P5, P1) &= 397,429
\end{aligned}$$

1.2. Cálculo da distância de P5 a P2

$$\begin{aligned}
d(P5, P2) &= \sqrt{\sum_{r=1}^3 (a_r^{(P2)} - a_r^{(P5)})^2} = \\
&= \sqrt{(a_1^{(P2)} - a_1^{(P5)})^2 + (a_2^{(P2)} - a_2^{(P5)})^2 + (a_3^{(P2)} - a_3^{(P5)})^2} = \\
&= \sqrt{(46 - 24)^2 + (23 - 35)^2 + (14 - 9)^2} = \sqrt{(22)^2 + (-12)^2 + (5)^2} = \\
&= \sqrt{484 + 144 + 25} = \sqrt{653} = 25,554 \Rightarrow \\
d(P5, P2) &= 25,554
\end{aligned}$$

1.3. Cálculo da distância de P5 a P3

$$\begin{aligned}
d(P5, P3) &= \sqrt{\sum_{r=1}^3 (a_r^{(P3)} - a_r^{(P5)})^2} = \\
&= \sqrt{(a_1^{(P3)} - a_1^{(P5)})^2 + (a_2^{(P3)} - a_2^{(P5)})^2 + (a_3^{(P3)} - a_3^{(P5)})^2} = \\
&= \sqrt{(55 - 24)^2 + (118 - 35)^2 + (15 - 9)^2} = \sqrt{(31)^2 + (83)^2 + (6)^2} =
\end{aligned}$$

$$= \sqrt{961 + 6889 + 36} = \sqrt{7886} = 88,803 \Rightarrow$$

$$d(P5, P3) = 88,803$$

1.4. Cálculo da distância de P5 a P4

$$\begin{aligned} d(P5, P4) &= \sqrt{\sum_{r=1}^3 (a_r^{(P4)} - a_r^{(P5)})^2} = \\ &= \sqrt{(a_1^{(P4)} - a_1^{(P5)})^2 + (a_2^{(P4)} - a_2^{(P5)})^2 + (a_3^{(P4)} - a_3^{(P5)})^2} = \\ &= \sqrt{(41 - 24)^2 + (7 - 35)^2 + (10 - 9)^2} = \sqrt{(17)^2 + (-28)^2 + (1)^2} = \\ &= \sqrt{289 + 784 + 1} = \sqrt{1074} = 32,772 \Rightarrow \\ d(P5, P4) &= 32,772 \end{aligned}$$

2. Obtenção da hipótese para a função alvo

A tabela 3.2 apresenta a consolidação das distâncias dos parlamentos à P5 e os respectivos portes institucionais.

<i>Parlamento</i>	<i>Distância à P5</i>	<i>Porte Institucional</i>
P1	397,429	Assembléia
P2	25,554	Assembléia
P3	88,803	Câmara
P4	32,772	Câmara

Tabela 3.2: Parlamentos, Distâncias e Portes Institucionais

Para $k = 1$, seleciona-se apenas o primeiro parlamento mais próximo, no caso, P2. Como seu porte institucional é Assembléia, o porte institucional de P5 será classificado como Assembléia.

Aplicando $\hat{f}$ definida na equação 3.1 na página 22, tem-se

$$\hat{f}(P5) = \underset{v \in V}{argmax} \left(\sum_{i=1}^1 \delta(v, f(P2)) \right)$$

onde $v = \{\text{Câmara, Assembléia}\}$ e $f(P2) = \text{Assembléia}$.

Para $v = \text{Assembléia}$, tem-se $\delta(\text{Assembléia, Assembléia}) = 1$.

Para $v = \text{Câmara}$, $\delta(\text{Câmara, Assembléia}) = 0$.

Logo, $\hat{f}(P5) = \text{Assembléia}$.

Esse caso particular da técnica, em que $k = 1$, é denominado vizinho mais próximo e denotado por VMP-1.

Já considerando $k = 3$, seleciona-se os três parlamentos com menores medidas de distância à P5, nesse caso, P2, P4 e P3. Como seus respectivos portes são Assembléia, Câmara e Câmara, o porte institucional de P5 será classificado como Câmara. A tabela 3.3 apresenta um resumo dos cálculos ao se aplicar a técnica vizinhos mais próximos por meio da equação 3.1 na página 22 .

$k = 3$	$v = \{\text{Câmara, Assembléia}\}$
	$f(x_1) = f(P2) = \text{Assembléia}$
	$f(x_2) = f(P4) = \text{Câmara}$
	$f(x_3) = f(P3) = \text{Câmara}$
$\sum_{i=1}^3 \delta(\text{Câmara}, f(x_i)) = 2$	$\sum_{i=1}^3 \delta(\text{Assembléia}, f(x_i)) = 1$
$\hat{f}(P5) = \underset{v \in V}{\operatorname{argmax}} \left(\sum_{i=1}^3 \delta(v, f(x_i)) \right) = \text{Câmara}$	

Tabela 3.3: Aplicação da Técnica Vizinhos Mais Próximos para $k = 3$

Concluindo, tem-se que por VMP-1, o parlamento P5 tem porte de Assembléia. Já por VMP-3, o parlamento P5 tem porte institucional de Câmara.

3.1.2 Variações da técnica

Há diversas variações da técnica por vizinhos mais próximos. Uma primeira é denominada vizinho mais próximo com opção de rejeição. Nela, é exigida uma frequência mínima de um valor da função alvo para que ele seja atribuído a uma instância $\vec{x}_q$. Assim, se o valor mais frequente dentre os k vizinhos mais próximos for y , essa frequência deve ser de pelo menos r vezes. Caso y não ocorra pelo menos r vezes, nenhuma classificação é feita. Em outras palavras, essa variação é similar ao processo de votação em que a candidata mais votada vencerá desde que obtenha um número mínimo de votos. As candidatas são as categorias. Como complemento a essa variação, pode-se fixar um número mínimo de votos a serem obtidos por cada categoria. Se apenas uma categoria ultrapassou a quantidade de votos mínima que lhe é exigida, então ela é a escolhida. Caso mais de uma categoria tenha ultrapassado seu respectivo limiar de votos, então a categoria vencedora pode ser uma dentre elas. A escolha

da vencedora pode ser feita de forma aleatória. Ou pode-se escolher a categoria que mais extrapolou em termos absolutos ou relativos o seu limiar de votos. Uma outra estratégia é partir da categoria mais representativa dentre os k vizinhos e verificar se ela atingiu seu limiar de votação. Caso não tenha atingido, passa-se para a segunda categoria com maior representatividade e assim sucessivamente.

Outra variante é a técnica do vizinho mais próximo condensada. Essa técnica tem como objetivo diminuir tempo de processamento. Para isso, a estratégia consiste em decrementar o número de exemplos a serem utilizados na classificação. Esses exemplos comporão uma amostra de treinamento menor que a amostra dada inicialmente. Ou seja, se $\mathcal{T}$ é a amostra de treinamento dada, então $\mathcal{T}'$ é a amostra condensada de $\mathcal{T}$ da seguinte maneira. Inicialmente $\mathcal{T}'$ é um conjunto vazio e $\mathcal{T}$ contém os exemplos de treinamento. Um exemplo x_r é escolhido de $\mathcal{T}$ e colocado em $\mathcal{T}'$. Essa escolha pode ser aleatória ou determinada a priori. A partir daí, cada exemplo em $\mathcal{T}$ é classificado por vizinho mais próximo utilizando-se $\mathcal{T}'$ como amostra de treinamento. Se o exemplo é classificado corretamente, então ele permanece no conjunto $\mathcal{T}$. Caso ele seja classificado erroneamente, então é transferido para $\mathcal{T}'$. Ao se classificar todos os exemplos de $\mathcal{T}$ repete-se o processo até não haver mais transferência de $\mathcal{T}$ para $\mathcal{T}'$. Esta é a amostra condensada. Dado que os exemplos dessa amostra, uma vez inseridos, não são excluídos, um complemento à variante condensada consiste em retirar os exemplos que sejam corretamente classificados a partir de outros exemplos da mesma amostra condensada. Essa técnica é denominada vizinho mais próximo reduzida.

3.2 Árvores de decisão

Uma árvore A é um par ordenado (V, E) com as seguintes propriedades:

1. V é um conjunto finito não vazio de elementos denominados **nodos**;
2. há um nodo especial em V denominado **raiz** da árvore;
3. os nodos em V exceto a raiz, caso existam, estão particionados em m conjuntos disjuntos $A_1, A_2, \dots, A_m$ em que cada um desses conjuntos é, por sua vez, uma árvore;
4. E é um subconjunto do conjunto de pares não ordenados de nodos distintos de V , cujos elementos são denominados **arestas**.

Partindo da definição formal recursiva acima, uma árvore é uma abstração matemática que impõe uma estrutura hierárquica sobre os nodos. Não há problema de circularidade na definição, uma vez que árvores com um nodo contém apenas a raiz e árvores com n nodos, $n > 1$, são definidas em termos de árvores com menos de n nodos.

Árvores podem ser representadas por meio de diagramas. Além disso, diversas situações podem ser convenientemente descritas por meio de árvores. Por exemplo, os nodos podem representar estruturas de uma norma jurídica, com arestas representando uma relação de agrupamento. A figura 3.1 descreve um título da Constituição da República Federativa do Brasil com seus capítulos, e estes com suas respectivas seções.

Figura 3.1: Exemplo de Árvore - Título V da Constituição Brasileira

Há diversos conceitos relacionados a árvores. Segue uma terminologia básica.

Dada uma árvore A com raiz r , cada conjunto $A_1, A_2, \dots, A_m$ é denominado **subárvore** do nodo r . Define-se **grau de um nodo** como o número de subárvores desse nodo. Um nodo com grau zero é denominado **folha**. O **grau de uma árvore** é o máximo dentre os graus de todos os nodos da árvore.

Dada uma árvore A com raiz r e subárvores $A_1, A_2, \dots, A_m$, as raízes de $A_1, A_2, \dots, A_m$ são denominadas **filhos** de r e r é denominado **pai** dos filhos. Os filhos do mesmo pai são denominados **irmãos**.

Um **caminho** numa árvore é uma sequência de nodos distintos em que os sucessivos nodos estão conectados por arestas da árvore.

A partir do conceito de caminho pode-se descrever a propriedade básica de uma árvore. A propriedade é que há precisamente um caminho conectando quaisquer dois nodos numa árvore. Se há mais de um caminho, ou se não há caminho entre alguns pares de nodos, então a estrutura descrita não é uma árvore e sim outro conceito matemático denominado grafo.

Os **ascendentes** de um nodo são todos os nodos ao longo do caminho desde a raiz da árvore até o respectivo nodo. Os **descendentes** de um nodo é o conjunto de todos os nodos de suas subárvores.

O **nível de uma árvore** é definido da seguinte maneira:

1. a raiz da árvore está no nível 1;
2. se um nodo está no nível n então seus filhos estarão no nível $n + 1$.

A **altura de uma árvore** é definida como o máximo dentre os níveis de todos os nodos da árvore.

Na técnica de classificação por árvore de decisão, a representação da função aprendida é feita por uma árvore de decisão. Uma árvore de decisão é uma árvore que classifica instâncias, percorrendo-se a árvore, da raiz até alguma folha. Essa árvore é construída a partir de um algoritmo. Esse algoritmo particiona o espaço de atributos, dividindo-o em regiões disjuntas e definindo os limites de decisão. Regiões em que os exemplos são classificados na mesma categoria possui maior grau de pureza que regiões onde os exemplos são classificados em categorias distintas. O problema algorítmico consiste em construir uma árvore que ofereça regiões puras. O passo inicial do algoritmo consiste em plantar a árvore através de uma raiz. A questão é como escolher a melhor semente (no caso, o melhor atributo) para ser plantada como a raiz da árvore. Para isso, em geral utiliza-se como heurística a *navalha de Ockham* (*filósofo, 1300-1349*), que consiste em preferir árvores “pequenas” a árvores “grandes”. Seguindo essa heurística, para selecionar o melhor atributo a ser plantado como raiz da árvore utiliza-se de medidas que avaliam a impureza da amostra de treinamento e o potencial de cada atributo em servir como divisor dessa amostra em dois conjuntos mais puros.

Seja T um conjunto de exemplos de treinamento. Pode-se calcular uma medida de impureza desse conjunto em relação a classificação alvo por meio de uma medida denominada **entropia**. Em termos estatísticos ela pode ser definida como segue.

Seja X uma variável aleatória discreta que assume n valores distintos. Seja P_x a distribuição de probabilidade para X e $Prob[X = x] = p_x$. A entropia da distribuição é

$$\boxed{Entropia(P_x) = - \sum_x p_x \log_2 p_x} \quad (3.2)$$

Figura 3.2: Função Logarítmica Binária

onde $\log_2$ é a função logarítmica binária, cujo gráfico é mostrado na figura 3.2 na página precedente.

Se X assume apenas dois valores, então $P_x = (p, 1 - p)$ e

$$Entropia(P_x) = -p \log_2 p - (1 - p) \log_2 (1 - p) \quad (3.3)$$

Nesse caso, a entropia possui o máximo em $p = 0.5$ como mostra a figura 3.3.

Figura 3.3: Entropia para uma Variável Aleatória Discreta X com 2 valores distintos

Para esse caso,

$$Entropia(T) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \quad (3.4)$$

onde

$p_{\oplus}$ = proporção de exemplos positivos

$p_{\ominus}$ = proporção de exemplos negativos

No cálculo de entropia, uma vez que as proporções podem assumir valor 0, a função logarítmica inicialmente definida de $\mathbb{R}_+^*$ em $\mathbb{R}$ é redefinida para $\mathbb{R}_+ \mapsto \mathbb{R}$, onde $\log_2 0 = 0$.

6 Dado um conjunto de exemplos de treinamento S definidos por um conjunto de atributos A , define-se uma propriedade estatística denominada **ganho de informação** que mede quanto cada atributo separa os exemplos de treinamento em relação a classificação alvo, definida como:

$$\boxed{Ganho(\mathcal{T}, A) = Entropia(\mathcal{T}) - \sum_{a \in A} \frac{|T_a|}{|\mathcal{T}|} Entropia(T_a)} \quad (3.5)$$

A partir dessas medidas pode-se construir uma árvore de decisão, como mostra o exemplo a seguir.

3.2.1 Exemplo

Considere instâncias de proposições legislativas da tabela 3.4 na próxima página que se tornaram normas jurídicas, definidas pelos seguinte vetor de atributos:

$P = [\text{identificador}, \text{tipo}, \text{assunto}, \text{autoria}, \text{parecer}]$

onde:

identificador denomina a proposição, tendo a forma P_n , $n \in \mathbb{Z}_+$ como, por exemplo, P_1 ;

tipo corresponde ao tipo da proposição, restrito ao conjunto $\{PLO, PLC\}$, onde $PLO =$ Projeto de Lei Ordinária e $PLC =$ Projeto de Lei Complementar;

assunto corresponde ao assunto da proposição, restrito ao conjunto $\{\text{educação, tecnologia, saúde}\}$;

autoria corresponde ao autor da proposição, restrita ao conjunto $\{D1, D2, D3\}$, que denota deputados genéricos, $D1 =$ Deputado 1, $D2 =$ Deputado 2, $D3 =$ Deputado 3;

parecer corresponde a existência de parecer pela inadmissibilidade ou rejeição da proposição em alguma das comissões parlamentares nas quais a proposição tramitou. O domínio desse atributo é $\{\text{Não, Sim}\}$.

A função alvo *ação* corresponde a existência de alguma ação de inconstitucionalidade contra a norma jurídica originada a partir da proposição. O conjunto de seus possíveis valores é $\{\text{Não, Sim}\}$.

<i>Proposição</i>	<i>Tipo</i>	<i>Assunto</i>	<i>Autoria</i>	<i>Parecer</i>	<i>Ação</i>
P1	PLO	Educação	D1	Não	Não
P2	PLC	Educação	D1	Não	Não
P3	PLO	Tecnologia	D1	Não	Sim
P4	PLO	Saúde	D2	Não	Sim
P5	PLO	Saúde	D3	Sim	Sim
P6	PLC	Tecnologia	D2	Não	Sim
P7	PLC	Tecnologia	D3	Sim	Sim
P8	PLO	Educação	D2	Não	Não
P9	PLO	Educação	D3	Sim	Sim
P10	PLO	Saúde	D2	Sim	Sim
P11	PLC	Educação	D2	Sim	Sim
P12	PLC	Saúde	D3	Sim	Não
P13	PLO	Tecnologia	D1	Sim	Sim
P14	PLC	Saúde	D2	Não	Não

Tabela 3.4: Proposições Legislativas ($\mathcal{T}$)

$$1. \mathcal{T} = \{P1, P2, P3, \dots, P14\}$$

1.1. Cálculo da Entropia de $\mathcal{T}$

$$\begin{aligned} Entropia(\mathcal{T}) &= Entropia([9+, 5-]) = -\frac{9}{14} \log_2^{9/14} - \frac{5}{14} \log_2^{5/14} = \\ &= -0,643 \cdot (-0,637) - 0,357 \cdot (-1,486) = 0,410 + 0,531 \Rightarrow \end{aligned}$$

$$Entropia(\mathcal{T}) = 0,941$$

1.2. Cálculo do Ganho de Informação de Tipo

$$Ganho(\mathcal{T}, Tipo) = Entropia(\mathcal{T}) - \sum_{a \in \{PLO, PLC\}} \frac{|\mathcal{T}_a|}{|\mathcal{T}|} Entropia(\mathcal{T}_a)$$

$$\begin{aligned} Entropia(\mathcal{T}_{PLO}) &= Entropia([6+, 2-]) = -\frac{6}{8} \log_2^{6/8} - \frac{2}{8} \log_2^{2/8} = \\ &= -0,750 \cdot (-0,415) - 0,250 \cdot (-2) = 0,311 + 0,500 = 0,811 \end{aligned}$$

$$Entropia(\mathcal{T}_{PLC}) = Entropia([3+, 3-]) = -\frac{3}{6} \log_2^{3/6} - \frac{3}{6} \log_2^{3/6} = 1$$

$$Ganho(\mathcal{T}, Tipo) = 0,941 - \left(\frac{8}{14} \cdot 0,811 + \frac{6}{14} \cdot 1 \right) =$$

$$= 0,941 - (0,463 + 0,429) = 0,941 - 0,892 \Rightarrow$$

$$Ganho(\mathcal{T}, Tipo) = 0,049$$

1.3. Cálculo do Ganho de Informação de Assunto

$$Ganho(\mathcal{T}, Assunto) = Entropia(\mathcal{T}) - \sum_{a \in \{E, T, S\}} \frac{|\mathcal{T}_a|}{|\mathcal{T}|} Entropia(\mathcal{T}_a)$$

$$\begin{aligned} Entropia(\mathcal{T}_E) &= Entropia([2+, 3-]) = -\frac{2}{5} \log_2^{2/5} - \frac{3}{5} \log_2^{3/5} = \\ &= -0,4 \cdot (-1,322) - 0,6 \cdot (-0,737) = 0,529 + 0,442 = 0,971 \end{aligned}$$

$$Entropia(\mathcal{T}_T) = Entropia([4+, 0-]) = -\frac{4}{4} \log_2^{4/4} - \frac{0}{4} \log_2^{0/4} = 0$$

$$Entropia(\mathcal{T}_S) = Entropia([3+, 2-]) = -\frac{3}{5} \log_2^{3/5} - \frac{2}{5} \log_2^{2/5} = 0,971$$

$$\begin{aligned} Ganho(\mathcal{T}, Assunto) &= 0,941 - \left(\frac{5}{14} \cdot 0,971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0,971 \right) = \\ &= 0,941 - (0,347 + 0 + 0,347) = 0,941 - 0,694 \Rightarrow \end{aligned}$$

$$Ganho(\mathcal{T}, Assunto) = 0,247$$

1.4. Cálculo do Ganho de Informação de Autoria

$$Ganho(\mathcal{T}, Autoria) = Entropia(\mathcal{T}) - \sum_{a \in \{D1, D2, D3\}} \frac{|\mathcal{T}_a|}{|\mathcal{T}|} Entropia(\mathcal{T}_a)$$

$$Entropia(\mathcal{T}_{D1}) = Entropia([2+, 2-]) = -\frac{2}{4} \log_2^{2/4} - \frac{2}{4} \log_2^{2/4} = 1$$

$$\begin{aligned} Entropia(\mathcal{T}_{D2}) &= Entropia([4+, 2-]) = -\frac{4}{6} \log_2^{4/6} - \frac{2}{6} \log_2^{2/6} = \\ &= -0,667 \cdot (-0,585) - 0,333 \cdot (-1,585) = 0,390 + 0,528 = 0,918 \end{aligned}$$

$$\begin{aligned} Entropia(\mathcal{T}_{D3}) &= Entropia([3+, 1-]) = -\frac{3}{4} \log_2^{3/4} - \frac{1}{4} \log_2^{1/4} = \\ &= -0,750 \cdot (-0,415) - 0,250 \cdot (-2) = 0,311 + 0,500 = 0,811 \end{aligned}$$

$$\begin{aligned} \text{Ganho}(\mathcal{T}, \text{Autoria}) &= 0,941 - \left(\frac{4}{14} \cdot 1 + \frac{6}{14} \cdot 0,918 + \frac{4}{14} \cdot 0,811 \right) = \\ &= 0,941 - (0,286 + 0,393 + 0,232) = 0,941 - 0,911 \Rightarrow \end{aligned}$$

$$\text{Ganho}(\mathcal{T}, \text{Autoria}) = 0,030$$

1.5. Cálculo do Ganho de Informação de Parecer

$$\text{Ganho}(\mathcal{T}, \text{Parecer}) = \text{Entropia}(\mathcal{T}) - \sum_{a \in \{N, S\}} \frac{|\mathcal{T}_a|}{|\mathcal{T}|} \text{Entropia}(\mathcal{T}_a)$$

$$\begin{aligned} \text{Entropia}(\mathcal{T}_N) &= \text{Entropia}([3+, 4-]) = -\frac{3}{7} \log_2^{3/7} - \frac{4}{7} \log_2^{4/7} = \\ &= -0,429 \cdot (-1,223) - 0,571 \cdot (-0,807) = 0,525 + 0,461 = 0,986 \end{aligned}$$

$$\begin{aligned} \text{Entropia}(\mathcal{T}_S) &= \text{Entropia}([6+, 1-]) = -\frac{6}{7} \log_2^{6/7} - \frac{1}{7} \log_2^{1/7} = \\ &= -0,857 \cdot (-0,222) - 0,143 \cdot (-2,808) = 0,190 + 0,402 = 0,592 \end{aligned}$$

$$\begin{aligned} \text{Ganho}(\mathcal{T}, \text{Parecer}) &= 0,941 - \left(\frac{7}{14} \cdot 0,986 + \frac{7}{14} \cdot 0,592 \right) = \\ &= 0,941 - (0,493 + 0,296) = 0,941 - 0,789 \Rightarrow \end{aligned}$$

$$\text{Ganho}(\mathcal{T}, \text{Parecer}) = 0,152$$

1.6. Escolha do Atributo com Maior Ganho de Informação

$$\text{Ganho}(\mathcal{T}, \text{Tipo}) = 0,049$$

$$\text{Ganho}(\mathcal{T}, \text{Assunto}) = 0,247 \quad \blacktriangleleft \text{Maior Ganho de Informação}$$

$$\text{Ganho}(\mathcal{T}, \text{Autoria}) = 0,030$$

$$\text{Ganho}(\mathcal{T}, \text{Parecer}) = 0,152$$

$$2. \mathcal{T}' = \{P1, P2, P8, P9, P11\}$$

2.1. Cálculo da Entropia de $\mathcal{T}'$

$$\begin{aligned} \text{Entropia}(\mathcal{T}') &= \text{Entropia}([2+, 3-]) = -\frac{2}{5} \log_2^{2/5} - \frac{3}{5} \log_2^{3/5} = \\ &= -0,4 \cdot (-1,322) - 0,6 \cdot (-0,737) = 0,529 + 0,442 \Rightarrow \end{aligned}$$

$$\text{Entropia}(\mathcal{T}') = 0,971$$

Figura 3.4: Construção de uma Árvore de Decisão (passo 1)

Proposição	Tipo	Assunto	Autoria	Parecer	Ação
P1	PLO	Educação	D1	Não	Não
P2	PLC	Educação	D1	Não	Não
P8	PLO	Educação	D2	Não	Não
P9	PLO	Educação	D3	Sim	Sim
P11	PLC	Educação	D2	Sim	Sim

Tabela 3.5: Proposições Legislativas (T')

2.2. Cálculo do Ganho de Informação de Tipo

$$Ganho(T', Tipo) = Entropia(T') - \sum_{a \in \{PLO, PLC\}} \frac{|T'_a|}{|T'|} Entropia(T'_a)$$

$$\begin{aligned} Entropia(T'_{PLO}) &= Entropia([1+, 2-]) = -\frac{1}{3} \log_2^{1/3} - \frac{2}{3} \log_2^{2/3} = \\ &= -0,333 \cdot (-1,585) - 0,667 \cdot (-0,585) = 0,528 + 0,390 = 0,918 \end{aligned}$$

$$Entropia(T'_{PLC}) = Entropia([1+, 1-]) = 1$$

$$\begin{aligned} Ganho(T', Tipo) &= 0,971 - \left(\frac{3}{5} \cdot 0,918 + \frac{2}{5} \cdot 1\right) = \\ &= 0,971 - (0,551 + 0,400) = 0,971 - 0,951 \Rightarrow \end{aligned}$$

$$Ganho(T', Tipo) = 0,020$$

2.3. Cálculo do Ganho de Informação de Autoria

$$Ganho(T', Autoria) = Entropia(T') - \sum_{a \in \{D1, D2, D3\}} \frac{|T'_a|}{|T'|} Entropia(T'_a)$$

$$Entropia(T'_{D1}) = Entropia([0+, 2-]) = -\frac{0}{2} \log_2^{0/2} - \frac{2}{2} \log_2^{2/2} = 0$$

$$Entropia(T'_{D2}) = Entropia([1+, 1-]) = 1$$

$$Entropia(T'_{D3}) = Entropia([1+, 0-]) = 0$$

$$Ganho(T', Autoria) = 0,971 - \left(\frac{2}{5} \cdot 0 + \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 0\right) = 0,971 - 0,400 \Rightarrow$$

$$Ganho(T', Autoria) = 0,571$$

2.4. Cálculo do Ganho de Informação de Parecer

$$\text{Ganho}(T', \text{Parecer}) = \text{Entropia}(T') - \sum_{a \in \{N, S\}} \frac{|T'_a|}{|T'|} \text{Entropia}(T'_a)$$

$$\text{Entropia}(T'_N) = \text{Entropia}([0+, 3-]) = 0$$

$$\text{Entropia}(T'_S) = \text{Entropia}([2+, 0-]) = 0$$

$$\text{Ganho}(T', \text{Parecer}) = 0,971 - 0 = 0,971$$

2.5. Escolha do Atributo com Maior Ganho de Informação

$$\text{Ganho}(T', \text{Tipo}) = 0,020$$

$$\text{Ganho}(T', \text{Autoria}) = 0,571$$

$$\text{Ganho}(T', \text{Parecer}) = 0,971 \quad \blacktriangleleft \text{Maior Ganho de Informação}$$

Proposição	Tipo	Assunto	Autoria	Parecer	Ação
P4	PLO	Saúde	D2	Não	Sim
P5	PLO	Saúde	D3	Sim	Sim
P10	PLO	Saúde	D2	Sim	Sim
P12	PLC	Saúde	D3	Sim	Não
P14	PLC	Saúde	D2	Não	Não

Tabela 3.6: Proposições Legislativas (T'')

$$3. \quad T'' = \{P4, P5, P10, P12, P14\}$$

3.1. Cálculo da Entropia de T''

$$\text{Entropia}(T'') = \text{Entropia}([3+, 2-]) = -\frac{3}{5} \log_2^{3/5} - \frac{2}{5} \log_2^{2/5} =$$

$$= -0,6 \cdot (-0,737) - 0,4 \cdot (-1,322) = 0,442 + 0,529 \quad \Rightarrow$$

$$\text{Entropia}(T'') = 0,971$$

Figura 3.5: Construção de uma Árvore de Decisão (passo 2)

3.2. Cálculo do Ganho de Informação de Tipo

$$Ganho(\mathcal{T}'', Tipo) = Entropia(\mathcal{T}'') - \sum_{a \in \{PLO, PLC\}} \frac{|\mathcal{T}_a''|}{|\mathcal{T}''|} Entropia(\mathcal{T}_a'')$$

$$Entropia(\mathcal{T}_{PLO}'') = Entropia([3+, 0-]) = 0$$

$$Entropia(\mathcal{T}_{PLC}'') = Entropia([0+, 2-]) = 0$$

$$Ganho(\mathcal{T}'', Tipo) = 0,971 - \left(\frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0\right) = 0,971 - 0 \Rightarrow$$

$$Ganho(\mathcal{T}'', Tipo) = 0,971$$

3.3. Cálculo do Ganho de Informação de Autoria

$$Ganho(\mathcal{T}'', Autoria) = Entropia(\mathcal{T}'') - \sum_{a \in \{D2, D3\}} \frac{|\mathcal{T}_a''|}{|\mathcal{T}''|} Entropia(\mathcal{T}_a'')$$

$$\begin{aligned} Entropia(\mathcal{T}_{D2}'') &= Entropia([2+, 1-]) = -\frac{2}{3} \log_2^{2/3} - \frac{1}{3} \log_2^{1/3} = \\ &= -0,667 \cdot (-0,585) - 0,333 \cdot (-1,585) = 0,390 + 0,528 = 0,918 \end{aligned}$$

$$Entropia(\mathcal{T}_{D3}'') = Entropia([1+, 1-]) = 1$$

$$\begin{aligned} Ganho(\mathcal{T}'', Autoria) &= 0,971 - \left(\frac{3}{5} \cdot 0,918 + \frac{2}{5} \cdot 1\right) = \\ &= 0,971 - (0,551 + 0,400) = 0,971 - 0,951 \Rightarrow \end{aligned}$$

$$Ganho(\mathcal{T}'', Autoria) = 0,020$$

3.4. Cálculo do Ganho de Informação de Parecer

$$Ganho(\mathcal{T}'', Parecer) = Entropia(\mathcal{T}'') - \sum_{a \in \{N, S\}} \frac{|\mathcal{T}_a''|}{|\mathcal{T}''|} Entropia(\mathcal{T}_a'')$$

$$Entropia(\mathcal{T}_N'') = Entropia([1+, 1-]) = 1$$

$$Entropia(\mathcal{T}_S'') = Entropia([2+, 1-]) = 0,918$$

$$\begin{aligned}
 \text{Ganho}(T'', \text{Parecer}) &= 0,971 - \left(\frac{2}{5} \cdot 1 + \frac{3}{5} \cdot 0,918\right) = \\
 &= 0,971 - (0,400 + 0,551) = 0,971 - 0,951 \Rightarrow
 \end{aligned}$$

$$\text{Ganho}(T'', \text{Parecer}) = 0,020$$

3.5. Escolha do Atributo com Maior Ganho de Informação

$\text{Ganho}(T'', \text{Tipo}) = 0,971 \blacktriangleleft$ Maior Ganho de Informação

$\text{Ganho}(T'', \text{Autoria}) = 0,020$

$\text{Ganho}(T'', \text{Parecer}) = 0,020$

[P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14]

Figura 3.6: Construção de uma Árvore de Decisão (passo 3)

3.2.2 Variações da técnica

Em geral, as variações da técnica por árvores de decisão se baseiam em variações nas medidas adotadas e nos algoritmos.

Variações nas medidas adotadas envolvem escolhas de medidas alternativas de impureza à entropia, como por exemplo a função Gini, dada por

$$Gini(n) = \sum_j \sum_{i \neq j} p(i|n)p(j|n) = 1 - \sum_j [p(j|n)]^2 \quad (3.6)$$

onde n é um nodo e i, j são categorias. Esta função também satisfaz ao critério de medir o potencial de purificação de um nodo.

Uma alternativa à medida ganho de informação é a taxa de ganho, dada por

$$TaxaGanho(\mathcal{T}, A) = \frac{Ganho(\mathcal{T}, A)}{InformacaoDivisao(\mathcal{T}, A)} \quad (3.7)$$

onde o denominador é dado por

$$InformacaoDivisao(\mathcal{T}, A) = - \sum_{i=1}^c \frac{|\mathcal{T}_i|}{|\mathcal{T}|} \log_2 \frac{|\mathcal{T}_i|}{|\mathcal{T}|} \quad (3.8)$$

$\mathcal{T}_1$ a $\mathcal{T}_k$ são os k subconjuntos de exemplos resultantes do particionamento de $\mathcal{T}$ pelo atributo de valor k do atributo A .

Com relação as variações dos algoritmos, um dos aspectos de variação é sobre podagem da árvore. Essa estratégia se baseia no fato das árvores tenderem a “decorar” os exemplos de treinamento, fenômeno denominado *overfitting*.

Os algoritmos mais conhecidos são CART, ID3, C4.5, ASSISTANT, CN2, ID5R, ITI, dentre outros.

3.3 Redes Neurais Artificiais

Denomina-se **neurônio artificial** a dupla ordenada $U = (\vec{w}, g)$ onde $\vec{w} = [w_1, w_2, \dots, w_n]$ é um vetor de n números reais denominado **vetor de pesos** e g é uma função denominada **função de ativação**.

No contexto de redes neurais artificiais, uma instância $\vec{x} = [a_1, a_2, \dots, a_n]$ é denominada **vetor de entrada** e seus atributos denominados **valores de entrada**.

Dados um vetor de pesos $\vec{w} = [w_1, w_2, \dots, w_n]$ e um vetor de entrada $\vec{x} = [a_1, a_2, \dots, a_n]$, o produto interno entre $\vec{w}$ e $\vec{x}$ é dado por

$$\langle \vec{w}, \vec{x} \rangle = w_1 a_1 + w_2 a_2 + \dots + w_n a_n$$

$\langle \vec{w}, \vec{x} \rangle$ é uma aplicação em $\mathbb{R}^n$ que transforma cada par ordenado $(\vec{w}, \vec{x}) \in \mathbb{R}^n \times \mathbb{R}^n$ em um número real.

Denomina-se **valor de saída** do neurônio artificial, o número real $g(\langle \vec{w}, \vec{x} \rangle)$ obtido na aplicação da função de ativação g ao produto interno $\langle \vec{w}, \vec{x} \rangle$.

A figura 3.7 exemplifica um neurônio, graficamente.

Figura 3.7: Exemplo de Neurônio Artificial

Um exemplo de função de ativação é $g : \mathbb{R} \rightarrow \mathbb{R}$, definida por

$$g(\langle \vec{w}, \vec{x} \rangle) = w_1 a_1 + w_2 a_2 + \dots + w_n a_n$$

isto é, $g(< \vec{w}, \vec{x} >) = < \vec{w}, \vec{x} >$. Uma vez que o valor da função de ativação é uma combinação linear, neurônios com essa função são denominados neurônios lineares.

Por outro lado, neurônios não lineares podem ser obtidos a partir de funções de ativação não lineares. Por exemplo, a função $g : \mathbb{R} \longrightarrow \{0, 1\}$, dada por

$$g(< \vec{w}, \vec{x} >) = \begin{cases} 1, & \text{se } w_1 a_1 + w_2 a_2 + \dots + w_n a_n \geq \theta \\ 0, & \text{se } w_1 a_1 + w_2 a_2 + \dots + w_n a_n < \theta \end{cases} \quad (3.9)$$

Essa função de ativação funciona como filtro, transformando a combinação linear $w_1 a_1 + w_2 a_2 + \dots + w_n a_n$ no valor 0 ou 1. Um neurônio com essa função de ativação possui um número real θ denominado **valor de limiar**. Nota-se que o valor de limiar é a quantidade a ser superada pela combinação linear $w_1 a_1 + w_2 a_2 + \dots + w_n a_n$ dos valores de entrada com os pesos para que o neurônio assuma um ou outro valor dado pela função de ativação.

O valor de limiar de um neurônio pode ser tratado como um peso adicional, associado a um valor de entrada fixado em 1. Isso é sempre possível uma vez que:

$$\begin{aligned} w_1 a_1 + w_2 a_2 + \dots + w_n a_n &\geq \theta \\ -\theta + w_1 a_1 + w_2 a_2 + \dots + w_n a_n &\geq 0 \\ (-\theta) \cdot 1 + w_1 a_1 + w_2 a_2 + \dots + w_n a_n &\geq 0 \\ w_0 a_0 + w_1 a_1 + w_2 a_2 + \dots + w_n a_n &\geq 0 \end{aligned}$$

onde $w_0 = -\theta$ e $a_0 = 1$. Assim, para efeitos de cálculos em neurônios não lineares, pode-se sempre considerar versões estendidas $\vec{x} = [a_0, a_1, a_2, \dots, a_n]$ e $\vec{w} = [w_0, w_1, w_2, \dots, w_n]$ dos vetores de entrada e de pesos. A figura 3.8 na página 55 mostra a ampliação.

Um exemplo adicional de função de ativação não linear é $g : \mathbb{R} \longrightarrow \{-1, 1\}$, dada por

$$g(< \vec{w}, \vec{x} >) = \begin{cases} 1, & \text{se } w_1 a_1 + w_2 a_2 + \dots + w_n a_n \geq \theta \\ -1, & \text{se } w_1 a_1 + w_2 a_2 + \dots + w_n a_n < \theta \end{cases} \quad (3.10)$$

Uma função de ativação não linear particularmente importante é $g : \mathbb{R} \longrightarrow [0..1]$, definida por

$$g(< \vec{w}, \vec{x} >) = \frac{1}{1 + e^{-< \vec{w}, \vec{x} >}} \quad (3.11)$$

e denominada função logística. Seu gráfico é exibido na figura 3.9 na página 55.

Uma forma mais genérica da função logística é dada por

$$g(r) = \frac{1}{1 + e^{-r/T}} \quad (3.12)$$

onde o parâmetro T oferece curvas suaves a íngremes à medida que seu valor é decrementado. As figuras 3.10 na página 56 a 3.13 na página 57 mostram esse efeito.

Além das funções de ativação apresentadas, há diversos outros tipos. Alguns exemplos são mostrados da figura 3.14 na página 58 à figura 3.17 na página 59.

Duas características relevantes numa função de ativação $y = g(\langle \vec{w}, \vec{x} \rangle)$ são a preservação de monotonicidade dentro de uma faixa de valores para $\langle \vec{w}, \vec{x} \rangle$ e a saturação fora dessa faixa [Kov02]. Para ilustrar, na função de ativação linear exibida na figura 3.14 na página 58, apenas a monotonicidade é preservada, não ocorrendo saturação. Por outro lado, na função degrau, mostrada na figura 3.15 na página 58, apenas a saturação se preserva. Um exemplo de função que apresenta tanto preservação de monotonicidade quanto saturação é a função dada por

$$g(r) = \frac{r^k}{r^k + \alpha^k}, \quad r \geq 0$$

proposta por Siebert [Sie68] ao modelar neurônios biológicos no sistema auditivo de vertebrados e apresentada na figura 3.18 na página 60, com $k = 8$, $0 \leq r \leq 1$ e $\alpha = 1/2$.

Similarmente à função de Siebert, as funções de ativação logística (figura 3.9 na página 55) e tangente hiperbólica (figura 3.17 na página 59) também apresentam conservação de monotonicidade e de saturação. Por isso, embora ambas possuam as mesmas faixas de saturação das funções de ativação 3.9 na página anterior e 3.10 na página precedente, respectivamente, elas são preferíveis a estas em decorrência da monotonicidade. Além disso, possuem as características adicionais de serem simétricas e terem todas suas derivadas contínuas. Particularmente, a proposição a seguir mostra a derivada primeira da função de ativação logística.

Proposição 3.1 *Seja a função logística*

$$y = g(r) = \frac{1}{1 + e^{-r}}$$

Então, tem-se que

$$\frac{dy}{dr} = g(r) \cdot (1 - g(r))$$

Demonstração:

$$\frac{dy}{dr} = \frac{d}{dr} \left(\frac{1}{1 + e^{-r}} \right) = \frac{\frac{d}{dr}(1) \cdot (1 + e^{-r}) - 1 \cdot \frac{d}{dr}(1 + e^{-r})}{(1 + e^{-r})^2} =$$

$$\begin{aligned} \frac{-\frac{d}{dr}(e^{-r})}{(1+e^{-r})^2} &= \frac{e^{-r}}{(1+e^{-r})^2} = \frac{1}{1+e^{-r}} \cdot \frac{e^{-r}}{1+e^{-r}} = g(r) \cdot \frac{1+e^{-r}-1}{1+e^{-r}} = \\ &= g(r) \cdot \left(1 - \frac{1}{1+e^{-r}}\right) = g(r) \cdot (1 - g(r)) \end{aligned}$$

Uma rede neural artificial é um objeto composto por neurônios artificiais organizados sob uma determinada arquitetura. Essa arquitetura envolve número de camadas e conexão entre os neurônios, dentre outras características.

Numa RNA (rede neural artificial), pode-se considerar a existência de pelo menos duas camadas, a saber, de entrada e de saída. A camada de entrada é composta pelo vetor de entrada e a camada de saída por neurônios cujos valores de saída são os valores de saída da rede. Além dessas camadas, podem existir camadas extras entre a camada de entrada e a camada de saída. Estas camadas adicionais são denominadas camadas ocultas. A figura 3.19 na página 60 mostra um exemplo de rede neural artificial com três camadas: camada de entrada, uma camada oculta e camada de saída.

Uma vez que a camada de entrada é composta pelo vetor de entrada e as demais camadas compostas por neurônios, utiliza-se o termo *nodo* para designar genericamente tanto valores de entrada quanto neurônios.

Com relação à conexão entre os nodos da rede, as RNA's podem ser acíclicas ou cíclicas. A característica de uma rede cíclica é que a saída de algum nodo numa camada i é utilizada como entrada num outro nodo numa camada menor ou igual a i . Em contraposição, numa rede acíclica, dado qualquer nodo numa camada i , a saída desse nodo não é utilizada como entrada nos nodos das camadas menores que i . Na figura 3.19 na página 60 tem-se um exemplo de rede neural artificial acíclica.

Uma vez descritos os elementos básicos de uma rede neural artificial, uma indagação a ser feita é como se dá o processo de aprendizado. Em redes neurais artificiais, esse processo consiste em obter valores para os pesos $\vec{w} = [w_0, \dots, w_n]$ dos neurônios, satisfazendo uma determinada condição geral para a rede, geralmente a minimização de alguma função de erro. Assim, a hipótese numa rede neural é um conjunto de vetores de pesos.

A seguir é mostrado um procedimento de aprendizado de uma rede com um único neurônio artificial.

Seja $\mathcal{T} = \left\{ \left(\vec{x}_i, f(\vec{x}_i) \right) \right\}_{i=1}^m$ uma amostra de treinamento e $\vec{w} = [w_1, w_2, \dots, w_n]$ um vetor de pesos. O **erro de treinamento** de $\vec{w}$, relativo a amostra de treinamento $\mathcal{T}$, é definido pela função

$$E(\vec{w}) = \frac{1}{2} \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x}))^2 \quad (3.13)$$

onde $f(\vec{x})$ é o valor da função alvo para um exemplo $\vec{x}$ e $s(\vec{x})$ é a saída obtida para um exemplo $\vec{x}$.

Seja $\vec{w}$ um vetor de pesos e $E(\vec{w})$ uma função erro de treinamento relativa a uma amostra de treinamento $\mathcal{T}$. O gradiente de $E(\vec{w})$, denotado por $\nabla E(\vec{w})$, é o vetor definido por

$$\nabla E(\vec{w}) = \nabla E([w_0, w_1, \dots, w_n]) = \left[\frac{\partial E(\vec{w})}{\partial w_0}, \frac{\partial E(\vec{w})}{\partial w_1}, \dots, \frac{\partial E(\vec{w})}{\partial w_n} \right]$$

O gradiente é um vetor que especifica a direção em que há a maior taxa de crescimento de E . Dessa forma, a negação do gradiente $-\nabla E(\vec{w})$ indica a direção em que há maior taxa de decréscimo do erro de treinamento E .

A regra de treinamento para o gradiente descendente é dada por

$$w_i \leftarrow w_i + \Delta w_i$$

sendo

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

onde η é a taxa de aprendizado.

Essa regra mostra como cada passo descendente é realizado. A derivada parcial pode ser obtida da seguinte forma:

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x}))^2 \\ &= \frac{1}{2} \sum_{\vec{x} \in \mathcal{T}} \frac{\partial}{\partial w_i} (f(\vec{x}) - s(\vec{x}))^2 \\ &= \frac{1}{2} \sum_{\vec{x} \in \mathcal{T}} 2(f(\vec{x}) - s(\vec{x})) \frac{\partial}{\partial w_i} (f(\vec{x}) - s(\vec{x})) \\ &= \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x})) \frac{\partial}{\partial w_i} (f(\vec{x}) - \vec{w}\vec{x}) \\ \frac{\partial E}{\partial w_i} &= \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x})) (-a_{i\vec{x}}) \end{aligned}$$

onde $a_{i\vec{x}}$ denota o valor de entrada a_i para o exemplo $\vec{x}$.

Assim,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = -\eta \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x})) (-a_{i\vec{x}})$$

Isto é,

$$\Delta w_i = \eta \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x})) a_{i\vec{x}}$$

e

$$w_i = w_i + \eta \sum_{\vec{x} \in \mathcal{T}} (f(\vec{x}) - s(\vec{x})) a_{i\vec{x}} \quad (3.14)$$

Conforme a equação 3.14 na página anterior, a regra de treinamento por gradiente descendente atualiza os pesos a partir de uma soma de todos os exemplos. Uma alternativa com menor custo computacional é uma variante do gradiente descendente denominada regra delta. Nela, a atualização dos pesos é feita exemplo a exemplo, da seguinte maneira:

$$\Delta w_i = \eta(f(\vec{x}) - s(\vec{x}))a_i \quad (3.15)$$

onde $f(\vec{x})$ é o valor da função alvo, $s(\vec{x})$ é o valor de saída e a_i o i -ésimo valor de entrada do exemplo considerado.

A função de erro é definida para cada exemplo da seguinte forma:

$$E_{\vec{x}}(\vec{w}) = \frac{1}{2}(f(\vec{x}) - s(\vec{x}))^2 \quad (3.16)$$

onde $f(\vec{x})$ é o valor da função alvo e $s(\vec{x})$ é o valor de saída para o exemplo $\vec{x}$.

A regra delta oferece uma boa aproximação do descendente gradiente em relação ao erro original da equação 3.13 na página 46. Por isso, ela é também denominada gradiente descendente incremental. Outra vantagem é com relação à convergência. O gradiente descende aplicado a neurônios lineares é uma regra que sempre converge para um mínimo global. Contudo, em neurônios não lineares, esse método pode convergir para possíveis mínimos locais. Exemplos de um mínimo global e de um mínimo local são mostrados na figura 3.20 na página 61. O gradiente descendente incremental pode algumas vezes evitar a convergência em mínimos locais uma vez que utiliza vários $\nabla E_{\vec{x}}(\vec{w})$ (equação 3.16) no lugar de $\nabla E(\vec{w})$ (equação 3.13 na página 46).

Utilizando a regra gradiente descendente para atualização de pesos, redes neurais artificiais compostas por neurônios não lineares com função de ativação logística podem expressar diversas superfícies de decisão não lineares. Isso é possível através do algoritmo *backpropagation*. Uma vez que a rede pode possuir mais de um neurônio que gera valor de saída da rede, a função erro é dado pela soma dos erros sobre todas as unidades de saída da rede, conforme a equação

$$E(\vec{w}) = \frac{1}{2} \sum_{\vec{x} \in T} \sum_{k \in \text{saídas}} (f_k(\vec{x}) - s_k(\vec{x}))^2 \quad (3.17)$$

onde saídas é o conjunto de neurônios cujos valores de saída são os valores de saída da rede, $f_k(\vec{x})$ e $s_k(\vec{x})$ são o valor alvo e valor de saída associados ao k -ésimo neurônio e exemplo $\vec{x}$.

Uma versão do algoritmo backpropagation é dada na tabela 3.7 na próxima página. Essa versão aplica-se a redes acíclicas com três camadas (camada de entrada, uma camada oculta e camada de saída), onde os neurônios destas duas últimas camadas possuem função de ativação logística.

Existem várias formas para a determinação do momento em que o treinamento deve ser encerrado. Algumas condições de término são:

Backpropagation(exemplos_treinamento, η , $n_{entrada}$, n_{saida} , n_{oculta})

Cada exemplo de treinamento é um par da forma $(\vec{x}, f(\vec{x}))$, onde

$\vec{x}$ é o vetor de valores de entrada da rede e $f(\vec{x})$ é o vetor de valores de saída.

η é a taxa de aprendizado (por exemplo 0,05), $n_{entrada}$ é o número de entradas da rede, n_{oculta} é o número de neurônios da camada oculta e

n_{saida} é o número de neurônios de saída.

A entrada do nodo i no nodo j é denotada por x_{ji} e o peso do neurônio i no neurônio j é denotado por w_{ji} .

- Criar uma rede acíclica com $n_{entrada}$, n_{oculta} neurônios ocultos e n_{saida} neurônios de saída.
- Inicializar todos os pesos da rede com pequenos números aleatórios (ex., entre $-0,05$ e $0,05$).
- Até que a condição de término seja satisfeita faça
 - para cada $(\vec{x}, f(\vec{x}))$ nos exemplos de treinamento faça
 1. Entre com a instância $\vec{x}$ na rede e calcule a saída s_u do neurônio u na rede
 2. Para cada neurônio de saída k , calcule o seu termo de erro δ_k

$$\delta_k \leftarrow s_k(1 - s_k)(f_k - s_k)$$
 3. Para cada neurônio oculto o , calcule o seu termo de erro σ_o

$$\delta_o \leftarrow s_o(1 - s_o) \sum_{k \in \text{saidas}} w_{ko} \delta_k$$
 4. Atualize cada peso w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

onde

$$\Delta w_{ji} = \eta \delta_j a_{ji}$$

Tabela 3.7: Algoritmo Backpropagation (adaptado de [Mit97])

1. encerrar após um número máximo de iterações
2. encerrar quando $E(\vec{w})$ for menor que um determinado valor
3. encerrar quando a proporção de classificações corretas numa amostra de teste estiver acima de um determinado valor

Uma abordagem é adotar como critério de parada uma combinação de algumas dessas condições.

3.3.1 Variações da técnica

Embora o algoritmo *backpropagation* seja útil, algumas variantes são aplicadas para minimizar seus pontos frágeis.

Por exemplo, um dos problemas que pode ocorrer com o algoritmo *backpropagation* é uma situação de *overfitting*. Esse fenômeno acontece quando, após um número de iterações do treinamento, a taxa de acerto da rede para os exemplos de teste começa a piorar em vez de melhorar. Para minimizar o efeito, uma variante do algoritmo consiste em decrementar

cada peso por um pequeno valor, em cada iteração. Isso é feito penalizando, durante o treinamento, hipóteses que tenham valores altos de pesos. Formalmente, gera-se vetores de pesos de norma mínima, redefinindo a função erro (equação 3.17 na página 48) para incluir a penalidade. A equação 3.18 mostra a alteração.

$$E(\vec{w}) = \frac{1}{2} \sum_{\vec{x} \in \mathcal{T}} \sum_{k \in \text{saídas}} (f_k(\vec{x}) - s_k(\vec{x}))^2 + \frac{1}{2} \lambda \|\vec{w}\|^2 \quad (3.18)$$

O parâmetro λ dita a taxa de decremento. Quando ele é nulo, a função erro permanece na forma sem *weight decay*. Caso possua valor grande, a solução de pesos tende para $\vec{w} = \vec{0}$.

Outros dois problemas possíveis são lentidão no processo de treinamento e convergência para mínimo local. Para contorná-los, uma variante consiste em adicionar um termo *momentum* na atualização dos pesos. Esse termo faz com que a atualização dos pesos numa iteração t dependa parcialmente da atualização que ocorreu na iteração anterior $t - 1$. Formalmente, ele é definido por

$$\psi = \alpha(w_{ji}(t) - w_{ji}(t - 1)) \quad (3.19)$$

O ajuste de pesos do algoritmo *backpropagation* passa então a ser dado por

$$\Delta w_{ji} = \eta \delta_j a_{ji} + \alpha(w_{ji}(t) - w_{ji}(t - 1)) \quad (3.20)$$

Além dessas variações, outros algoritmos como Quickprop e Rprop foram propostos para acelerar o processo de treinamento e melhorar o desempenho na classificação.

3.4 Abordagem para análise de desempenho

Nesta seção são apresentados aspectos de análise de desempenho de classificadores, adotando-se um enfoque inferencial estatístico.

3.4.1 Análise de desempenho

Denomina-se **erro observado** do classificador $\hat{f} = \mathcal{A}(\mathcal{T})$ com relação a uma função alvo f e a uma amostra de treinamento $\mathcal{T}$, denotado por $erro_{\mathcal{T}}(\hat{f})$, a função definida por

$$erro_{\mathcal{T}}(\hat{f}) = \frac{1}{m} \sum_{i=1}^m \delta(f(x_i), \hat{f}(x_i))$$

onde m é o tamanho da amostra $\mathcal{T}$ e

$$\delta(f(x), \hat{f}(x)) = \begin{cases} 1, & \text{se } f(x) \neq \hat{f}(x) \\ 0, & \text{caso contrário} \end{cases}$$

Seja $\mathcal{D}$ uma distribuição de probabilidade. Denomina-se **erro verdadeiro** do classificador $\hat{f} = \mathcal{A}(\mathcal{T})$ com relação a uma função alvo f e a distribuição de probabilidade $\mathcal{D}$, denotado por $\text{erro}_{\mathcal{D}}(\hat{f})$, a função definida por

$$\text{erro}_{\mathcal{D}}(\hat{f}) = P(f(x) \neq \hat{f}(x)), \quad x \in \mathcal{D}$$

Esse é o erro esperado ao se aplicar o classificador $\hat{f}$ a outros exemplos, além dos exemplos da amostra de treinamento.

Analisar o desempenho de um classificador consiste em avaliar quão bem o erro observado a partir de uma amostra de teste estima o erro verdadeiro do classificador sobre o espaço de exemplos.

Sob o ponto de vista estatístico, o erro verdadeiro é um parâmetro e o erro observado é um estimador para o erro verdadeiro. Ambos são variáveis aleatórias e, conseqüentemente, possuem distribuições de probabilidade. A distribuição de probabilidade do erro observado é uma distribuição amostral.

O problema de estimar o erro verdadeiro dado o erro observado é o mesmo problema de estimar a proporção de uma população que contém uma característica, dada uma proporção observada de alguma amostra aleatória dessa população. No caso específico, a população é o espaço de exemplos e a característica é o exemplo ser classificado erroneamente pela hipótese. A proporção é o erro verdadeiro e a proporção observada é o erro observado. A amostra aleatória é a amostra de treinamento.

O erro observado, visto como uma proporção observada, segue o modelo binomial. A distribuição binominal é uma distribuição discreta de probabilidade. Ela se aplica sempre que o processo de amostragem é de Bernoulli. Um processo de Bernoulli é um processo de amostragem constituído por uma sequência de ensaios, denominados ensaios de Bernoulli, que possui as seguintes propriedades:

1. em cada ensaio de Bernoulli, existem dois resultados possíveis mutuamente exclusivos, denominados respectivamente de sucesso e fracasso;
2. os ensaios são constituídos de eventos independentes, ou seja, a probabilidade de um resultado em cada ensaio não depende dos resultados ocorridos nos ensaios anteriores, nem dos resultados nos ensaios posteriores;
3. o processo é estacionário, isto é, as probabilidades de sucesso e de fracasso permanecem constantes de ensaio para ensaio. Em particular, a probabilidade de ocorrer sucesso em cada ensaio é denotada por p e a de fracasso por $1 - p$.

A distribuição binominal é utilizada para determinar a probabilidade de se obter um dado número de sucessos em um processo de Bernoulli. Três valores são necessários: o número de sucessos, denotado por r , o número de ensaios, denotado por n , e a probabilidade de sucesso

em cada ensaio (p). A equação para se determinar a probabilidade de um certo número de sucessos r em uma distribuição binominal é dada por

$$P(X = r) = \binom{n}{r} \cdot p^r \cdot (1 - p)^{n-r} = \frac{n!}{r!(n-r)!} \cdot p^r \cdot (1 - p)^{n-r} \quad (3.21)$$

Uma variável aleatória que segue o modelo binomial pode ser expressa em termos de proporção de sucessos em lugar de número r de sucessos. Essa proporção, denotada por $\bar{p}$, é a razão do número de sucessos para o número de ensaios,

$$\bar{p} = \frac{r}{n} \quad (3.22)$$

Assim como na equação 3.21, a probabilidade de observar exatamente a proporção $\bar{p}$ de sucessos em n ensaios de Bernoulli é

$$P(X = \frac{r}{n}) = \binom{n}{r} \cdot p^r \cdot (1 - p)^{n-r} \quad (3.23)$$

Para medir a precisão de uma hipótese, pode-se estimar um intervalo de confiança para o erro verdadeiro $erro_D(\hat{f})$. Isso equivale a construir um intervalo de confiança para uma proporção.

Conforme foi visto, a distribuição de probabilidades aplicável a proporções é a distribuição de probabilidade binomial. Contudo, a construção de um intervalo de confiança para uma proporção populacional por meio da distribuição binomial pode ser bastante trabalhosa. Consequentemente, em geral se utiliza a distribuição normal como aproximação da distribuição binomial para a construção de intervalos de confiança para as proporções.

Assim, apesar de $erro_T(\hat{f})$ nem $erro_D(\hat{f})$ terem distribuições normais, o Teorema Central do Limite garante ser possível uma aproximação da distribuição binomial para a normal se o tamanho da amostra é grande. O erro observado $erro_T(\hat{f})$ é um estimador pontual para $erro_D(\hat{f})$. Um estimador por intervalo para $erro_D(\hat{f})$ pode ser dado por $erro_T(\hat{f})$ pelo seguinte intervalo de confiança:

$$IC(erro_D(\hat{f}), N\%) = erro_T(\hat{f}) \pm Z_N \cdot \sqrt{\frac{erro_D(\hat{f}) \cdot (1 - erro_D(\hat{f}))}{m}} \quad (3.24)$$

onde Z_N é uma constante para um coeficiente de confiança aproximadamente $N\%$.

Uma vez que na equação 3.24 o $erro_D(\hat{f})$ é desconhecido, há duas abordagens para estimá-lo.

A primeira é adotar $erro_T(\hat{f})$ como estimador para $erro_D(\hat{f})$. Assim, o intervalo de confiança para $erro_D(\hat{f})$ fica dado por

$$IC(erro_D(\hat{f}), N\%) = erro_T(\hat{f}) \pm Z_N \cdot \sqrt{\frac{erro_T(\hat{f}) \cdot (1 - erro_T(\hat{f}))}{m}} \quad (3.25)$$

Essa forma é denominada abordagem otimista, pois parte do princípio de que o estimador $erro_T(\hat{f})$ já está suficiente próximo de $erro_D(\hat{f})$, de modo que a variância $erro_D(\hat{f})(1 - erro_T(\hat{f}))/n$ é bem aproximada por $erro_T(\hat{f})(1 - erro_T(\hat{f}))/n$.

A outra maneira é adotar o valor de $p(1 - p)$ como sendo $1/4$. Esta abordagem é denominada conservativa e é possível devido a função $p(1 - p)$ assumir valor máximo de 0.25 , como mostra a figura 3.21 na página 62.

A partir desses conceitos teóricos contextualizados para análise de desempenho, é possível avaliar os classificadores gerados pelas técnicas apresentadas no capítulo anterior. Aspectos adicionais relacionados aos conceitos teóricos estão descritos no apêndice B. Como complemento, questões práticas de implementação precisam ser consideradas. Esse é o assunto do próximo capítulo.

3.4.2 Estratégias para partição dos dados

Há algumas estratégias para partição dos dados. A seguir, são descritas as estratégias $2/3 + 1/3$, *randomized*, *k-fold cross validation* e *leave-one-out*.

Na estratégia $2/3 + 1/3$, a amostra de treinamento é dividida em duas partes, contendo dois terços dos exemplos e um terço, respectivamente. Gera-se os classificadores com a primeira ($2/3$) e analisa-se o desempenho com a segunda ($1/3$).

Já na estratégia *randomized*, define-se uma quantidade fixa de exemplos para a análise de desempenho e extrai-se essa quantidade aleatoriamente da amostra de treinamento. Utiliza-se então o restante dos exemplos da amostra para a geração dos classificadores.

Na *k-fold cross validation*, a amostra de treinamento é dividida em k partes e são realizadas k etapas de geração/análise de desempenho. Em cada etapa, uma parte é utilizada para análise e as outras $k - 1$ partes são utilizadas para geração. Uma situação particular dessa estratégia é a *2-fold cross validation*. Nela, a amostra é dividida em duas partes, cada uma contendo a metade dos exemplos. Os classificadores são gerados com a primeira parte e analisados com a segunda. Depois o processo é invertido, gerando-se os classificadores com a segunda parte e analisando-se com a primeira. Além da *2-fold cross validation*, outra opção algumas vezes utilizada é a *10-fold cross validation*. Nela, a amostra é dividida em dez partes. Separa-se a primeira parte para análise e as outras nove para geração. Depois, separa-se a segunda parte para análise e as outras nove para geração e assim sucessivamente. Ao final, cada uma das dez partes foi utilizada para análise, utilizando-se as outras nove partes que sobram para geração.

Por fim, na estratégia *leave-one-out*, se a amostra de treinamento é composta por m exemplos, $\mathcal{T} = \left\{ \left(x_i, f(x_i) \right) \right\}_{i=1}^m$, então são realizadas m iterações de geração/análise de desempenho. Na primeira iteração, gera-se um classificador com os exemplos $x_2, \dots, x_m$ e seu desempenho é analisado com o exemplo x_1 . Na segunda iteração, gera-se um classificador com

os exemplos $x_1, x_3, \dots, x_m$ e analisa-se o desempenho com o exemplo x_2 . Na i -ésima iteração, gera-se um classificador com os exemplos $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m$ e avalia-se com o exemplo x_i . Na última iteração, gera-se o classificador com os exemplos $x_1, \dots, x_{m-1}$ e o desempenho é analisado com o exemplo x_m .

A partir desse leque de opções, foi adotada a estratégia de partição $2/3 + 1/3$ para os experimentos. Em cada um deles, a correspondente amostra foi dividida em duas partes, contendo dois terços dos exemplos e um terço, respectivamente. Os classificadores foram gerados com a primeira ($2/3$) parte e os desempenhos analisados com a segunda ($1/3$).

Figura 3.8: Neurônio Artificial com Valor de Limiar θ

Figura 3.9: Função Logística

Figura 3.11: Função Logística ($T = 1.5$)

Figura 3.10: Função Logística ($T = 2$)

Figura 3.13: Função Logística ($T = 1/2$)

Figura 3.12: Função Logística ($T = 1$)

Figura 3.14: Função de Ativação Linear

Figura 3.15: Função de Ativação Degrau

Figura 3.16: Função de Ativação Rampa

Figura 3.17: Função de Ativação Tangente Hiperbólica

Figura 3.18: Função de Siebert para $k = 8$, na faixa $0 \leq r \leq 1$, com $\alpha = 1/2$

Figura 3.19: Exemplo de Rede Neural Artificial

Figura 3.20: Mínimo Global e Mínimo Local de uma Função

Figura 3.21: Máximo da função $p(1-p) = 1/4$ quando $0 \leq p \leq 1$

Capítulo 4

Aspectos de implementação

Neste capítulo são apresentados aspectos de implementação, sobretudo a plataforma de software utilizada. É feita uma descrição sobre a plataforma R e de seus pacotes estreitamente relacionados com o trabalho: pacotes *class*, *rpart* e *nnet*.

Adicionalmente é descrito o pacote *linnaeus*, desenvolvido como produto parcial da pesquisa.

4.1 A plataforma R

R pode ser visto como um software livre, um ambiente para análise de dados, uma linguagem de programação ou uma distribuição, dentre outras perspectivas. Levando isso em conta, R será designada como plataforma R.

Em primeiro lugar, R é software livre, licenciado sob os termos da *GNU General Public License*, versão 2, de junho de 1991 [FSF]. Portanto, valem as liberdades contidas na tabela 4.1 a seguir.

1. liberdade para distribuir cópias do R;2. liberdade para receber ou obter o seu código fonte;3. liberdade para utilizá-lo;4. liberdade para alterar o código fonte ou inserir partes desse código em novos software livres.

Tabela 4.1: Liberdades de um Software Livre

Além de software livre, R é um ambiente para manipular dados, efetuar cálculos e exibir gráficos. O ambiente é interativo, oferecendo as seguintes funcionalidades, dentre outras:

manipulação e armazenamento de dados; possibilidade de cálculos com vetores e matrizes; funcionalidades para análise de dados; funcionalidades gráficas para exibição de dados. R oferece a implementação de diversas técnicas estatísticas clássicas e modernas. Essas técnicas estão no pacote base do ambiente e em outros pacotes. Modelagem linear e não linear, teste estatísticos, análise de séries temporais, classificação e *clustering* são alguns exemplos das técnicas implementadas. Por conseguinte, R pode ser visto como um ambiente para computação estatística.

Uma das essências da plataforma R é sua linguagem de programação. Nela, pode-se programar compondo unidades denominadas funções. Estas funções são escritas na linguagem específica da plataforma - linguagem R - podendo, contudo, fazer interfaces com programas escritos nas linguagens C, C++ e FORTRAN. A linguagem de programação R inclui condicionais, laços, funções recursivas definidas pelo usuário e funcionalidades de entrada e saída, dentre outras. Para a execução de programas, é oferecido um ambiente interpretado além da possibilidade de execução de programas armazenados em arquivos *scripts*. Adicionalmente, esse ambiente pode ser utilizado para desenvolvimento de software, estando disponível recursos como depurador, acesso a editores de texto e bibliotecas, dentre outras facilidades.

R também é uma distribuição. Uma distribuição consiste num conjunto de diversos módulos denominados pacotes. Uma distribuição R contém um pacote principal, denominado *base*, e pacotes adicionais, denominados pacotes padrões, que estão representados na figura 4.1 na página seguinte. Além desses pacotes, a distribuição traz um outro conjunto, denominados pacotes recomendados, tal como mostra a figura 4.2 na página 66. Como complemento aos pacotes padrões e recomendados, módulos adicionais estão disponíveis para propósitos específicos na CRAN - Comprehensive R Archive Network - uma coleção de sítios relacionados ao R, incluindo pacotes dos projetos Omegahat e Bioconductor.

Dentre os pacotes recomendados, destacam-se para o contexto deste trabalho os pacotes *class*, *rpart* e *nnet*, apresentados a seguir.

Figura 4.1: Pacotes Padrões de uma Distribuição R

Figura 4.2: Pacotes Recomendados da CRAN

4.2 O pacote class

Dentre os pacotes recomendados, *class* é um pacote para classificação. Ele contém funcionalidades relacionadas à técnica do vizinho mais próximo (VMP), além de outras técnicas de classificação, como *self-organizing maps* e *LVQ methods*, mas não consideradas neste trabalho.

As funções do pacote *class* referentes à técnica dos vizinhos mais próximos estão apresentadas na figura 4.3. Os valores de saída e alguns parâmetros de entrada dessas funções estão descritos brevemente na tabela 4.2 na próxima página à tabela 4.4 na página 70.

Figura 4.3: Funcionalidades de Vizinhos Mais Próximos do Pacote *class*

condense()	<p>Condensa uma amostra de treinamento para um classificador VMP-k.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. As linhas são os exemplos e as colunas são os valores de seus atributos. 2- um vetor com a classificação dos exemplos. 3- um vetor com os índices dos exemplos que devem compor a amostra condensada. Caso nenhum exemplo específico deva compor a amostra condensada, é escolhido um exemplo, aleatoriamente. <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com os índices dos exemplos da amostra condensada.
reduce.nn()	<p>Reduz uma amostra condensada para classificação VMP-k.</p> <p>Desconsidera da amostra condensada os exemplos que, ao serem retirados, não causem classificação errônea dos exemplos da amostra de treinamento.</p> <p>Utilizada após a função condense().</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. As linhas são os exemplos e as colunas são os valores de seus atributos. 2- um vetor com os índices dos exemplos que compõem uma amostra condensada. 3- um vetor com a classificação dos exemplos de treinamento. <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com os índices dos exemplos que compõem a amostra condensada reduzida.
knn1()	<p>Realiza classificação por VMP.</p> <p>Para cada exemplo de teste, é encontrado o exemplo mais próximo na amostra de treinamento. Sua classificação é usada para classificar o exemplo de teste.</p> <p>A métrica utilizada para calcular proximidade é a distância euclidiana.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. <p>As linhas são os exemplos e as colunas são os valores de seus atributos.</p> <ul style="list-style-type: none"> 2- uma matriz com os exemplos de teste. <p>As linhas são os exemplos e as colunas são os valores de seus atributos.</p> <ul style="list-style-type: none"> 3- um vetor com a classificação dos exemplos de treinamento. <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com a classificação dos exemplos de teste.

Tabela 4.2: Descrição das Funções de Vizinho Mais Próximo do Pacote *class*

knn()	<p>Realiza classificação por VMP-k.</p> <p>Para cada exemplo de teste, os k-ésimos exemplos mais próximos na amostra de treinamento são encontrados.</p> <p>A classificação do exemplo de teste é feita por meio de voto majoritário sobre a classificação dos k-ésimos exemplos de treinamento encontrados.</p> <p>A métrica utilizada para calcular proximidade é a distância euclidiana.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. <p>As linhas são os exemplos e as colunas são os valores de seus atributos.</p> <ol style="list-style-type: none"> 2- uma matriz com os exemplos de teste. <p>As linhas são os exemplos e as colunas são os valores de seus atributos.</p> <ol style="list-style-type: none"> 3- um vetor com a classificação dos exemplos de treinamento. 4- o número de vizinhos a serem considerados na classificação. 5- o número de votos mínimos exigidos para realizar uma classificação. 6- um valor lógico, indicando se todos os exemplos possivelmente empatados no k-ésimo maior vizinho devem ser ou não levados em consideração na classificação. <p>Caso nem todos devam ser considerados, um exemplo é escolhido aleatoriamente dentre eles, para que a classificação seja feita com exatos k-vizinhos.</p> <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com a classificação dos exemplos de teste.
knn.cv()	<p>Realiza classificação por VMP-k cruzada de uma amostra de treinamento.</p> <p>A classificação é do tipo <i>leave-one-out</i> onde, para cada exemplo, são encontrados, dentre os restantes, os k-ésimos exemplos mais próximos.</p> <p>A classificação é feita por voto majoritário sobre a classificação dos k-ésimos exemplos mais próximos.</p> <p>A métrica utilizada para calcular proximidade é a distância euclidiana.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. <p>As linhas são os exemplos e as colunas são os valores de seus atributos.</p> <ol style="list-style-type: none"> 2- um vetor com a classificação dos exemplos. 3- o número de vizinhos a serem considerados na classificação. 4- o número de votos mínimos exigidos para realizar uma classificação. 5- um valor lógico, indicando se todos os exemplos possivelmente empatados no k-ésimo maior vizinho devem ser ou não levados em consideração na classificação. <p>Caso nem todos devam ser considerados, um exemplo é escolhido aleatoriamente dentre eles, para que a classificação seja feita com exatos k-vizinhos.</p> <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com a classificação dos exemplos.

Tabela 4.3: Descrição das Funções de Vizinho Mais Próximo do Pacote *class* (cont.)

multiedit()	<p>Multiedita uma classificação VMP-k.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. 2- um vetor com a classificação dos exemplos. 3- o número de vizinhos utilizados na classificação VMP-k. 4- número de partes em que a amostra de treinamento deve ser dividida. <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor com os índices dos exemplos que compõem a amostra multieditada.
-------------	---

Tabela 4.4: Descrição das Funções de Vizinho Mais Próximo do Pacote *class* (cont.)

4.3 O pacote rpart

O pacote *rpart* contém as funções para árvores de decisão. Estas funções estão apresentadas na figura 4.4 na próxima página e descritas brevemente da tabela 4.5 na página 72 à tabela 4.7 na página 74.

Figura 4.4: Funcionalidades de Árvores de Decisão do Pacote *rpart*

<code>path.rpart()</code>	<p>Informa caminhos de nodos de uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- um vetor com os números dos nodos para os quais os caminhos são desejados. <p>Uma opção ao vetor é utilizar a função interativamente.</p> <p>Retorna:</p> <ul style="list-style-type: none"> - os caminhos desde a raiz aos nodos desejados.
<code>plot()</code>	<p>Exibe o dendograma de uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. <p>Retorna:</p> <ul style="list-style-type: none"> - uma lista com as coordenadas dos nodos.
<code>plotcp()</code>	<p>Oferece uma representação visual de resultados de validação cruzada.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão.
<code>post()</code>	<p>Gera o gráfico de uma árvore de decisão em formato <i>postscript</i>.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- um título para o gráfico. 3- um nome de arquivo para o gráfico. 4- opção geração no formato retrato ou paisagem.
<code>predict()</code>	<p>Realiza classificação por árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- valores que devem ser preditos. 3- o tipo de valor a ser retornado como predito. <p>No caso de uma árvore de decisão, o padrão é retornar uma matriz de probabilidades.</p> <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor contendo classificações.
<code>print()</code>	<p>Exibe uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> uma árvore de decisão.
<code>printcp()</code>	<p>Exibe uma tabela CP de uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão.

Tabela 4.5: Descrição das Funções de Árvores de Decisão do Pacote *rpart*

<code>prune()</code>	<p>Poda uma árvore de decisão.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma árvore de decisão. 2- um parâmetro de complexidade. <p>Retorna:</p> <ul style="list-style-type: none"> - uma nova árvore de decisão podada pelo parâmetro de complexidade.
<code>residuals()</code>	<p>Exibe resíduos de uma árvore de decisão.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma árvore de decisão. 2- o tipo de resíduo desejado (<i>usual</i>, <i>pearson</i>, <i>deviance</i>). <p>Retorna:</p> <ul style="list-style-type: none"> - um vetor de resíduos do tipo desejado.
<code>rpart()</code>	<p>Gera e retorna uma árvore de decisão.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma fórmula no formato “alvo atributos”. 2- o tipo de árvore a ser gerada. <p>Uma árvore de decisão é gerada com <code>method = “class”</code>.</p> <ol style="list-style-type: none"> 3- opção por entropia ou gini. <p>Retorna:</p> <ul style="list-style-type: none"> - uma árvore de decisão.
<code>rpart.control()</code>	<p>Controle para árvore de decisão.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- um número mínimo de exemplos que devem existir num nodo. 2- um número mínimo de exemplos em cada folha. 3- um parâmetro de complexidade. 4- o número variações a serem mostradas que foram alternativas à divisão efetivamente realizada. 5- número de validações cruzadas. 6- profundidade máxima possível de qualquer nodo da árvore. <p>Retorna:</p> <ul style="list-style-type: none"> - uma lista contendo as opções.
<code>rsq.rpart()</code>	<p>Exibe um gráfico contendo o erro relativo pelo número de divisões da árvore.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma árvore de decisão.

Tabela 4.6: Descrição das Funções de Árvores de Decisão do Pacote *rpart* (cont.)

snip()	<p>Elimina subárvores de uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- um vetor com os números dos nodos das subárvores a serem eliminadas. <p>Retorna:</p> <ul style="list-style-type: none"> - uma árvore de decisão contendo os nodos restantes após a eliminação de subárvores.
summary()	<p>Exibe informações sobre uma árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- parâmetro de complexidade mínimo dos nodos a serem exibidos.
text()	<p>Insere texto num dendograma de árvore de decisão.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma árvore de decisão. 2- um valor lógico indicando se as divisões da árvore devem ser rotuladas com os critérios de divisão. 3- os rótulos dos nodos. 2- um valor lógico indicando se todos os nós ou apenas as folhas devem ser rotuladas.
xpred()	<p>Realiza uma classificação por validação cruzada.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1. uma árvore de decisão. 2. número de grupos para validação cruzada. 3. lista de valores de complexidade desejados. <p>Retorna:</p> <p>uma matriz com uma linha para cada exemplo e uma coluna para cada valor de complexidade.</p>

Tabela 4.7: Descrição das Funções de Árvores de Decisão do Pacote *rpart* (cont.)

4.4 O pacote nnet

nnet é um pacote contendo funcionalidades para redes neurais artificiais acíclicas com uma única camada oculta. Além das funcionalidades para redes neurais, o pacote contém funções adicionais para modelos log-lineares multinomiais, que não são consideradas no contexto desse trabalho.

Uma rede neural com arquitetura acíclica (*feedforward*) é aquela onde a saída de um neurônio de uma i -ésima camada da rede não pode ser usada como entrada de neurônios em camadas de índice menor ou igual a i . Esta arquitetura se contrapõe a uma arquitetura cíclica (*feedback*), onde a saída de algum neurônio numa i -ésima camada da rede é usada como entrada de neurônios em camadas de índice menor ou igual a i .

O pacote *nnet* contém as funções para redes neurais artificiais apresentadas na figura 4.5 e descritas brevemente nas tabelas 4.8 na próxima página e 4.9 na página 77.

Figura 4.5: Funcionalidades de Redes Neurais do Pacote *nnet*

<code>class.ind()</code>	<p>Gera uma matriz de indicadores de categorias.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- um vetor de categorias. <p>Retorna:</p> <ul style="list-style-type: none"> - uma matriz com valores binários. <p>Cada linha corresponde a uma categoria.</p> <p>Em cada linha, os valores das colunas são zero, exceto para a coluna relativa à própria categoria, que possui valor 1.</p>
<code>coef.nnet()</code>	<p><code>coef.nnet()</code> é uma especialização da função genérica <code>coef()</code>.</p> <p><code>coef()</code> retorna os coeficientes de um dado modelo de objetos.</p> <p>Para o modelo de objetos redes neurais artificiais, <code>coef.nnet()</code> retorna os valores dos pesos da rede.</p>
<code>eval.nn()</code>	<p><code>eval()</code> é uma função genérica que avalia uma expressão.</p> <p><code>eval.nn()</code> especializa <code>eval()</code>, avaliando uma rede neural.</p> <p>Dada uma rede, <code>eval.nn()</code> retorna sua arquitetura, o número de pesos e parâmetros utilizados na construção da rede, com seus respectivos valores.</p>
<code>nnet()</code>	<p>Constrói uma rede neural artificial com uma única camada oculta.</p> <p>As unidades possuem função logística.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz com os exemplos de treinamento. 2- uma matriz com os valores alvos para os exemplos. 3- peso para cada exemplo de treinamento. 4- número de unidades na camada oculta. 5- opção por unidades logísticas ou unidades lineares. 6- intervalo de valores para os pesos iniciais. 7- valor para <i>weight decay</i>. 8- número máximo de iterações. 9- valor absoluto a ser adotado como critério de parada. 10- valor relativo a ser adotado como critério de parada. <p>Retorna:</p> <ul style="list-style-type: none"> - uma rede neural artificial com o melhor conjunto de pesos encontrado.
<code>nnetHess()</code>	<p>Dada uma rede neural, exemplos de treinamento e respectivos valores alvos, avalia o Hessiano (matriz de derivadas segunda) da rede neural especificada.</p>

Tabela 4.8: Descrição das Funções de Redes Neurais do Pacote *nnet*

<code>predict.nnet()</code>	<p>Prediz as categorias de exemplos de teste.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma rede neural. 2- uma matriz com exemplos de teste. <p>Retorna:</p> <ul style="list-style-type: none"> - a classificação dos exemplos de teste.
<code>print.nnet()</code>	<p><code>print.nnet()</code> é uma especialização da função genérica <code>print()</code>.</p> <p>Exibe a arquitetura da rede, o número de pesos e os parâmetros com respectivos valores utilizados na construção da rede.</p>
<code>residuals.nnet()</code>	<p><code>residuals()</code> é uma função genérica que retorna resíduos de modelos de objetos.</p> <p><code>residuals.nnet()</code> é uma especialização de <code>residuals()</code> para o modelo de redes neurais artificiais.</p>
<code>summary.nnet()</code>	<p><code>summary.nnet()</code> é uma especialização da função genérica <code>summary()</code>.</p> <p>Exibe a arquitetura da rede, a quantidade de pesos, parâmetros com respectivos valores utilizados na construção da rede e valores dos pesos da rede.</p>
<code>which.is.max()</code>	<p>Função acessória. Dado um vetor, retorna o índice do seu valor máximo.</p>

Tabela 4.9: Descrição das Funções de Redes Neurais do Pacote *nnet* (cont.)

4.5 O pacote linnaeus

Linnaeus foi um pacote desenvolvido com o objetivo primário de oferecer funcionalidades para análise de desempenho de classificadores. Sua estrutura é inspirada no pacote *Main Library of Venables and Ripley's MASS*, composto por bases de dados e funções desenvolvidas por Venables e Ripley em seus trabalhos de pesquisa.

O pacote linnaeus contém funções desenvolvidas no decorrer da pesquisa bem como bases de dados e *scripts* de experimentos. Esse pacote, juntamente com outros pacotes relacionados à classificação integram uma solução computacional que gera e seleciona classificadores a partir de uma análise estatística de desempenho. Similarmente aos pacotes quem compõem a plataforma R, linnaeus também é software livre.

Assim como as bases existentes no pacote MASS, no linnaeus há as bases de dados *adin*, *parlamentos* e *imunopediatria*. A primeira está relacionada à existência de ação de inconstitucionalidade sobre normas jurídicas. Estas normas são representadas por suas proposições legislativas, por sua vez caracterizadas por seu tipo, assunto, autoria e parecer. A segunda base de dados contém exemplos de parlamentos com seus respectivos portes institucionais. Os parlamentos são caracterizados pelo número de parlamentares, volume orçamentário e número de comissões permanentes. A terceira base contém alguns dados clínicos sobre pacientes imunodeficientes e não imunodeficientes do Ambulatório de Imunologia Pediátrica da Universidade Federal de São Paulo.

O pacote *linnaeus* contém funções para análise de desempenho de classificadores apresentadas na figura 4.6 na próxima página e descritas brevemente nas tabelas 4.10 na página 80 a 4.12 na página 82. Além dessas funções, outras foram desenvolvidas para dar suporte às funções principais. Entretanto, também podem ser úteis em outras situações. Por exemplo, dada uma sequência de nomes de variáveis $x_1, x_2, \dots, x_n, y$, uma função do pacote denominada `gera_formula` retorna um objeto $y \sim x_1 + x_2 + \dots + x_n$ da classe fórmula.

Por fim, o nome dado ao pacote é uma alusão à Carl Linnaeus (cientista sueco, 1707-1778). Linnaeus foi cientista, classificador e taxonomista. Ainda hoje, os animais e as plantas são classificados de acordo com suas idéias. Antes de Linnaeus, classificavam-se as “coisas vivas” em plantas e animais. Linnaeus avançou na idéia de classificação. Ela consistiu em criar categorias para animais ou plantas e classificá-los segundo gênero e espécie. Para ilustrar, um gato doméstico é classificado como *Felis domesticus*, um leão como *Felis leo* e um cachorro como *Canis familiaris*. Esse sistema de classificação atualmente é conhecido como sistema binomial de Linnaeus.

Figura 4.6: Funcionalidades do Pacote *linnaeus*

<code>desempenho.classificador()</code>	<p>Gera um resultado de análise de desempenho de um classificador.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- número de erros de um classificador. 2- número de exemplos utilizados no teste. 3- nível de confiança para intervalo de confiança. <p>Retorna:</p> <ul style="list-style-type: none"> - lista contendo: <ul style="list-style-type: none"> . erro observado, . desvio padrão do erro observado, . intervalo de confiança aproximado otimista e . intervalo de confiança aproximado conservativo.
<code>desempenho.tecnicas()</code>	<p>Obtém um resultado de análise de desempenho de técnicas de geração de classificadores.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- vetor de erros observados da técnica 1. 2- vetor de erros observados da técnica 2. 3- nível de confiança para intervalo de confiança. <p>Retorna:</p> <ul style="list-style-type: none"> - lista contendo: <ul style="list-style-type: none"> . número de amostras de testes, . média estimada da diferença de erro entre as técnicas de geração, . desvio padrão da média estimada da diferença de erro, . graus de liberdade, . intervalo de confiança para a média estimada da diferença de erro e . nível de confiança do intervalo de confiança.
<code>gera.classificadores()</code>	<p>Gera diversos classificadores por vizinhos mais próximos, árvores de decisão e redes neurais.</p> <p>Recebe:</p> <ol style="list-style-type: none"> 1- uma amostra de treinamento/testes. 2- número de partes para k-fold cross validation. 3- limites inferiores das partes. 4- limites superiores das partes. 5- nível de confiança para intervalo de confiança. <p>Retorna:</p> <ul style="list-style-type: none"> - uma lista de classificadores. <p>Para cada classificador são oferecidas as seguintes informações:</p> <ul style="list-style-type: none"> . agente computacional (o “classificador”) . número de erros do classificador . erro observado do classificador . desvio padrão do erro observado . intervalo de confiança para o erro verdadeiro . nível de confiança do intervalo

Tabela 4.10: Descrição das Funções do Pacote *linnaeus*

<code>gera.classificadores.2()</code>	<p>Gera diversos classificadores por vizinhos mais próximos, árvores de decisão e redes neurais.</p> <p>Recebe:</p> <p>1- uma amostra de treinamento/testes.</p> <p>Retorna:</p> <p>- uma lista de classificadores.</p> <p>Para cada classificador a função oferece as seguintes informações:</p> <ul style="list-style-type: none"> . agente computacional (o “classificador”) . número de erros do classificador . erro observado do classificador . desvio padrão do erro observado . intervalo de confiança para o erro verdadeiro . nível de confiança do intervalo
<code>melhores.classificadores()</code>	<p>Seleciona os melhores classificadores para um problema de classificação.</p> <p>Recebe:</p> <p>1. lista de classificadores.</p> <p>Retorna:</p> <p>- lista contendo os classificadores com menor erro observado.</p>
<code>melhor.classificador()</code>	<p>Seleciona o melhor classificador para um problema de classificação.</p> <p>Recebe:</p> <p>1. lista de classificadores.</p> <p>Retorna:</p> <p>- o classificador com menor erro observado.</p> <p>Em caso de empate, retorna o primeiro melhor classificador.</p>
<code>numero.erros.ad()</code>	<p>Calcula o número de erros de um classificador por árvore de decisão.</p> <p>Recebe:</p> <p>1- vetor com categorias preditas.</p> <p>2- vetor com categorias alvos.</p> <p>Retorna:</p> <p>- diferença entre categorias preditas e alvos.</p>
<code>numero.erros.rna()</code>	<p>Calcula o número de erros de um classificador por rede neural artificial.</p> <p>Recebe:</p> <p>1- vetor com categorias preditas.</p> <p>2- vetor com categorias alvos.</p> <p>Retorna:</p> <p>- diferença entre categorias preditas e alvos.</p>

Tabela 4.11: Descrição das Funções do Pacote *linnaeus* (cont.)

<code>numero.erros.vmp()</code>	<p>Calcula o número de erros de um classificador por vizinho mais próximo.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- vetor com categorias preditas. 2- vetor com categorias alvos. <p>Retorna:</p> <ul style="list-style-type: none"> - diferença entre categorias preditas e alvos.
<code>piores.classificadores()</code>	<p>Seleciona os piores classificadores para um problema de classificação.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1. lista de classificadores. <p>Retorna:</p> <ul style="list-style-type: none"> - lista contendo os classificadores com maior erro observado.
<code>pior.classificador()</code>	<p>Seleciona o pior classificador para um problema de classificação.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1. lista de classificadores. <p>Retorna:</p> <ul style="list-style-type: none"> - o classificador com maior erro observado. <p>Em caso de empate, retorna o primeiro pior classificador.</p>
<code>traco()</code>	<p>Calcula o traço de uma matriz.</p> <p>Recebe:</p> <ul style="list-style-type: none"> 1- uma matriz. <p>Retorna:</p> <ul style="list-style-type: none"> - traço da matriz. <p>O traço de uma matriz é a soma dos elementos de sua diagonal principal. A diagonal principal de uma matriz $[a_{ij}]$ é o conjunto dos elementos tais que $i = j$.</p>

Tabela 4.12: Descrição das Funções do Pacote *linnaeus* (cont.)

Capítulo 5

Experimentos e Resultados

Este capítulo apresenta resultados de experimentos sobre geração e análise de desempenho de diversos classificadores, utilizando as técnicas vizinhos mais próximos, árvores de decisão e redes neurais artificiais. A primeira seção descreve os classificadores por meio de seus parâmetros de configuração. A seção seguinte discute brevemente estratégias para partição dos dados. As demais seções mostram os resultados de três experimentos de geração e análise de desempenho de classificadores em bases de dados distintas.

5.1 Classificadores

As tabelas a seguir apresentam os parâmetros dos classificadores gerados nos experimentos. Para a técnica dos vizinhos mais próximos, foram gerados classificadores variando-se o número de vizinhos ($k = 1, 3, 5, 7, 9, 11$ e 12).

Classificadores gerados por redes neurais artificiais consistiram em redes acíclicas de 3 (três) camadas, com 2 (dois) entradas e 2 (duas) saídas.

Para a técnica de árvores de decisão foram gerados classificadores por ganho de informação e gini.

Número de vizinhos considerados: 1 (um)

Votos mínimos necessários para tomar decisão de classificar: 0 (zero)

Tratamento de empates no 1-ésimo vizinho: é feita a escolha de um vizinho, aleatoriamente

Tabela 5.1: Especificação dos Parâmetros para VMP_1

<i>Número de vizinhos considerados: 3 (três)</i>
<i>Votos mínimos necessários para tomar de decisão de classificar: 0 (zero)</i>
<i>Tratamento de empates no k-ésimo maior vizinho</i>
Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.2: Especificação dos Parâmetros para VMP_2

<i>Número de vizinhos considerados: 5 (cinco)</i>
<i>Votos mínimos necessários para tomar de decisão de classificar: 0 (zero)</i>
<i>Tratamento de empates no k-ésimo maior vizinho</i>
Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.3: Especificação dos Parâmetros para VMP_3

<i>Número de vizinhos considerados: 7 (sete)</i>
<i>Votos mínimos necessários para tomar de decisão de classificar: 0 (zero)</i>
<i>Tratamento de empates no k-ésimo maior vizinho</i>
Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.4: Especificação dos Parâmetros para VMP_4

<i>Número de vizinhos considerados: 9 (nove)</i>
<i>Votos mínimos necessários para tomar de decisão de classificar: 0 (zero)</i>
<i>Tratamento de empates no k-ésimo maior vizinho</i>
Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.5: Especificação dos Parâmetros para VMP_5

<i>Número de vizinhos considerados: 11 (onze)</i>
<i>Tratamento de empates no k-ésimo maior vizinho</i>
Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.6: Especificação dos Parâmetros para VMP_6

<i>Número de vizinhos considerados:</i> 12 (doze)
<i>Votos mínimos necessários para tomar de decisão de classificar:</i> 0 (zero)
<i>Tratamento de empates no k-ésimo maior vizinho</i> Incluir todos os k-ésimos maiores vizinhos possivelmente empatados.

Tabela 5.7: Especificação dos Parâmetros para VMP_7

<i>Número de neurônios em cada camada:</i> 2-1-2
<i>Função de ativação:</i> logística
<i>Algoritmo de aprendizado:</i> error backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.8: Descrição da Rede Neural Artificial RNA_1

<i>Número de neurônios em cada camada:</i> 2-2-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.9: Descrição da Rede Neural Artificial RNA_2

<i>Número de neurônios em cada camada:</i> 2-3-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.10: Descrição da Rede Neural Artificial RNA_3

<i>Número de neurônios em cada camada:</i> 2-4-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.11: Descrição da Rede Neural Artificial RNA_4

<i>Número de neurônios em cada camada:</i> 2-2-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 100 (cem)

Tabela 5.12: Descrição da Rede Neural Artificial RNA_5

<i>Número de neurônios em cada camada:</i> 2-2-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation
<i>Valor do weight decay:</i> 0
<i>Número máximo de épocas de treinamento:</i> 100 (cem)

Tabela 5.13: Descrição da Rede Neural Artificial RNA_6

<i>Número de neurônios em cada camada:</i> 2-10-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.14: Descrição da Rede Neural Artificial RNA_7

<i>Número de neurônios em cada camada:</i> 2-100-2
<i>Função de ativação:</i> Logística
<i>Algoritmo de aprendizado:</i> Error Backpropagation com weight decay
<i>Valor do weight decay:</i> 5e-04
<i>Número máximo de épocas de treinamento:</i> 200 (duzentas)

Tabela 5.15: Descrição da Rede Neural Artificial RNA_8

Número de neurônios em cada camada: 2-3-2

Função de ativação: Logística

Algoritmo de aprendizado: Error Backpropagation

Valor do weight decay: 0

Número máximo de épocas de treinamento: 200 (duzentas)

Tabela 5.16: Descrição da Rede Neural Artificial RNA_9

Número de neurônios em cada camada: 2-3-2

Função de ativação: Logística

Algoritmo de aprendizado: Error Backpropagation com weight decay

Valor do weight decay: 5e-04

Número máximo de épocas de treinamento: 100 (cem)

Tabela 5.17: Descrição da Rede Neural Artificial RNA_{10}

<i>Medida de homogeneidade dos exemplos:</i> ganho de informação
<i>Podação da árvore:</i> não
<i>Custos diferentes para atributos:</i> não

Tabela 5.18: Parâmetros para a Árvore de Decisão AD_1

<i>Medida de homogeneidade dos exemplos:</i> gini
<i>Podação da árvore:</i> não
<i>Custos diferentes para atributos:</i> não

Tabela 5.19: Parâmetros para a Árvore de Decisão AD_2

5.2 Experimento 1

<i>Origem dos dados</i>
Bases de dados synth.tr e synth.te, contidas no pacote <i>MASS</i> da plataforma R.
<i>Número de exemplos</i>
1250 (mil duzentos e cinquenta) exemplos.
<i>Atributos</i>
Cada exemplo é definido por 2 (dois) atributos: coordenada x e coordenada y. Os atributos são contínuos.
<i>Categorias</i>
As instâncias são classificadas em uma de duas categorias, codificadas como 0 e 1.
<i>Distribuição dos exemplos nas categorias</i>
Os exemplos estão distribuídos nas categorias na proporção 625:625.

Tabela 5.20: Experimento 1 - Descrição da Base de Dados *synth*

A tabela 5.21 na página seguinte mostra o desempenho dos classificadores gerados para a base descrita na tabela 5.20. Nela, as seguintes convenções foram utilizadas:

Número de erros - erros de classificação da amostra de teste

$erro_T(\hat{f})$ - número de erros / número de exemplos da amostra de teste

DP_1 - Desvio padrão otimista

IC_1 - Intervalo de confiança otimista

DP_2 - Desvio padrão conservativo

IC_2 - Intervalo de confiança conservativo

Observando a tabela 5.21, nota-se que o classificador com melhor desempenho foi o VMP_4 .

Classificador	Número de erros	$erro_T(\hat{f})$	DP_1	IC_1	DP_2	IC_2
VMP_1	83	0,133	0,014	[0,106;0,159]	0,02	[0,094;0,172]
VMP_2	66	0,106	0,012	[0,082;0,130]	0,02	[0,066;0,145]
VMP_3	59	0,094	0,012	[0,071;0,117]	0,02	[0,055;0,134]
VMP_4	53	0,085	0,011	[0,063;0,107]	0,02	[0,046;0,124]
VMP_5	56	0,090	0,011	[0,067;0,112]	0,02	[0,050;0,129]
VMP_6	56	0,090	0,011	[0,067;0,112]	0,02	[0,050;0,129]
VMP_7	60	0,096	0,012	[0,073;0,119]	0,02	[0,057;0,135]
RNA_1	80	0,128	0,013	[0,102;0,154]	0,02	[0,089;0,167]
RNA_2	64	0,102	0,012	[0,079;0,126]	0,02	[0,063;0,142]
RNA_3	57	0,091	0,012	[0,069;0,114]	0,02	[0,052;0,130]
RNA_4	62	0,099	0,012	[0,076;0,123]	0,02	[0,060;0,138]
RNA_5	64	0,102	0,12	[0,079;0,126]	0,02	[0,063;0,142]
RNA_6	80	0,128	0,013	[0,102;0,154]	0,02	[0,089;0,167]
RNA_7	63	0,101	0,012	[0,077;0,124]	0,02	[0,062;0,140]
RNA_8	65	0,104	0,012	[0,080;0,128]	0,02	[0,065;0,143]
RNA_9	66	0,106	0,012	[0,082;0,123]	0,02	[0,066;0,145]
RNA_{10}	58	0,093	0,012	[0,070;0,116]	0,02	[0,054;0,132]
AD_1	88	0,141	0,014	[0,114;0,168]	0,02	[0,102;0,180]
AD_2	71	0,114	0,013	[0,089;0,138]	0,02	[0,074;0,153]

Tabela 5.21: Experimento 1 - Análise de Desempenho de Classificadores para *synth*

5.3 Experimento 2

<i>Origem dos dados</i>
Base de dados iris, coletada por Anderson [And35] e contida no pacote <i>base</i> da plataforma R. A base de dados apresenta a medida em centímetros do tamanho e largura de sépalas (peça do cálice da flor) e pétalas de 50 (cinquenta) flores de cada uma das 3 (três) espécies de íris (planta da família das Iridáceas).
<i>Número de exemplos</i>
150 (cento e cinquenta) exemplos.
<i>Atributos</i>
Cada exemplo é definido por 4 (quatro) atributos: Sepal.Length, Sepal.Width, Petal.Length e Petal.Width.
<i>Categorias</i>
As instâncias são classificadas em 3 (três) categorias, a saber: <i>setosa</i> , <i>versicolor</i> e <i>virginica</i> .
<i>Distribuição dos exemplos nas categorias</i>
Os exemplos estão distribuídos nas categorias na proporção 50:50:50.

Tabela 5.22: Experimento 2 - Descrição da Base de Dados *iris*

Considerando a base de dados descrita na tabela 5.22, a tabela 5.23 na próxima página mostra diversos classificadores que obtiveram melhor desempenho. Especificamente, os classificadores VMP_2 , VMP_3 , VMP_5 e VMP_6 e as redes neurais artificiais RNA_1 a RNA_6 , RNA_8 e RNA_{10} .

Classificador	Número de erros	$erro_T(\hat{f})$	DP_1	IC_1	DP_2	IC_2
VMP_1	4	0,053	0,026	[0,002;0,104]	0,058	[-0,060;0,166]
VMP_2	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
VMP_3	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
VMP_4	4	0,053	0,026	[0,002;0,104]	0,058	[-0,060;0,166]
VMP_5	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
VMP_6	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
VMP_7	4	0,053	0,026	[0,002;0,104]	0,058	[-0,060;0,166]
RNA_1	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_2	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_3	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_4	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_5	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_6	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_7	8	0,107	0,036	[0,037;0,177]	0,058	[-0,006;0,220]
RNA_8	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
RNA_9	30	0,400	0,057	[0,289;0,511]	0,058	[0,287;0,513]
RNA_{10}	3	0,040	0,023	[-0,004;0,084]	0,058	[-0,073;0,153]
AD_1	4	0,053	0,026	[0,002;0,104]	0,058	[-0,060;0,166]
AD_2	4	0,053	0,026	[0,002;0,104]	0,058	[-0,060;0,166]

Tabela 5.23: Experimento 2 - Análise de Desempenho de Classificadores para *iris*

5.4 Experimento 3

<i>Origem dos dados</i>
Base de dados Imunopediatria. Essa base contém dados clínicos de pacientes do Ambulatório de Imunopediatria da UNIFESP/EPM - Universidade Federal de São Paulo / Escola Paulista de Medicina [Tav04]. Foram extraídos da base o número de internações, episódios de otites, episódios de sinusites e episódios de pneumonias de 126 pacientes imunocompetentes e 132 pacientes imunodeficientes.
<i>Número de exemplos</i>
258 (duzentos e cinquenta e oito) exemplos.
<i>Atributos</i>
Cada exemplo é definido por 4 (quatro) atributos: Internações, Otites, Sinusites e Pneumonias.
<i>Categorias</i>
As instâncias são classificadas em 2 (duas) categorias, a saber: imunocompetente e imunodeficiente.
<i>Distribuição dos exemplos nas categorias</i>
Os exemplos estão distribuídos nas categorias na proporção 126:132.

Tabela 5.24: Experimento 3 - Descrição da Base de Dados *imunopediatria*

Observando a tabela 5.25 na página seguinte, percebe-se que o classificador com melhor desempenho foi o VMP_5 .

Classificador	número de erros	$erro_T(\hat{f})$	DP_1	IC_1	DP_2	IC_2
VMP_1	59	0,457	0,044	[0,371;0,543]	0,044	[0,371;0,544]
VMP_2	55	0,426	0,044	[0,341;0,512]	0,044	[0,340;0,513]
VMP_3	58	0,450	0,044	[0,364;0,535]	0,044	[0,363;0,536]
VMP_4	53	0,411	0,043	[0,326;0,496]	0,044	[0,325;0,497]
VMP_5	51	0,395	0,043	[0,311;0,480]	0,044	[0,309;0,482]
VMP_6	58	0,450	0,044	[0,364;0,535]	0,044	[0,363;0,536]
VMP_7	56	0,434	0,044	[0,349;0,520]	0,044	[0,348;0,520]
RNA_1	64	0,496	0,044	[0,410;0,582]	0,044	[0,410;0,582]
RNA_2	60	0,465	0,044	[0,379;0,551]	0,044	[0,379;0,551]
RNA_3	67	0,519	0,044	[0,433;0,606]	0,044	[0,433;0,606]
RNA_4	62	0,481	0,044	[0,394;0,567]	0,044	[0,394;0,567]
RNA_5	57	0,442	0,044	[0,356;0,528]	0,044	[0,356;0,528]
RNA_6	61	0,473	0,044	[0,387;0,559]	0,044	[0,387;0,559]
RNA_7	61	0,473	0,044	[0,387;0,559]	0,044	[0,387;0,559]
RNA_8	60	0,465	0,044	[0,379;0,551]	0,044	[0,379;0,551]
RNA_9	70	0,543	0,044	[0,457;0,629]	0,044	[0,456;0,629]
RNA_{10}	63	0,488	0,044	[0,402;0,575]	0,044	[0,402;0,575]
AD_1	62	0,481	0,044	[0,394;0,567]	0,044	[0,394;0,567]
AD_2	66	0,512	0,044	[0,425;0,598]	0,044	[0,425;0,598]

Tabela 5.25: Experimento 3 - Análise de Desempenho de Classificadores para *imunopediatria*

Capítulo 6

Conclusão

6.1 Um apanhado geral

Neste trabalho, estudou-se sobre classificadores com ênfase em técnicas para geração a partir de exemplos de treinamento e análise de desempenho. Descreveu-se conceitos básicos e um processo para geração de classificadores. Utilizou-se uma abordagem estatística para análise de desempenho dos classificadores gerados. Fez-se experimentos utilizando essa abordagem. Além disso, avaliou-se uma solução de software que possibilite a integração das técnicas existentes e as que venham a ser desenvolvidas. Como complemento à solução, implementou-se um pacote para análise de desempenho que permite a seleção automática dos melhores classificadores gerados.

As três técnicas integradas e testadas foram vizinhos mais próximos, árvores de decisão e redes neurais artificiais com algoritmo *backpropagation*.

A plataforma R foi estudada e se mostrou solução como ambiente para integração de diversas técnicas para geração de classificadores. Dentre outras, características como software livre, ambiente interativo com diversas funcionalidades, linguagem de programação e distribuição com diversos pacotes tornam R especialmente atrativo para esse objetivo.

Funcionalidades de classificação são encontradas nos pacotes *class*, *rpart* e *nnet* da distribuição oficial R. Adicionalmente, o pacote *linnaeus* insere funcionalidades voltadas para análise de desempenho de classificadores. Testes e experimentos foram feitos para verificar a solução integrada.

6.2 Considerações finais

A partir desse estudo, surgiram idéias adicionais que podem vir a se tornar acréscimos ao que foi proposto. Tais idéias são brevemente descritas a seguir.

Um complemento sobre técnicas de classificação em geral é que, além das estudadas, há diversas outras como abordagem bayesiana, por mínima distância, funções de decisão

lineares e funções de decisão lineares generalizadas, dentre outras. Tais técnicas podem ser implementadas na plataforma R. Adicionalmente, podem ser implementadas técnicas de redução de dimensionalidade.

Extensões do pacote *linnaeus* são um passo futuro desta pesquisa. Exemplos são inserção de outras medidas de desempenho de classificadores, medidas de desempenho de técnicas de classificação, opções de escolha das estratégias de partição dos dados etc.

Finalmente e talvez num sentido mais amplo, ao passar da idade contemporânea para a atual idade da informação, muitas organizações vêm se deparando com a necessidade de obter novas perspectivas e visões para cumprir melhor seus objetivos. Dentre essas organizações, estão as instituições do poder legislativo brasileiro. Nelas, um aspecto dos mais importantes é o processo de tomada de decisão. A idéia de utilizar classificadores para dar suporte a esse processo se mostra bastante promissora, podendo se tornar objetivo de pesquisas futuras. Aspectos práticos disso podem envolver a criação de bases públicas de dados legislativos e a modelagem de problemas como, por exemplo, o problema do controle de constitucionalidade de normas jurídicas.

Apêndice A

Exemplos de dados dos experimentos

xs	ys	yc
0.05100797	0.16086164	0
-0.59823393	0.98701425	1
-0.74807425	0.08904024	0
-0.20194736	0.62101680	1
-0.77293371	0.26317168	0
0.47146103	0.48221146	1
0.21837360	0.12706142	0
-0.09821987	0.58755577	1
0.37268336	0.49656200	0
-0.35657658	0.63709705	1
-0.62931544	0.63202159	0
0.63881392	0.42112135	1
-0.43307167	0.14479166	0
0.62980614	0.28146085	1
-0.84151970	-0.19131316	0
-0.46223286	0.61661031	1
0.47525648	0.22483671	0
-0.07331555	0.55821736	1
0.32082976	0.32721288	0

Tabela A.1: Experimento 1 - Alguns Exemplos da Base de Dados *synth*

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	7.0	3.2	4.7	1.4	versicolor
10	6.4	3.2	4.5	1.5	versicolor
11	6.9	3.1	4.9	1.5	versicolor
12	5.5	2.3	4.0	1.3	versicolor
13	6.5	2.8	4.6	1.5	versicolor
14	5.7	2.8	4.5	1.3	versicolor
15	6.3	3.3	4.7	1.6	versicolor
16	4.9	2.4	3.3	1.0	versicolor
17	6.3	3.3	6.0	2.5	virginica
18	5.8	2.7	5.1	1.9	virginica
19	7.1	3.0	5.9	2.1	virginica
20	6.3	2.9	5.6	1.8	virginica
21	6.5	3.0	5.8	2.2	virginica
22	7.6	3.0	6.6	2.1	virginica
23	4.9	2.5	4.5	1.7	virginica
24	7.3	2.9	6.3	1.8	virginica
25	6.7	2.5	5.8	1.8	virginica

Tabela A.2: Experimento 2 - Alguns Exemplos da Base de Dados *iris*

Figura A.1: Experimento 2 - Plotagens e Histogramas dos Atributos de *íris*

Figura A.2: Experimento 2 - Plotagens e Histogramas dos Atributos de *íris* (cont.)

Figura A.3: Experimento 2 - Diagrama em Caixas dos Atributos de *íris*

Figura A.4: Experimento 2 - Atributos da Base de Dados *íris*

Figura A.5: Experimento 3 - Plotagens e Histogramas dos Atributos

	Internações	Otites	Sinusites	Pneumonias	Imunodeficiência
1	1	0	1	1	d
2	1	0	11	1	d
3	0	6	0	0	d
4	1	11	0	2	d
5	3	4	0	0	d
6	0	3	3	0	d
7	15	11	0	11	d
8	3	1	0	0	d
9	0	1	0	1	d
10	1	11	2	0	d
11	0	0	11	0	d
12	3	3	0	1	d
13	4	11	0	0	c
14	3	1	0	3	c
15	9	1	1	3	c
16	3	5	0	2	c
17	0	11	0	0	c
18	0	0	11	1	c
19	2	0	0	2	c
20	0	1	0	0	c
21	2	11	0	11	c
22	3	5	11	0	c
23	0	0	1	6	c
24	2	4	1	0	c
25	0	11	3	0	c

Tabela A.3: Experimento 3 - Alguns Exemplos da Base de Dados *imunopediatria*

Figura A.6: Experimento 3 - Plotagens e Histogramas dos Atributos (cont.)

Figura A.7: Experimento 3 - Diagrama em Caixas dos Atributos

Figura A.8: Experimento 3 - Relação entre os Atributos

Figura A.9: Experimento 1 - Diagrama em Caixas de VMP versus RNA sobre *synth*

Figura A.10: Experimento 2 - Diagrama em Caixas de VMP versus RNA sobre *íris*

Referências Bibliográficas

- [AB92] M. Anthony and N. Biggs. *Computational Learning Theory - An Introduction*. Cambridge University Press, 1992.
- [AFV94] M. Adams, C. Fields, and J. Venter, editors. *Automated DNA Sequencing and Analysis*. Academic Press, 1994.
- [AGK98] K. Al-Ghoneim and B. Kumar. Unified Decision Combination Framework. *Pattern Recognition*, 31(12):2077–2089, December 1998.
- [Alp98] E. Alpaydin. Techniques for Combining Multiple Learners. In *Engineering of Intelligent Systems EIS'98*, volume 2, pages 6–12, Tenerife, Spain, February 1998.
- [And35] E. Anderson. The irises of the Gaspé Peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.
- [BB91] G. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM. J. Appl. Math.*, 51:1782–1792, December 1991.
- [BDT97] J. Barrera, E. Dougherty, and N. Tomita. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.
- [BO98] A. Baxevanis and B. Ouellette, editors. *Bioinformatics - A Practical Guide to the Analysis of Genes and Proteins*. John Wiley, 1998.
- [Bru03] M. Brun. *Projeto de Operadores Morfológicos Multi-Escala por Otimização Estatística*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brasil, 2003.

- [BTHH00] J. Barrera, R. Terada, R. Hirata, and N. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.
- [CM98] V. Cherkassky and F. Mulier. *Learning from Data – Concepts, Theory and Methods*. John Wiley & Sons, 1998.
- [Dag95] J. Daghljan. *Lógica e Álgebra de Boole*. Atlas, São Paulo, fourth edition, 1995.
- [Dan00] C. Dantas. *Probabilidade: Um Curso Introductório*. EDUSP, São Paulo, second edition, 2000. ISBN 85-314-0399-5.
- [Die00] T. Dietterich. Ensemble Methods in Machine Learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems: First International Workshop, MCS 2000*, volume LNCS 1857 of *Lecture Notes in Computer Science*, pages 1–15. 2000.
- [FSF] FSF. GNU General Public License [online]. Disponível em <http://www.gnu.org/copyleft/gpl.htm> [último acesso: 2004 Dez 20].
- [Has00] R. Hashimoto. *Mudança de Estrutura de Representação de Operadores em Morfologia Matemática*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brasil, 2000.
- [Hau92] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150, 1992.
- [Hei94] H. Heijmans. *Morphological Image Operators*. Academic Press, Boston, 1994.
- [Hir00] N. Hirata. *Projeto Automático de Operadores: Explorando Conhecimentos a Priori*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brasil, Outubro 2000.
- [JDM00] A. Jain, R. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.
- [Jr.01] R. Hirata Jr. *Projeto de Operadores Morfológicos para Imagens e Sinais: Abordagem de Reticulados Finitos*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brasil, 2001.
- [KHDM98] J. Kittler, M. Hatef, R. Duin, and J. Matas. On Combining Classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

- [KHW97] J. Kittler, A. Hojjatoleslami, and T. Windeatt. Strategies for Combining Classifiers Employing Shared and Distinct Pattern Representations. *Pattern Recognition Letters*, 18(11-13):1373–1377, November 1997.
- [Kov02] Z. Kovács. *Redes Neurais Artificiais - Fundamentos e Aplicações*. Livraria da Física, São Paulo, third edition, 2002. ISBN 85-88325-14-4.
- [Men77] E. Mendelson. *Álgebra Booleana e Circuitos de Chaveamento*. McGraw-Hill, 1977.
- [Mey83] P. Meyer. *Probabilidade - Aplicações à Estatística*. LTC, Rio de Janeiro, second edition, 1983. ISBN 85-216-0294-4.
- [Mit97] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [ML04] M. Magalhães and A. Lima. *Noções de Probabilidade e Estatística*. EDUSP, São Paulo, sixth edition, 2004.
- [PY73] F. Preparata and R. Yeh. *Introduction to Discrete Structures for Computer Science and Engineering*. Computer Science and Information Processing. Addison-Wesley, 1973.
- [R] R. R Project [online]. Disponível em <http://www.r-project.org> [último acesso: 2005 Abr 15].
- [R D03a] R Development Core Team. *A Introduction to R*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-12-7.
- [R D03b] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-00-3.
- [R D03c] R Development Core Team. *R Data Import/Export*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-10-0.
- [R D03d] R Development Core Team. *R Installation and Administration*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-09-7.
- [R D03e] R Development Core Team. *R Language Definition*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-13-5.
- [R D03f] R Development Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2003. ISBN 3-900051-11-9.
- [Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

- [Sie68] W. Siebert. Stimulus Transformation in Peripheral Auditory System. In Murray Eden, editor, *Recognizing Patterns*, Cambridge, 1968. MIT Press.
- [Tav04] F. Tavares. Análise das Características Clínicas e Epidemiológicas de Pacientes com e sem Imunodeficiência Primária do Ambulatório de Imunologia Pediátrica da UNIFESP/EPM. Master's thesis, UNIFESP/EPM, São Paulo, 2004.
- [Tom96] N. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. Dissertação de Mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, SP, Brasil, Março 1996.
- [Vap98] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.
- [Vis] Vision. Computer Vision Research Group [online]. Disponível em <http://www.vision.ime.usp.br> [último acesso: 2004 Fev 20].