Seleção e Construção
de *Features* Relevantes
para o Aprendizado de Máquina

Huei Diana Lee

Orientação:
Profa. Dra. Maria Carolina Monard

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo — ICMC/USP, como parte dos requisitos para a obtenção do título de Mestre em Ciências - Área de Computação e Matemática Computacional.

USP - São Carlos
Março de 2000

Esta dissertação foi preparada com o formatador de textos $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. O sistema de citações de referências bibliográficas utiliza o padrão *Apalike* do sistema $\text{BIB}_{\text{E}}\text{X}$.

“Enthusiasm is the element of success in everything. It is the light that leads and the strength that lifts men on and up in the great struggles of scientific pursuits and of professional labor. It robs endurance of difficulty and makes pleasure of duty. ”

*Aos meus pais, Lee e Polly,
meus irmãos Jen e Linda,
ao meu marido, Paulo.*

Agradecimentos

À Profa. Maria Carolina Monard, minha querida professora, pela confiança, incentivo, extrema paciência com que me acompanhou durante o desenvolvimento deste trabalho e principalmente por sua amizade. Mais que uma professora, um exemplo de educadora a ser seguido.

Aos Profs. Solange Rezende e Ricardo Freitas pela amizade e incentivo.

Aos amigos Gustavo, Augusto e Walter pelo companheirismo, amizade e apoio.

À Jaque pela amizade, apoio e sugestões na versão final desse trabalho.

Aos amigos do Labic e de outros laboratórios, Claudinha, Chandler, Robson, Adenilso, Cláudia, Gedson, Teresinha, Adriano, Valéria, Raquel e muitos outros.

Aos funcionários do ICMC, em especial à Marília, Laura, Beth, Paulinho e Jacques.

Ao amigo Prof. Juan Carlos Sotuyo pela inestimável amizade, apoio e constante incentivo.

Ao Prof. Jorge Habib H. El Khouri pela eficiência em resolver os entraves burocráticos, amizade e apoio.

Ao amigo Sandro C. Esteves pela imensa colaboração nos estudos de casos Processamento de Sêmen Diagnóstico e Hiper-estimulação Ovariana Controlada.

Ao amigos e colegas, Profa. Letícia Mara Peres, Prof. Fabiano Silva e Prof. Antonio Marcos Hachisuca pela amizade, incentivo e apoio na realização de um sonho meu, um sonho nosso.

Aos colegas do Departamento de Ciência da Computação e funcionários da Unioeste.

À Unioeste e ao Itai pelo apoio financeiro.

À toda minha família, meus sogros, e em especial à minha mãe por todo apoio, incentivo e dedicação.

Ao meu marido Paulo, com carinho, pelo constante incentivo e apoio, especialmente nos momentos mais difíceis e por sua incansável dedicação.

Resumo

No Aprendizado de Máquina Supervisionado — AM — é apresentado ao algoritmo de indução um conjunto de instâncias de treinamento, no qual cada instância é um vetor de features rotulado com a classe. O algoritmo de indução tem como tarefa induzir um classificador que será utilizado para classificar novas instâncias.

Algoritmos de indução convencionais baseam-se nos dados fornecidos pelo usuário para construir as descrições dos conceitos. Uma representação inadequada do espaço de busca ou da linguagem de descrição do conjunto de instâncias, bem como erros nos exemplos de treinamento, podem tornar os problemas de aprendizado difíceis.

Um dos problemas centrais em AM é a Seleção de um Subconjunto de Features — SSF — na qual o objetivo é tentar diminuir o número de features que serão fornecidas ao algoritmo de indução.

São várias as razões para a realização de SSF. A primeira é que a maioria dos algoritmos de AM, computacionalmente viáveis, não trabalham bem na presença de muitas features, *i.e.* a precisão dos classificadores gerados pode ser melhorada com a aplicação de SSF. Ainda, com um número menor de features, a compreensibilidade do conceito induzido pode ser melhorada. Uma terceira razão é o alto custo para coletar e processar grande quantidade de dados. Existem, basicamente, três abordagens para a SSF: *embedded*, *filtro* e *wrapper*.

Por outro lado, se as features utilizadas para descrever os exemplos de treinamento são inadequadas, os algoritmos de aprendizado estão propensos a criar descrições excessivamente complexas e imprecisas. Porém, essas features, individualmente inadequadas, podem algumas vezes serem, convenientemente, combinadas gerando novas features que podem mostrar-se altamente representativas para a descrição de um conceito. O processo de construção de novas features é conhecido como Construção de Features ou Indução Construtiva — IC.

Neste trabalho são enfocadas as abordagens filtro e wrapper para a realização de SSF, bem como a IC guiada pelo conhecimento. É descrita uma série de experimentos usando SSF e IC utilizando quatro conjuntos de dados naturais e diversos algoritmos simbólicos de indução. Para cada conjunto de dados e cada indutor, são realizadas várias medidas, tais como, precisão, tempo de execução do indutor e número de features selecionadas pelo indutor. São descritos também diversos experimentos realizados utilizando três conjuntos de dados do mundo real. O foco desses experimentos não está somente na avaliação da performance dos algoritmos de indução, mas também na avaliação do conhecimento extraído. Durante a extração de conhecimento, os resultados foram apresentados aos especialistas para que fossem feitas sugestões para experimentos futuros. Uma parte do conhecimento extraído desses três estudos de casos foram considerados muito interessantes pelos especialistas. Isso mostra que a interação de diferentes áreas de conhecimento, neste caso específico, áreas médica e computacional, pode produzir resultados interessantes.

Assim, para que a aplicação do Aprendizado de Máquina possa gerar frutos é necessário que dois grupos de pesquisadores sejam unidos: aqueles que conhecem os métodos de AM existentes e aqueles com o conhecimento no domínio da aplicação para o fornecimento de dados e a avaliação do conhecimento adquirido.

Abstract

In supervised Machine Learning — ML — an induction algorithm is typically presented with a set of training instances, where each instance is described by a vector of feature values and a class label. The task of the induction algorithm (inducer) is to induce a classifier that will be useful in classifying new cases.

Conventional inductive-learning algorithms rely on existing (user) provided data to build their descriptions. Inadequate representation space or description language as well as errors in training examples can make learning problems be difficult.

One of the main problems in ML is the Feature Subset Selection — FSS — problem, *i.e.* the learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention, while ignoring the rest.

There are a variety of reasons that justify doing FSS. The first reason that can be pointed out is that most of the ML algorithms, that are computationally feasible, do not work well in the presence of a very large number of features. This means that FSS can improve the accuracy of the classifiers generated by these algorithms. Another reason to use FSS is that it can improve comprehensibility, *i.e.* the human ability of understanding the data and the rules generated by symbolic ML algorithms. A third reason for doing FSS is the high cost in some domains for collecting data. Finally, FSS can reduce the cost of processing huge quantities of data. Basically, there are three approaches in Machine Learning for FSS: embedded, filter and wrapper approaches.

On the other hand, if the provided features for describing the training examples are inadequate, the learning algorithms are likely to create excessively complex and inaccurate descriptions. These individually inadequate features can sometimes be combined conveniently, generating new features which can turn out to be highly representative to the description of the concept. The process of constructing new features is called Constructive Induction — CI.

In this work we focus on the filter and wrapper approaches for FSS as well as Knowledge-driven CI. We describe a series of experiments for FSS and CI, performed on four natural datasets using several symbolic ML algorithms. For each dataset, various measures are taken to compare the inducers performance, for example accuracy, time taken to run the inducers and number of selected features by each evaluated induction algorithm. Several experiments using three real world datasets are also described. The focus of these three case studies is not only comparing the induction algorithms performance, but also the evaluation of the extracted knowledge. During the knowledge extraction step results were presented to the specialist, who gave many suggestions for the development of further experiments. Some of the knowledge extracted from these three real world datasets were found very interesting by the specialist. This shows that the interaction between different areas, in this case, medical and computational areas, may produce interesting results. Thus, two groups of researchers need to be put together if the application of ML is to bear fruit: those that are acquainted with the existing ML methods, and those with expertise in the given application domain to provide training data.

Sumário

1	Introdução	1
2	Seleção de Features	6
2.1	Considerações Iniciais	6
2.2	O Problema da Seleção de Features	6
2.3	Relevância de Features	7
2.4	Seleção de um Subconjunto de Features	16
2.4.1	Seleção de Features como uma Busca Heurística	16
2.4.2	Abordagens para a Seleção de Features	19
2.4.2.1	<i>Embedded</i>	19
2.4.2.2	<i>Filtro</i>	21
2.4.2.3	<i>Wrapper</i>	22
2.5	Indução Construtiva	23
2.5.1	Abordagens para Indução Construtiva	25
2.5.2	Um Exemplo de Indução Construtiva	25
2.6	Considerações Finais	26
3	Algoritmos, Ferramentas e Conjuntos de Dados	28
3.1	Considerações Iniciais	28
3.2	Formato de Dados	28
3.3	Ferramentas	29
3.3.1	A Biblioteca <i>MCC</i> ++	29
3.3.2	MineSet	30
3.4	Indutores	31
3.4.1	ID3	31
3.4.2	<i>C4.5</i>	31
3.4.3	<i>C4.5rules</i>	32
3.4.4	<i>See5</i>	32

3.4.5	$\mathcal{CN}2$	33
3.5	Descrição dos Conjuntos de Dados	34
3.6	Considerações Finais	37
4	Seleção de um Subconjunto de Features Utilizando as Abordagens Wrapper e Filtro: Avaliação Experimental	38
4.1	Considerações Iniciais	38
4.2	Descrição dos Experimentos	38
4.3	Resultados Obtidos	40
4.3.1	Pima	41
4.3.2	Cmc	41
4.3.3	Smoke	42
4.3.4	Hepatitis	42
4.4	Análise de Resultados	43
4.5	Considerações Finais	48
5	Indução Construtiva - Avaliação Experimental	50
5.1	Considerações Iniciais	50
5.2	Descrição dos Experimentos	50
5.3	Resultados Obtidos	53
5.3.1	Pima e Conjuntos de Dados Derivados	54
5.3.2	Cmc e Conjuntos de Dados Derivados	56
5.3.3	Smoke e Conjunto de Dados Derivados	57
5.3.4	Hepatitis e Conjunto de Dados Derivados	59
5.4	Análise de Resultados	61
5.5	Considerações Finais	63
6	Estudo de Casos	64
6.1	Considerações Iniciais	64
6.2	Fases do Estudo de Caso	64
6.3	Estudo de Caso 1 - Processamento de Sêmen Diagnóstico	66
6.3.1	Coleta, Preparação e Limpeza dos Dados	67
6.3.2	Descrição do Conjunto de Dados	67
6.3.3	Experimentos Realizados e Resultados	69
6.3.4	Algumas Considerações	75
6.4	Estudo de Caso 2 - Hiper-estimulação Ovariana Controlada	76
6.4.1	Coleta, Preparação e Limpeza dos Dados	77

6.4.2	Descrição do Conjunto de Dados	78
6.4.3	Experimentos Realizados e Resultados	79
6.4.4	Algumas Considerações	83
6.5	Estudo de Caso 3 - Análise do Parâmetro Físico-Mecânico Tensão Resultante Final sobre a Anastomose Cólica em Ratos	83
6.5.1	Descrição do Conjunto de Dados	85
6.5.2	Experimentos Realizados e Resultados	86
6.5.3	Algumas Considerações	88
6.6	Considerações Finais	88
7	Conclusões	90

Lista de Figuras

1.1	Sistema de Aprendizado	2
1.2	Sistema de Classificação	2
2.4.1	Exemplo de Espaço de Estados de Features	17
2.4.2	Abordagem <i>Embedded</i>	20
2.4.3	Abordagem <i>Filtro</i>	22
2.4.4	Uma Visão da Relevância do Conjunto de Features	23
2.4.5	Abordagem <i>Wrapper</i>	24
2.5.1	Árvore e Regras de Decisão para o Exemplo de Robôs Amigos e Inimigos .	26
2.5.2	Árvore e Regras de Decisão para o Exemplo de Robôs Amigos e Inimigos após a Construção de Features	26
3.5.1	Dimensionalidade dos Conjuntos de Dados	37
4.2.1	Seleção de um Subconjunto de Features — Passos dos Experimentos	39
4.4.2	$\mathcal{C}4.5$ — Diferença em Desvios Padrões dos Erros	45
4.4.3	$\mathcal{CN}2$ — Diferença em Desvios Padrões dos Erros	46
4.4.4	$\mathcal{C}4.5rules$ — Diferença em Desvios Padrões dos Erros	47
4.4.5	Diferença em Desvios Padrões dos Erros entre os Melhores Métodos para SSF	48
5.2.1	Indução Construtiva - Passos dos Experimentos	51
6.2.1	Fases do Estudo de Caso	65
6.4.1	Injeção Intracitoplasmática do Espermatozóide no Oócito — ICSI	76
6.5.1	Alça Cólica do Grupo 1 Durante Experimento	84
6.5.2	Alça Cólica do Grupo 3 Durante Experimento	85
6.5.3	Visualização Tridimensional do Conjunto de Dados Tensão - Média, FT e Tcirc	88

Lista de Tabelas

2.3.1 Formato Padrão do Conjunto de Exemplos	8
2.3.2 Exemplos para Ilustrar Definições de Relevância	9
2.3.3 Exemplos Gerados pela Função de Paridade	10
2.3.4 Features Relevantes e Irrelevantes	11
2.3.5 Exemplos para Ilustrar a Redundância de Features	12
2.3.6 Exemplos do Conceito Meta $Y = X_1 \oplus X_2$	14
2.3.7 Features Relevantes e Irrelevantes segundo as Definições de Relevância	14
2.3.8 Exemplos para Ilustrar que uma Feature ser Relevante não Implica ser Ótima	16
2.4.1 Amostra de Exemplos para Ilustrar o Paradigma da Busca Heurística para a Seleção de Features	19
2.5.1 Exemplos de Robôs Amigos e Inimigos	25
2.5.2 Exemplos de Robôs Amigos e Inimigos depois da Construção de Features	26
3.2.1 Formato Atributo-Valor	28
3.5.1 Pima – Descrição de Features	34
3.5.2 Cmc – Descrição de Features	35
3.5.3 Smoke – Descrição de Features	35
3.5.4 Hepatitis – Descrição de Features	36
3.5.5 Resumo dos Conjuntos de Dados	36
4.3.1 Pima – Features Seleccionadas e Tempo para Seleccionar Features	41
4.3.2 Pima – Erros	41
4.3.3 Cmc – Features Seleccionadas e Tempo para Seleccionar Features	41
4.3.4 Cmc – Erros	41
4.3.5 Smoke – Features Seleccionadas e Tempo para Seleccionar Features	42
4.3.6 Smoke – Erros	42
4.3.7 Hepatitis – Features Seleccionadas e Tempo para Seleccionar Features	42
4.3.8 Hepatitis – Erros	42
4.4.9 Proporção e Média de Features Seleccionadas	43

4.4.10	Tempo (em segundos) para a Seleção de Features	43
4.4.11	Melhora nas Precisoões com um Grau de Confiança de 95%	45
5.2.1	Conjunto de Dados Originais e Derivados	51
5.3.2	Pima e Conjuntos de Dados Derivados – Features Originais e Construídas .	55
5.3.3	Pima e Conjuntos de Dados Derivados – Features Seleccionadas, Tempo para Seleccionar Features e Número de Instâncias Duplicadas ou Conflitantes	55
5.3.5	Pima e Conjuntos de Dados Derivados – Erros	56
5.3.7	Cmc e Conjuntos de Dados Derivados – Features Originais e Construídas .	57
5.3.8	Cmc e Conjuntos de Dados Derivados – Features Seleccionadas, Tempo para Seleccionar Features e Número de Instâncias Duplicadas ou Conflitantes . .	57
5.3.9	Cmc e Conjuntos de Dados Derivados – Erros	57
5.3.10	Smoke e Conjuntos de Dados Derivados – Features Originais e Construídas	58
5.3.11	Smoke e Conjuntos de Dados Derivados – Features Seleccionadas, Tempo para Seleccionar Features e Número de Instâncias Duplicadas ou Conflitantes	59
5.3.12	Smoke e Conjuntos de Dados Derivados – Erros	59
5.3.13	Hepatitis e Conjuntos de Dados Derivados – Features Originais e Construídas	60
5.3.14	Hepatitis e Conjuntos de Dados Derivados – Features Seleccionadas, Tempo para Seleccionar Features e Número de Instâncias Duplicadas ou Conflitantes	60
5.3.15	Hepatitis e Conjuntos de Dados Derivados – Erros	60
5.4.16	Exemplo de Instâncias Duplicadas ou Conflitantes	61
5.4.17	Exemplo de Instâncias Duplicadas ou Conflitantes Depois da Remoção das Features Primitivas	61
5.4.18	Resumo dos Resultados	63
6.3.1	Processamento – Descrição das Features	68
6.3.2	Processamento — Resumo dos Conjuntos de Dados	69
6.3.3	Processamento — Resumo dos Resultados	70
6.3.4	Processamento — Resumo dos Conjuntos de Dados Derivados	72
6.4.1	HOC – Descrição das Features	78
6.4.2	HOC — Resumo do Conjunto de Dados	79
6.4.3	HOC — Resumo dos Resultados	79
6.4.4	HOC — Matriz de Confusão Utilizando Todos os Exemplos Disponíveis . .	80
6.4.5	HOC02 – Descrição do Conjunto Reduzido de Features	82
6.4.6	HOC02 — Resumo dos Resultados	83
6.5.1	Tensão – Descrição das Features	85
6.5.2	Tensão — Resumo do Conjunto de Dados	86
6.5.3	Tensão — Resumo dos Resultados	87

Lista de Abreviações

AM		Aprendizado de Máquina
FIV		Fertilização In Vitro
HOC		Hiper-estimulação Ovariana Controlada
IA		Inteligência Artificial
IC		Indução Construtiva
ICSI		Injeção Intracitoplasmática do Espermatozóide no Oócito
IUI		Inseminação Intra Uterina
NN		Nearest Neighbor
SF		Seleção de Features
SSF		Seleção de um Subconjunto de Features

Capítulo 1

Introdução

Conceitos e técnicas de Inteligência Artificial têm sido incorporados cada vez mais às soluções de problemas reais de diversas áreas através de Sistemas de Inteligência Artificial. Tais sistemas dependem fortemente do conhecimento, o qual deve ser adquirido, representado e manipulado. Os processos de aquisição, representação e manipulação de conhecimento estão relacionados ao processo de aprendizado, o qual mostra-se essencial para que uma máquina possa exibir um comportamento inteligente.

A aquisição de conhecimento pode ser realizada de duas formas: manualmente ou automaticamente. Na primeira, o conhecimento é adquirido através de pesquisa direta sobre material relacionado à área em questão ou através de entrevistas com especialistas do domínio de interesse, ambas realizadas por um Engenheiro de Conhecimento, o qual colhe as informações e explicita o conhecimento adquirido. Na segunda forma, a aquisição de conhecimento é realizada pela máquina através de Sistemas de Aprendizado de Máquina — AM.

Desse modo, Aprendizado de Máquina pode ser definido como uma sub-área da Inteligência Artificial que pesquisa métodos computacionais relacionados à aquisição automática de novos conhecimentos, novas habilidades e novas formas de organizar o conhecimento já existente. Ainda, com o crescimento contínuo do volume de informações geradas e armazenadas por sistemas modernos, o Aprendizado de Máquina pode ser aplicado como meio de vencer um dos maiores problemas em Sistemas de Inteligência Artificial — o gargalo da aquisição de conhecimento.

Um Sistema de AM é então, um programa de computador que toma decisões baseado em experiências acumuladas contidas em casos resolvidos com sucesso (Weiss and Kulikowski, 1990). Mais especificamente, a atenção está voltada para uma das tarefas mais básica e proeminente a *predição*.

O projeto de um Sistema de AM típico envolve a definição do paradigma de aprendizado sob o qual esse trabalha, bem como a estratégia de aprendizado. Os paradigmas de aprendizado determinam a forma como o conhecimento adquirido será representado. Exemplos desses são o paradigma simbólico, estatístico, conexionista, entre outros. A estratégia de aprendizado irá então definir o modo como o aprendizado ocorre. Assim, podem ser enumeradas cinco estratégias de aprendizado segundo o grau de complexidade de inferência: estratégia de aprendizado por hábito, por instrução, por dedução, por analogia e aprendizado por indução. A primeira ilustra a estratégia de aprendizado com menor complexidade de inferência, enquanto que a estratégia de aprendizado por indução

é a que exige a maior complexidade de inferência, isto é, na qual o aprendiz desenvolve a maior parte do esforço exigido para o aprendizado (Monard et al., 1997). Assim, o aprendizado por indução é uma das estratégias de maior interesse para o desenvolvimento de pesquisas.

Neste trabalho é tratado o aprendizado supervisionado por indução. Nele, conforme ilustra a Figura 1.1, um conjunto de exemplos e suas respectivas classes é fornecido a um Sistema de Aprendizado (algoritmo de indução ou simplesmente indutor). Quando os valores das classes são discretos a tarefa de predição é chamada de *classificação*; quando os valores das classes são contínuos a tarefa é chamada de *regressão*. Neste trabalho são considerados problemas de classificação. Assim, a saída gerada pelo Sistema de Aprendizado é um classificador, o qual mapeia instâncias não classificadas para rótulos (classes) utilizando alguma estrutura interna armazenada — Figura 1.2.

Figura 1.1: Sistema de Aprendizado

Desse modo, o objetivo fundamental de um Sistema de AM está na extração de conhecimento na forma de, por exemplo, regras de decisão, a partir de um conjunto de dados (exemplos) tal que essas regras sejam aplicáveis a novos dados ou, ainda, extrair o máximo de informação dos exemplos e estimar a precisão de futuras classificações. “Aprender” consiste em escolher ou adaptar parâmetros da representação do modelo, no paradigma escolhido, de maneira que o modelo classifique corretamente os exemplos conhecidos bem como novos exemplos.

Figura 1.2: Sistema de Classificação

Esses exemplos são representados por registros de características. As características potencialmente relevantes para um problema em particular são também referenciadas como *features*, atributos¹, variáveis, testes e medições (Weiss and Kulikowski, 1990).

¹Os termos *features* e atributos serão utilizados indistintamente neste trabalho por serem amplamente aceitos pela comunidade acadêmica da área de Inteligência Artificial.

Pode-se dizer, então, que um Sistema de AM objetiva moldar a estrutura de classificação para um problema específico, encontrando uma forma genérica de relatar qualquer padrão particular de observações (conceito). Assim, quanto mais significativas as features utilizadas para descrever os exemplos, mais confiável será a classificação.

Em outras palavras, se os atributos captam as propriedades essenciais do conceito, o papel do Sistema de AM é rearranjar esses atributos numa forma genérica de relatar tal conceito. Entretanto, se os atributos não expressarem as relações inerentes ao processo classificatório, as generalizações produzidas por qualquer algoritmo de aprendizado terão baixo desempenho na classificação de novos casos (Nicoletti, 1994). Desse modo, a capacidade de predição das features é fundamental para o sucesso de qualquer Sistema de AM.

Um outro fator que influencia no desempenho das predições é o número de features levadas em consideração no processo de aprendizado, pois a maioria dos algoritmos computacionalmente viáveis de Aprendizado de Máquina não trabalham bem na presença de um número grande de features. A granularidade das features determina a complexidade da regra de classificação. Sendo assim, features de alto nível facilitam a representação do conceito, porém a determinação de features apropriadas para descrever um conceito é um problema complexo.

Este trabalho tem como objetivo principal pesquisar e analisar métodos computacionais para encontrar, possivelmente, o melhor conjunto de features para um classificador, denominado Seleção de Features – SF.

Existem duas abordagens principais para o tratamento de SF:

1. Seleção de um Subconjunto de Features — SSF ou FSS² e
2. Indução Construtiva — IC, também denominada de Aprendizado Construtivo ou Construção de Features.

No primeiro caso, dado um conjunto de n features, após o processo de Seleção de um Subconjunto de Features, será obtido um subconjunto de features de tamanho m tal que $m \leq n$. Em outras palavras, SSF produz uma diminuição do número de features que serão posteriormente consideradas no processo de indução. Já no caso da Indução Construtiva, dado o mesmo conjunto de features de tamanho n , através da combinação de features originais, ditas features primitivas, obtém-se novas features que, possivelmente, podem melhorar a linguagem de descrição do conjunto de exemplos e, conseqüentemente, a precisão do classificador. Desse modo, contrariamente à Seleção de um Subconjunto de Features, a construção de features aumenta o número de atributos, isto é, ao final deste processo serão obtidas m features, sendo $m > n$.

Um problema central da Seleção de Features é como determinar quais features devem ser selecionadas ou quais features devem ser ignoradas. Intuitivamente, seria desejável que um indutor utilizasse apenas as “features relevantes” para o aprendizado do conceito. Porém, a inexistência de uma análise matemática que permita a avaliação do desempenho de um método de aprendizado sobre um conjunto de dados faz com que estudos experimentais sejam muito importantes na área de Aprendizado de Máquina. Assim, também para a tarefa de Seleção de Features é importante que estudos empíricos sejam realizados sobre os

²Feature Subset Selection.

conjuntos de dados de interesse, a fim de avaliar que métodos de SF são mais apropriados, ou apresentam o melhor desempenho, para indutores e conjuntos de dados específicos.

Neste trabalho são apresentadas, além de diversas definições de relevância de features encontradas na literatura, uma série de avaliações empíricas de Seleção de Features utilizando vários conjuntos de dados. Os conjuntos de dados para extração de conhecimento têm sido classificados pela comunidade de AM como (Batista, 2000):

- *mundo real*, por exemplo, os conjuntos de dados extraídos diretamente de bases de dados de empresas, de registros da área médica e outros;
- *naturais* quando são obtidos de repositórios de dados como o repositório da UCI (Blake et al., 1998) e
- *artificiais* quando os dados são gerados por programas computacionais; nesses conjuntos de dados sabe-se exatamente o comportamento dos dados e são utilizados geralmente quando o objetivo é testar capacidades específicas de algoritmos de AM.

As avaliações empíricas apresentadas neste trabalho utilizam tanto dados naturais quanto dados do mundo real.

Este trabalho está organizado da seguinte forma.

No Capítulo 2 são apresentadas diversas definições de relevância de features encontradas na literatura. Também são descritas as duas principais abordagens para Seleção de Features: Seleção de um Subconjunto de Features e Indução Construtiva. Para SSF é descrito um paradigma que a coloca sob o ponto de vista de uma busca heurística, na qual cada estado no espaço de busca é composto por um subconjunto de features. As três abordagens para SSF, *embedded*, *filtro* e *wrapper*, são apresentadas bem como quatro possíveis abordagens para realizar Indução Construtiva.

No Capítulo 3 são apresentados as ferramentas e os algoritmos de indução utilizados para realizar diversas avaliações empíricas de Seleção de Features, o esquema geral do formato de dados utilizados pelas ferramentas e algoritmos e os conjuntos de dados usados nos experimentos.

No Capítulo 4 são descritos os experimentos realizados utilizando a abordagem wrapper e a abordagem filtro para a SSF. Ambas as abordagens são avaliadas e comparadas considerando precisão, tempo e número de features selecionadas.

No Capítulo 5 são descritos os experimentos realizados utilizando a segunda abordagem para a Seleção de Features, *i.e.* Indução Construtiva. Esses experimentos tem como objetivo avaliar a construção de features com o auxílio de um usuário/especialista, a qual é denominada IC guiada pelo conhecimento. Com esse fim, é por nós proposta uma metodologia genérica para a aplicação de IC guiada pelo conhecimento composta de três passos, os quais são descritos em detalhes nesse capítulo e exemplificados pelos experimentos apresentados.

No Capítulo 6 são descritos três estudos de casos do mundo real, aos quais são aplicadas algumas das técnicas e metodologias por nós propostas nos capítulos anteriores. Os casos analisados são: Processamento de Sêmen Diagnóstico, Hiper-estimulação Ovariana Controlada e Análise do Parâmetro Físico-Mecânico Tensão Resultante Final sobre a Anastomose Cólica em Ratos.

Para cada um deles são descritas todas as etapas desenvolvidas durante os estudos de casos, bem como dificuldades e resultados alcançados.

Finalmente, no Capítulo 7 são apresentadas as conclusões deste trabalho e pesquisas futuras.

Capítulo 2

Seleção de Features

2.1 Considerações Iniciais

A evolução tecnológica permitiu a coleta e o armazenamento de uma quantidade cada vez maior de dados. Com a aplicação cada vez mais freqüente de técnicas de Aprendizado de Máquina a essas bases de dados, o problema de focalizar-se a informação mais relevante tornou-se muito importante. Assim, um dos principais problemas em AM é a seleção de features relevantes e a eliminação das irrelevantes.

Existem diversas razões para a realização de Seleção de Features. Uma dessas razões é que a maioria dos algoritmos de AM computacionalmente viáveis, não trabalham bem na presença de um número grande de features, isto é, Seleção de Features pode melhorar a precisão dos classificadores gerados por esses algoritmos. Outra razão é que a SF melhora a compreensibilidade, ou seja, a habilidade de compreensão dos dados pelos humanos e também, por exemplo, das regras de indução geradas por algoritmos simbólicos de AM. Uma terceira razão para a realização de SF é o alto custo para a coleta dos dados, já que em muitos domínios isso pode ser um processo muito caro. Um exemplo disso seriam as informações que são coletadas de pacientes para os registros médicos. Muitas das informações contidas nesses registros, tais como exames laboratoriais, exigiram um custo alto para sua geração e, possivelmente, nem toda a informação seja necessária para o aprendizado de um conceito. Finalmente, a Seleção de Features pode reduzir os custos de processamento de grande quantidade de dados.

Neste capítulo são discutidas e exemplificadas diversas definições de relevância de features. Também são descritas as duas abordagens para Seleção de Features: Seleção de um Subconjunto de Features e Indução Construtiva ou Construção de Features.

2.2 O Problema da Seleção de Features

Do ponto de vista conceitual, a tarefa de aprendizado de conceitos pode ser dividida em duas sub-tarefas:

1. decidir que features utilizar na descrição do conceito e
2. decidir como combinar essas features.

Sob essa visão, a seleção de features relevantes e a eliminação das irrelevantes constitui um dos principais problemas a serem tratados em AM.

Do ponto de vista prático, é desejado que algoritmos de indução trabalhem bem em domínios que contenham muitas features irrelevantes. Em outras palavras, um dos objetivos é que o número de exemplos de treinamento necessários para se atingir uma dada precisão (complexidade da amostra) cresça lentamente em relação ao número de features irrelevantes. Por exemplo, na tarefa de classificação de texto, não é incomum a representação de exemplos utilizando de 10^4 a 10^7 atributos, sabendo-se que apenas uma pequena fração desses é crucial para o aprendizado do conceito. Nos últimos anos, uma quantidade crescente de pesquisas em AM, tanto teóricas quanto práticas, tem sido voltadas para o desenvolvimento de algoritmos que trabalhem bem na presença de muitas features irrelevantes.

Assim, intuitivamente, seria desejável que um indutor utilizasse apenas as “features relevantes” para o aprendizado do conceito. Na seção seguinte são descritas diversas noções formais de “relevância” no contexto de Aprendizado de Máquina Supervisionado, as quais auxiliam na ilustração de alguns dos objetivos gerais dos algoritmos de seleção de features.

2.3 Relevância de Features

Existem diferentes definições na literatura para o significado da relevância de uma feature. A razão para a existência dessa variedade de definições está relacionada a pergunta:

Relevante em relação a que ?

Desse modo, existem diversas definições de relevância dependendo dos objetivos a serem alcançados, tais como relevância em relação ao conceito meta e em relação a distribuição de probabilidade, entre outros. Antes de iniciar a apresentação das diversas definições de relevância de features, algumas considerações devem ser feitas referentes a notação utilizada:

1. cada exemplo do conjunto de instâncias é descrito por n features;
2. o domínio de cada feature é denotado por D_i ;
3. uma feature pode ser do tipo categórico¹, isto é, pode assumir um número finito de valores discretos, ou pode ser do tipo contínuo e
4. um exemplo (instância, caso ou registro) é um ponto no espaço de instâncias $D_1 \times D_2 \times \dots \times D_n$.

O algoritmo de aprendizado recebe como entrada um conjunto S de exemplos de treinamento, no qual cada exemplo é associado a uma classe (rótulo), que pode ser discreta ou contínua. A Tabela 2.3.1 ilustra o formato dos exemplos e a classe associada.

Dessa forma, o problema de classificação pode ser resumido da seguinte forma:

¹Nesse trabalho os termos categórico, discreto e nominal serão utilizados indistintamente.

	Atributos				
Exemplos	X_1	X_2	$\dots$	X_n	Classe (Y)
E_1	x_{11}	x_{12}	$\dots$	x_{1n}	y_1
E_2	x_{21}	x_{22}	$\dots$	x_{2n}	y_2
E_3	x_{31}	x_{32}	$\dots$	x_{3n}	y_3
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
E_m	x_{m1}	x_{m2}	$\dots$	x_{mn}	y_m

Tabela 2.3.1: Formato Padrão do Conjunto de Exemplos

dados exemplos E_i compostos pelos valores de n atributos $X_1, X_2, \dots, X_n$, isto é, $E_i = (x_{i1}, x_{i2}, \dots, x_{in})$ e a classe y associada a cada exemplo, a tarefa é encontrar o conceito meta (ou função) c , tal que $y = c(E)$.

Como mencionado anteriormente, os valores assumidos por Y são tipicamente valores discretos no caso da classificação, ou valores contínuos em problemas de regressão.

Embora o algoritmo de aprendizado visualize apenas a amostra fixa S , geralmente é útil considerar duas informações adicionais que auxiliam na definição da relevância de features:

1. distribuição de probabilidade $\mathcal{D}$ sobre o espaço de instâncias e
2. função de classificação c que mapeia os exemplos para as classes.

Sendo assim, é possível modelar a amostra S como sendo gerada pela seleção repetida de exemplos a partir de $\mathcal{D}$ rotulando esses exemplos de acordo com a função de classificação c . Pode-se notar que a função de classificação c pode tanto ser determinística quanto probabilística, nesse último caso, para um determinado exemplo E , $c(E)$ seria a distribuição de probabilidade sobre as classes ao invés de uma única classe.

Feitas as considerações necessárias quanto a notação empregada neste trabalho, a seguir serão apresentadas diversas definições de relevância existentes na literatura.

Uma definição de relevância muito simples que considera que tanto as features quanto a classe são booleanas, além de não existir ruído nos dados, é a seguinte:

Definição 2.3.1 (*Relevância em relação ao Conceito Meta*) (Almuallim and Diettrich, 1991) *Uma feature X_i é relevante para o conceito meta c se X_i aparece em toda fórmula booleana que representa c e irrelevante caso contrário.*

Por exemplo, considere que um determinado indutor $\mathcal{I}$ gerou um classificador composto pelas seguintes regras:

Se $X_1=1$ e $X_2=0$ então classe=1

Se $X_1=1$ e $X_3=0$ então classe=1

Se $X_1=0$ e $X_2=1$ então classe=0

Segundo essa definição, a feature X_1 seria relevante, pois ela aparece em todas as regras que compõem c , enquanto que X_2 e X_3 seriam irrelevantes.

Uma outra definição muito simples, que define features relevantes como aquelas que variam sistematicamente com a classe associada, é apresentada a seguir. Nessa definição, bem como nas definições seguintes, é considerado $x_i \in \{x_{1i}, x_{2i}, \dots, x_{mi}\}$.

Definição 2.3.2 (*Relevância - Abordagem Probabilística*) (Gennari et al., 1989) Uma feature X_i é relevante sss² existe algum x_i e y para os quais $p(X_i = x_i) > 0$ tal que³

$$p(Y = y | X_i = x_i) \neq p(Y = y)$$

Segundo essa definição, X_i é relevante se, conhecido seu valor, a estimativa para a classe Y é modificada ou, em outras palavras, se Y é condicionalmente dependente de X_i .

Exemplos	X_1	X_2	X_3	Classe
E_1	♣	♥	♦	+
E_2	♣	♣	♦	+
E_3	♥	♥	♣	+
E_4	♥	♥	♥	-

Tabela 2.3.2: Exemplos para Ilustrar Definições de Relevância

Por exemplo, considerando E_1 — Tabela 2.3.2 — e a feature X_1 tem-se:

$$p(Y = + | X_1 = \clubsuit) \neq p(Y = +)$$

pois $p(Y = + | X_1 = \clubsuit) = 1$ enquanto que $p(Y = +) = \frac{3}{4}$, ou seja, X_1 é relevante. Se as probabilidades forem calculadas para as features X_2 e X_3 , poderá verificar-se que ambas também serão consideradas relevantes segundo essa definição.

Pode-se notar que essa definição considera apenas a feature em questão e a classe para o cálculo da relevância. Porém, essa definição falha na captura da relevância de features para a Função de Paridade⁴, na qual todos os exemplos são equiprováveis.

Considere a Função de Paridade definida na Tabela 2.3.3. Segundo a Definição 2.3.2 se

$$p(Y = y | X_i = x_i) \neq p(Y = y)$$

então X_i é relevante. Calculando as probabilidades para as features X_1 , X_2 , X_3 e a classe Y tem-se:

$$\begin{aligned} p(Y = 1 | X_1 = 1) &= p(Y = 0 | X_1 = 1) = p(Y = 1 | X_1 = 0) = p(Y = 0 | X_1 = 0) = \frac{1}{2} \\ p(Y = 1 | X_2 = 1) &= p(Y = 0 | X_2 = 1) = p(Y = 1 | X_2 = 0) = p(Y = 0 | X_2 = 0) = \frac{1}{2} \\ p(Y = 1 | X_3 = 1) &= p(Y = 0 | X_3 = 1) = p(Y = 1 | X_3 = 0) = p(Y = 0 | X_3 = 0) = \frac{1}{2} \end{aligned}$$

²se e somente se.

³Em geral, as definições dadas aqui aplicam-se a features discretas, porém podem ser estendidas para features contínuas substituindo-se $p(X = x)$ por $p(X \leq x)$.

⁴A Função de Paridade retorna 1 sss um número par de *inputs* (entradas) é 1.

e

$$p(Y = 1) = \frac{1}{2}$$

Desse modo, para o caso da Função de Paridade

$$p(Y = y|X_i = x_i) = p(Y = y)$$

e portanto nenhuma das três features é considerada relevante segundo essa definição.

Exemplos	X_1	X_2	X_3	Classe
E_1	0	0	0	0
E_2	0	0	1	1
E_3	0	1	0	1
E_4	0	1	1	0
E_5	1	0	0	0
E_6	1	0	1	1
E_7	1	1	0	1
E_8	1	1	1	0

Tabela 2.3.3: Exemplos Gerados pela Função de Paridade

Assim, para que possa ser levada em consideração a influência de todas as features e o caso em que todos os exemplos são equiprováveis, a Definição 2.3.2 pode ser redefinida da seguinte forma.

Seja $S_i = \{ X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n \}$ o subconjunto de todas as features exceto X_i , e s_i uma atribuição de valores a todas as features em S_i .

Definição 2.3.3 (*Relevância - Abordagem Probabilística*) (*John et al., 1994*) Uma feature X_i é relevante sss existe algum x_i , y e s_i para $p(X_i = x_i) > 0$ tal que

$$p(Y = y, S_i = s_i|X_i = x_i) \neq p(Y = y, S_i = s_i)$$

Na definição seguinte, X_i é relevante se a probabilidade da classe, dadas todas as features, pode mudar quando o conhecimento sobre o valor de X_i é eliminado.

Definição 2.3.4 (*Relevância - Abordagem Probabilística*) (*John et al., 1994*) Uma feature X_i é relevante sss existe algum x_i , y e s_i para $p(X_i = x_i, S_i = s_i) > 0$ tal que

$$p(Y = y|X_i = x_i, S_i = s_i) \neq p(Y = y|S_i = s_i)$$

Para os exemplos da Tabela 2.3.2, as features consideradas relevantes e irrelevantes pelas Definições 2.3.3 e 2.3.4 não são as mesmas, como mostra a Tabela 2.3.4. Para tanto foram calculadas as probabilidades das features X_1 , X_2 e X_3 para cada um dos possíveis valores assumidos pelas mesmas. Uma feature será então considerada relevante, segundo a Definição 2.3.3, se pelo menos um dos valores assumidos por X_i e S_i permitirem que

$$p(Y = y, S_i = s_i|X_i = x_i) \neq p(Y = y, S_i = s_i).$$

Por exemplo, para $X_1 = \clubsuit$ segundo a Definição 2.3.3:

$$p(Y = +, S_1 = \{\heartsuit, \diamondsuit\} | X_1 = \clubsuit) \neq p(Y = +, S_1 = \{\heartsuit, \diamondsuit\})$$

Dessa forma, X_1 é uma feature relevante segundo essa definição, pois $p(Y = +, S_1 = \{\heartsuit, \diamondsuit\} | X_1 = \clubsuit) = \frac{1}{2}$ e $p(Y = +, S_1 = \{\heartsuit, \diamondsuit\}) = \frac{1}{4}$. Fazendo o mesmo cálculo para as features X_2 e X_3 , pode-se verificar que as mesmas também serão consideradas relevantes segundo essa definição.

Do mesmo modo, para determinar quais features são relevantes segundo a Definição 2.3.4, é necessário encontrar apenas um X_i e S_i para os quais

$$p(Y = y | X_i = x_i, S_i = s_i) \neq p(Y = y | S_i = s_i)$$

Assim, X_1 é considerada irrelevante segundo essa definição pois para nenhum dos valores assumidos por X_1 e S_1 a definição é satisfeita:

$$p(Y = + | S_1 = \{\heartsuit, \diamondsuit\}, X_1 = \clubsuit) = p(Y = + | S_1 = \{\heartsuit, \diamondsuit\}) = 1$$

$$p(Y = + | S_1 = \{\heartsuit, \clubsuit\}, X_1 = \heartsuit) = p(Y = + | S_1 = \{\heartsuit, \clubsuit\}) = 1$$

Igualmente, para os valores assumidos por X_2 , tem-se:

$$p(Y = + | S_2 = \{\clubsuit, \diamondsuit\}, X_2 = \heartsuit) = p(Y = + | S_2 = \{\clubsuit, \diamondsuit\}) = 1$$

$$p(Y = + | S_2 = \{\clubsuit, \diamondsuit\}, X_2 = \clubsuit) = p(Y = + | S_2 = \{\clubsuit, \diamondsuit\}) = 1$$

sendo portanto X_2 também irrelevante segundo essa definição. Porém, realizando o cálculo para a feature X_3 pode-se verificar que:

$$p(Y = + | S_3 = \{\heartsuit, \heartsuit\}, X_3 = \clubsuit) \neq p(Y = + | S_3 = \{\heartsuit, \heartsuit\})$$

pois $p(Y = + | S_1 = \{\heartsuit, \heartsuit\} | X_3 = \clubsuit) = 1$ e $p(Y = + | S_3 = \{\heartsuit, \heartsuit\}) = \frac{1}{2}$. Sendo assim, essa feature é considerada relevante segundo essa definição.

Definição	X_1	X_2	X_3
Definição 2.3.3	Relevante	Relevante	Relevante
Definição 2.3.4	Irrelevante	Irrelevante	Relevante

Tabela 2.3.4: Features Relevantes e Irrelevantes

A seguir, é apresentada a definição de relevância de features segundo (Blum and Langley, 1997), a qual é equivalente a definição de (John et al., 1994) — Definição 2.3.4.

Definição 2.3.5 (*Relevância em relação ao Conceito Meta*) (Blum and Langley, 1997)
 Uma feature X_i é relevante para o conceito meta c sss existe um par de exemplos E_i e E_j , $i \neq j$, no espaço de exemplos tal que E_i e E_j diferem somente na atribuição de valores à feature X_i e $c(E_i) \neq c(E_j)$.

Em outras palavras, a feature X_i é relevante se existe algum exemplo no espaço de instâncias para o qual a modificação do valor de X_i afeta a classificação dada pelo conceito meta. Os exemplos E_3 e E_4 na Tabela 2.3.2 (pg. 9), ilustram essa definição, com E_3 e E_4 diferindo somente no valor da feature X_3 e $c(E_3) \neq c(E_4)$. Assim, segundo essa definição a feature X_3 seria relevante, bem como segundo a Definição 2.3.4.

Observe que a Definição 2.3.5 — Relevância em relação ao Conceito Meta — é muito mais restrita que a Definição 2.3.2 — Relevância - Abordagem Probabilística — (pg. 9), pois a primeira exige que todos os valores das features (exceto o valor da feature em questão e a classe) de dois exemplos quaisquer sejam iguais. Por outro lado, a Definição 2.3.2 mostra-se mais abrangente, pois atribui a relevância a uma dada feature comparando a probabilidade de uma dada classe ocorrer sob duas condições: considerando a influência daquela feature ou não.

Note também que a Definição 2.3.4, e a Definição 2.3.5 equivalente, têm a desvantagem de que o algoritmo de aprendizado, dado acesso a apenas a amostra S , não é necessariamente capaz de determinar quando alguma feature X_i é relevante ou não. Pior ainda, se a representação das features é redundante, pode não ser possível encontrar dois exemplos que diferem apenas pelo valor de uma feature, já que pelo menos um desses exemplos teria probabilidade zero. Por exemplo, considere a Tabela 2.3.5, na qual $D_1 = \{\bullet, \diamond\}$, $D_2 = \{\triangleleft, \natural\}$ e $D_3 = \{\star, \oslash\}$.

Exemplos	X_1	X_2	X_3	Classe
E_1	$\bullet$	$\natural$	$\star$	+
E_2	$\diamond$	$\triangleleft$	$\star$	−
E_3	$\bullet$	$\natural$	$\oslash$	+
E_4	$\diamond$	$\triangleleft$	$\oslash$	−

Tabela 2.3.5: Exemplos para Ilustrar a Redundância de Features

Pode-se observar que não é possível encontrar nesse exemplo um único par E_i e E_j tal que seja possível determinar uma feature relevante, justamente por causa da redundância das features X_1 e X_2 , que verificam

$$X_1 = \bullet \text{ sss } X_2 = \natural \text{ e } X_1 = \diamond \text{ sss } X_2 = \triangleleft$$

Para tentar contornar algumas das desvantagens apresentadas por essa definição, (John et al., 1994) coloca a necessidade de distinguir entre dois graus de relevância: forte e fraca. A relevância deve ser definida em termos do classificador de Bayes⁵ — o classificador ótimo para um dado problema.

Assim, uma feature X_i é dita *fortemente relevante* se a simples remoção dessa feature resultar na deterioração da performance do classificador de Bayes. Já uma feature X_i é dita *fracamente relevante* se não for fortemente relevante e existir um subconjunto de features, F , tal que a performance do classificador de Bayes sobre F é pior que a performance sobre $F \cup \{X_i\}$. Ambos os graus de relevância são definidos a seguir; observe que a definição de forte relevância é simplesmente a repetição da Definição 2.3.4.

⁵Um classificador de Bayes é uma regra que prevê a mais provável classe para uma dada instância, baseado na distribuição, assumida como sendo conhecida, do conjunto de dados considerado.

Definição 2.3.6 (*Forte Relevância*) (John et al., 1994) Uma feature X_i é fortemente relevante sss existe algum x_i , y e s_i para $p(X_i = x_i, S_i = s_i) > 0$ tal que

$$p(Y = y|X_i = x_i, S_i = s_i) \neq p(Y = y|S_i = s_i)$$

Definição 2.3.7 (*Fraca Relevância*) (John et al., 1994) Uma feature X_i é fracamente relevante sss não é fortemente relevante e existe algum subconjunto S'_i de S_i para o qual existe algum x_i , y e s'_i para $p(X_i = x_i, S'_i = s'_i) > 0$ tal que

$$p(Y = y|X_i = x_i, S'_i = s'_i) \neq p(Y = y|S'_i = s'_i)$$

Em outras palavras, forte relevância implica que a feature é indispensável no sentido de que essa feature não pode ser removida sem a perda de precisão da classificação pelo classificador de Bayes. Por outro lado, fraca relevância implica que a feature pode algumas vezes contribuir para a acurácia da predição. Considera-se que uma feature é relevante se ela é fortemente ou fracamente relevante, caso contrário é dita irrelevante.

Para exemplificar essas definições, considere o mesmo conjunto de exemplos da Tabela 2.3.2 (pg. 9). As features X_1 e X_2 são consideradas como não sendo de forte relevância segundo a Definição 2.3.6 (equivalente à Definição 2.3.4). Porém, aplicando-se a definição de fraca relevância para a feature X_1 tem-se que:

$$p(Y = +|S'_1 = \{\heartsuit\}, X_1 = \clubsuit) \neq p(Y = +|S'_1 = \{\heartsuit\})$$

concluindo-se que X_1 é de fraca relevância, pois $p(Y = +|S'_1 = \{\heartsuit\}, X_1 = \clubsuit) = 1$ e $p(Y = +|S'_1 = \{\heartsuit\}) = \frac{2}{3}$. No entanto, o mesmo não pode ser comprovado para a feature X_2 . Portanto, segundo as definições anteriores, as features X_1 e X_3 seriam consideradas relevantes, enquanto que X_2 seria considerada como uma feature irrelevante.

As definições de forte e fraca relevância definidas por (John et al., 1994) são apresentadas de maneira equivalente por (Blum and Langley, 1997) da seguinte forma:

Definição 2.3.8 (*Forte relevância em relação à amostra/distribuição*) (Blum and Langley, 1997) Uma feature X_i é fortemente relevante para uma amostra S se existe um par de exemplos E_i e E_j , para $i \neq j$, tal que E_i e E_j diferem somente na atribuição de valores a feature X_i e possuem diferentes classes (ou possuem diferentes distribuições das classes se esses aparecem múltiplas vezes em S). Similarmente, a feature X_i é fortemente relevante para a meta c e distribuição $\mathcal{D}$ se existem exemplos E_i e E_j , com probabilidade não nula sobre $\mathcal{D}$, que diferem somente na atribuição de valores a feature X_i e satisfazem $c(E_i) \neq c(E_j)$.

Definição 2.3.9 (*Fraca relevância em relação à amostra/distribuição*) (Blum and Langley, 1997) Uma feature X_i é fracamente relevante para uma amostra S ou para a meta c e distribuição $\mathcal{D}$ se for possível remover um subconjunto de features tal que a feature X_i torna-se fortemente relevante.

Essas noções de relevância são úteis sob o ponto de vista de um algoritmo que deve decidir quais features manter e quais ignorar. É importante manter as features fortemente relevantes, pois, em geral, removê-las significa introduzir ambiguidade na amostra. Já

features fracamente relevantes podem ou não ser importantes dependendo de quais outras features são ignoradas.

Para mostrar como as diversas definições de relevância podem classificar diferentemente e muitas vezes de modo inesperado as mesmas features, a seguir é apresentado um exemplo que utiliza o conceito XOR — Tabela 2.3.6.

Sejam as features X_1, X_2, X_3, X_4 e X_5 booleanas tal que X_4 e X_5 são negações de X_2 e X_3 respectivamente, isto é, $X_4 = \overline{X_2}$ e $X_5 = \overline{X_3}$. O espaço de instâncias é composto por apenas oito exemplos. Assume-se que todos sejam equiprováveis. O conceito meta é definido por:

$$Y = X_1 \oplus X_2 \text{ } (\oplus \text{ denota XOR})$$

Exemplos	X_1	X_2	X_3	X_4	X_5	Classe (Y)
E_1	1	1	1	0	0	0
E_2	1	1	0	0	1	0
E_3	1	0	1	1	0	1
E_4	1	0	0	1	1	1
E_5	0	1	1	0	0	0
E_6	0	1	0	0	1	0
E_7	0	0	1	1	0	1
E_8	0	0	0	1	1	1

Tabela 2.3.6: Exemplos do Conceito Meta $Y = X_1 \oplus X_2$

Note que o conceito meta possui uma expressão booleana equivalente, $Y = X_1 \oplus \overline{X_4}$. As features consideradas relevantes e irrelevantes para esse exemplo, segundo cada uma das definições dadas até agora, são apresentadas na Tabela 2.3.7.

Definição	Features Relevantes	Features Irrelevantes
Definição 2.3.1	X_1	X_2, X_3, X_4, X_5
Definição 2.3.2	Nenhuma	Todas
Definição 2.3.3	Todas	Nenhuma
Definição 2.3.4	X_1	X_2, X_3, X_4, X_5

Tabela 2.3.7: Features Relevantes e Irrelevantes segundo as Definições de Relevância

Segundo a Definição 2.3.1, apenas a feature X_1 é considerada relevante, pois $Y = X_1 \oplus X_2$ ou $Y = X_1 \oplus \overline{X_4}$. Já de acordo com a Definição 2.3.2 todas as features são consideradas irrelevantes pois todos os exemplos são equiprováveis, isto é, para cada valor assumido por Y e X_i , existem dois exemplos que combinam com esses valores. Segundo a Definição 2.3.3 todas as features são consideradas relevantes pois $p(Y = y, S_i = s_i | X_i = x_i) \neq p(Y = y, S_i = s_i)$ para todos os X_i e S_i , isto é, cada valor de X_i restringe o espaço de exemplos à metade e, sendo cada exemplo único, tem-se que $p(Y = y, S_i = s_i | X_i = x_i) = \frac{1}{4}$. Já $p(Y = y, S_i = s_i) = \frac{1}{8}$, pois como anteriormente dito não existem exemplos iguais para a mesma classe. Finalmente, segundo a Definição 2.3.4, X_3 e X_5 são claramente irrelevantes, já que conhecer ou não seus valores não influencia a decisão por uma classe ou por outra. O mesmo ocorre com X_2 e X_4 , pois os valores dessas features não acrescentam nenhuma informação a S_2 e S_4 , respectivamente.

Sob um ponto de vista diferente das definições anteriormente colocadas, em muitos casos, ao invés de determinar exatamente quais features são relevantes, o que se quer é empregar

a relevância como uma *medida de complexidade*. Isto significa que quer-se utilizar a relevância para indicar “o quão complicada” uma função é, isto é, deseja-se que o algoritmo de aprendizado tenha uma boa performance quando essa quantidade for baixa. A definição apresentada a seguir considera essa medida.

Definição 2.3.10 (*Relevância como uma Medida de Complexidade*) (Blum and Langley, 1997) Dada uma amostra de dados S e um conjunto de conceitos C , seja $r(S, C)$ o número de features relevantes segundo a Definição 2.3.1 para um conceito em C que, de todos os conceitos cujo erro sobre S é mínimo, possui o menor número de features relevantes.

Em outras palavras, o que se quer é o menor número de features necessárias para se alcançar uma performance ótima sobre S através de um conceito em C . A razão pela qual especifica-se o conjunto de conceitos C é a possibilidade da existência de uma feature, tal como o CPF de uma pessoa, o qual é altamente relevante sob o ponto de vista do conteúdo da informação, porém essa feature pode ser inútil dependendo do tipo de conceito que está sendo considerado.

As definições de relevância apresentadas até então são independentes do algoritmo de aprendizado que irá utilizar o subconjunto de features selecionado como entrada. É importante observar que não existe nenhuma garantia de que uma feature, ainda que relevante, será útil para um determinado indutor e vice-versa.

Uma outra definição, bastante natural visualiza o problema da seleção de um subconjunto de features reduzindo-o ao problema de encontrar um subconjunto ótimo para um determinado indutor. Esta definição considera o algoritmo de indução, suas heurísticas e seu *bias*.

Definição 2.3.11 (*Relevância em relação a Precisão*) (Kohavi and John, 1997) Dados um indutor $\mathcal{I}$ e uma amostra de dados S com features $X_1, X_2, \dots, X_n$ e uma distribuição $\mathcal{D}$ sobre o espaço de instâncias classificadas, um subconjunto ótimo de features, X_{opt} , é um subconjunto de features tal que a precisão do classificador induzido $\mathcal{C} = \mathcal{I}(\mathcal{D})$ é máxima.

Deve-se notar que um *subconjunto ótimo de features* não precisa ser único, pois é possível alcançar a mesma precisão utilizando diferentes subconjuntos. Um exemplo disso seria a substituição de uma feature por outra que é perfeitamente correlacionada a aquela.

Um dos problemas de utilizar, na prática, essa definição é que a distribuição $\mathcal{D}$ dos exemplos não é conhecida. Assim, é necessário estimar a precisão do algoritmo de indução utilizando a amostra de exemplos. A definição a seguir contempla esse aspecto.

Definição 2.3.12 (*Utilidade Incremental*) (Caruana and Freitag, 1994) Dada uma amostra de dados S , um indutor $\mathcal{I}$ e um conjunto de features F , uma feature X_i é incrementalmente útil para $\mathcal{I}$ em relação a F se a precisão da hipótese produzida por $\mathcal{I}$ considerando o conjunto de features $\{X_i\} \cup F$ é melhor que a precisão alcançada utilizando-se apenas o conjunto de features F .

Essa definição é especialmente natural para os algoritmos de seleção de features que realizam a busca no espaço de subconjuntos de features, adicionando e removendo features desses subconjuntos, tais como as abordagens *embedded* e *wrapper* descritas na Seção 2.4.

É importante ressaltar que, na prática, classificadores gerados por algoritmos de indução podem beneficiar-se da omissão de features, inclusive das features de relevância forte. Assim, é interessante observar que a relevância de uma feature não implica que ela esteja no subconjunto ótimo de features.

Para exemplificar isso, considere o conjunto de exemplos compostos por três features, X_1 , X_2 e X_3 , o qual tem como universo de possíveis exemplos $\{0, 1\}^3$ — Tabela 2.3.8. Seja o conceito meta $c(E) = (X_1 \wedge X_2) \vee X_3$, no qual $E = \{X_1, X_2, X_3\}$. Sob qualquer uma das definições de relevância anteriormente descritas, todas as três features são consideradas relevantes.

Exemplos	X_1	X_2	X_3	Classe
E_1	1	1	1	1
E_2	1	1	0	1
E_3	1	0	1	1
E_4	1	0	0	0
E_5	0	1	1	1
E_6	0	1	0	0
E_7	0	0	1	1
E_8	0	0	0	0

Tabela 2.3.8: Exemplos para Ilustrar que uma Feature ser Relevante não Implica ser Ótima

Nesse caso, porém, o único subconjunto ótimo de features é $\{X_3\}$, pois utilizando X_3 para prever a classe tem-se uma precisão de $\frac{7}{8}$, isto é, a máxima precisão para todos os possíveis subconjuntos de features. Empregando-se qualquer outra feature ou qualquer outra conjunção de features a precisão será no máximo de $\frac{5}{8}$.

Desse modo, uma feature ser considerada relevante não implica que ela estará necessariamente no subconjunto ótimo de features.

2.4 Seleção de um Subconjunto de Features

Nesta seção é descrito um paradigma para a representação da Seleção de Features que a coloca sob o ponto de vista de uma busca heurística, na qual cada estado no espaço de busca é composto por um subconjunto de features. São descritas também três abordagens para a seleção de um subconjunto de features: a abordagem *embedded*, *filtro* e *wrapper*.

2.4.1 Seleção de Features como uma Busca Heurística

Segundo (Blum and Langley, 1997), um paradigma conveniente para a representação das diversas abordagens para a tarefa de seleção de features (principalmente para aquelas que realizam seleção explícita) é a busca heurística, na qual cada estado no espaço de busca é composto por um subconjunto de possíveis features. Desse modo, qualquer método de seleção de features pode ser caracterizado por sua instanciização em relação a quatro questões básicas, as quais determinam a natureza do processo de busca heurística. São elas:

1. o(s) ponto(s) de partida;

2. a organização da busca;
3. a estratégia de busca e
4. o critério de parada.

A primeira questão que deve ser tratada é a determinação do ponto de partida (ou pontos de partida) no espaço de busca. Esta determinação por sua vez influencia a direção em que a busca será realizada e os operadores que serão utilizados para a geração dos estados sucessores. A Figura 2.4.1 (Langley, 1996) ilustra o espaço de busca para quatro features. Pode-se observar que existe uma ordenação parcial dos estados, pois cada um deles possui uma feature a mais que o estado anterior, sendo o estado inicial (mais a esquerda) estabelecido como vazio, isto é, o subconjunto vazio de features. Essa abordagem é geralmente conhecida como seleção *forward*. Já a abordagem que inicia o ponto de partida com o subconjunto contendo todas as features e sucessivamente removendo-as, é denominado de eliminação *backward*. Podem também ser empregadas variações das abordagens mencionadas anteriormente selecionando-se um estado inicial em algum ponto do espaço de busca e movendo-se a partir desse ponto — seleção *outward*.

Figura 2.4.1: Exemplo de Espaço de Estados de Features

A segunda decisão a ser tomada está relacionada a organização da busca. A cada ponto na busca, modificações locais ao conjunto de atributos são consideradas, uma dessas é selecionada e uma nova iteração é realizada. Claramente, uma busca exaustiva em todo o espaço de estados é impraticável, já que para um número a de atributos existem 2^a possíveis estados. Uma abordagem mais realística é a utilização de um método *greedy* para a travessia do espaço de busca; em cada ponto da busca, consideram-se alterações locais sobre o conjunto corrente de features, seleciona-se uma e ocorre uma nova iteração. Por exemplo, a abordagem *hill-climbing*, conhecida como seleção ou eliminação *stepwise*, considera tanto a adição quanto a remoção de features em cada ponto de decisão, permitindo

ainda que a recuperação de uma decisão anterior possa ser realizada. Dentre essas opções, pode-se considerar todos os estados gerados e selecionar o melhor, ou simplesmente escolher o primeiro estado que melhora a precisão sobre o conjunto corrente. Pode-se também substituir o método *greedy* por métodos mais sofisticados como por exemplo o método *best-first*, o qual, embora sendo mais custoso, é ainda tratável para alguns domínios.

A terceira questão a ser tratada considera a estratégia utilizada na avaliação dos subconjuntos alternativos de features. Uma métrica usualmente empregada envolve a habilidade de uma feature discriminar entre as classes que ocorrem num conjunto de treinamento. Diversos algoritmos de indução incorporam um critério baseado na Teoria da Informação, enquanto outros medem diretamente a precisão do conjunto de treinamento ou de um conjunto separado de avaliação.

Outro aspecto importante é como a estratégia de Seleção de Features interage com o algoritmo básico de indução. Essa interação pode ser subdividida em três abordagens (Kohavi and John, 1997):

1. *embedded*, na qual a SF é embutida no algoritmo básico de indução;
2. *filtro*, na qual a SF é utilizada para filtrar features durante um passo de pré-processamento ignorando o algoritmo de indução e
3. *wrapper*, a qual emprega o próprio algoritmo de indução como uma “caixa preta”.

Essas abordagens são discutidas brevemente na Seção 2.4.2.

Finalmente, a quarta decisão a ser tomada refere-se ao critério de parada da busca. Alguns critérios seriam:

- parar de remover ou adicionar atributos quando nenhuma das alternativas melhora a precisão da estimativa para classificação;
- continuar revisando o subconjunto de features enquanto a precisão não se degrada;
- continuar gerando subconjuntos candidatos até que o outro extremo do espaço de busca seja alcançado e escolher o melhor desses subconjuntos;
- parar quando cada combinação de valores para o subconjunto de features selecionado mapear apenas uma classe (assumindo que não há ruído nos dados) e
- ordenar as features segundo alguma pontuação de relevância e utilizar um parâmetro de sistema para determinar o ponto de parada, por exemplo, o número de features desejado para o subconjunto. Essa alternativa é mais robusta que a anterior.

Um aspecto importante do paradigma apresentado para representar as diversas abordagens de seleção de features é que ele é diretamente aplicável a qualquer algoritmo de AM que realiza a sua própria seleção de features.

Para exemplificar o paradigma da busca heurística utilizada para representar a seleção de features, considere os conceitos que podem ser expressados como uma disjunção de features e a estratégia conhecida como algoritmo *greedy*:

Inicie com a disjunção de zero features. Entre as features não presentes em nenhum exemplo negativo (feature dita “segura”) escolha aquela que incluída na hipótese corrente proporcione o maior acréscimo no número de exemplos positivos corretamente classificados. Repetir até que não existam mais features “seguras” que poderiam aumentar o número de exemplos positivos classificados e então pare.

Assim, para a estrutura anteriormente descrita tem-se:

- ponto de partida: ponto mais a esquerda da Figura 2.4.1, isto é, seleção *forward* (iniciando com um subconjunto vazio de features);
- organização da busca: move-se incrementalmente apenas a direita;
- estratégia: avalia os subconjuntos baseados na performance do indutor sobre o conjunto de dados com uma penalidade infinita para a classificação errada de exemplos negativos;
- critério de parada: quando não é mais possível melhorar a performance.

Exemplos	Atributos					Classe
	X_1	X_2	X_3	X_4	X_5	
E_1	1	0	0	0	0	+
E_2	1	1	0	0	0	+
E_3	0	0	1	1	0	+
E_4	0	0	0	0	1	+
E_5	0	0	0	0	0	−

Tabela 2.4.1: Amostra de Exemplos para Ilustrar o Paradigma da Busca Heurística para a Seleção de Features

Então, considerando os cinco exemplos fornecidos pelo conjunto de dados apresentados na Tabela 2.4.1, o algoritmo apresentado anteriormente selecionaria primeiramente a feature X_1 , depois provavelmente X_3 , depois provavelmente a feature X_5 e pararia.

2.4.2 Abordagens para a Seleção de Features

Como mencionado anteriormente, as abordagens para a seleção de features podem ser agrupadas em: *embedded*, *filtro* e *wrapper* descritas a seguir (Baranauskas and Monard, 1998). Essas três abordagens realizam a SF de forma explícita.

2.4.2.1 *Embedded*

Nessa abordagem a tarefa da seleção de features é realizada internamente pelo próprio algoritmo. Em outras palavras, a seleção de features está embutida no algoritmo de AM como ilustra a Figura 2.4.1.

A maioria dos algoritmos pertencentes a classe de métodos *eager* possuem uma abordagem *embedded* para a seleção de features. Os métodos *eager* gulosamente substituem os

Figura 2.4.2: Abordagem *Embedded*

exemplos de treinamento pelo conceito aprendido, geralmente na forma de conjunto de regras, árvore de decisão ou rede neural. Posteriormente, somente o conceito aprendido é utilizado para classificar novos exemplos. São exemplos desses métodos os algoritmos CN2 (Clark and Niblett, 1989), C4.5 (Quinlan, 1993) e ID3 (Quinlan, 1983).

Em contraste, os algoritmos pertencentes a classe de métodos *lazy* utilizam o próprio conjunto de treinamento para classificar novos exemplos. Esses métodos exibem as seguintes características:

- adiam o processamento do conjunto de treinamento até que uma nova consulta (classificar um novo exemplo) seja requisitada; simplesmente armazenam seu conjunto de treinamento;
- são dirigidos pela demanda, isto é, atendem a requisições de consultas através da combinação da informação armazenada nos exemplos de treinamento e
- não armazenam a consulta construída e também não armazenam quaisquer resultados intermediários.

Alguns exemplos de algoritmos *lazy* são: *Nearest-Neighbor* — NN — e *Instance-Based* — IB (Mitchell, 1997).

Para ilustrar essas duas classes de métodos, considere os algoritmos NN e ID3.

Em linhas gerais, dado um conjunto de treinamento e um novo exemplo a ser classificado, o que o algoritmo NN faz é calcular, segundo alguma métrica, qual o exemplo armazenado em seu classificador que está mais próximo do novo exemplo a ser classificado, sendo a classe desse atribuída ao novo exemplo. Desse modo, o classificador gerado por um algoritmo NN é composto, de uma forma simplificada, pelo conjunto de treinamento e pela métrica utilizada para o cálculo da distância entre os exemplos.

O algoritmo ID3 aprende árvores de decisão, construindo-as de uma forma *top-down*, iniciando com a pergunta:

Qual feature deveria ser considerada na raiz da árvore ?

Para responder essa pergunta, cada feature do conjunto de treinamento é avaliada utilizando algum critério para determinar o quão bem essa feature, por si só, classifica os exemplos de treinamento. A melhor feature, segundo esse critério, é atribuída a raiz da árvore e o processo é repetido para o restante da árvore. As folhas da árvore contêm o valor das classes. No fim desse processo, os exemplos de treinamento são excluídos do classificador.

Segundo (Aha, 1998), métodos *eager* assumem que seu *bias* é apropriado para a tarefa a ser realizada. Quando essa suposição está correta têm-se benefícios, tais como o aumento da velocidade do tempo de resposta à consulta. Porém, existe o risco dessa suposição estar errada e informação importante para a geração de respostas mais precisas ser perdida durante a abstração do conceito. Desse modo, uma das principais vantagens de algoritmos *lazy* é a capacidade de responder a consultas nunca antes realizadas em uma forma não disponível para todos os algoritmos *eager*.

Uma outra vantagem computacional que pode ser citada para métodos *lazy* é o seu pequeno custo de treinamento, por exemplo, armazenar um exemplo e atualizar alguns índices, apesar disso ser freqüentemente contra-balanceado com custos mais altos para a geração de predições (a menos que uma estrutura eficiente de indexação seja implementada).

Assim, pode-se notar que a principal diferença entre os métodos *lazy* e *eager* é que o primeiro permite a opção da seleção de uma hipótese diferente a cada novo exemplo a ser classificado, ou permite a seleção de uma aproximação local para a classe de cada novo exemplo a ser classificado. O segundo está comprometido com uma única hipótese que cobre todo o espaço de instâncias (Mitchell, 1997).

2.4.2.2 *Filtro*

Essa abordagem de seleção de features introduz um processo separado, o qual ocorre antes da aplicação do algoritmo de indução propriamente dito — Figura 2.4.3. A idéia é filtrar atributos irrelevantes, segundo algum critério, tais como os descritos na Seção 2.4.1, antes da indução ocorrer (John et al., 1994). Esse passo de pré-processamento considera características gerais do conjunto de dados para selecionar algumas features e excluir outras. Sendo assim, métodos de filtros são independentes do algoritmo de indução que, simplesmente, receberá como entrada a saída fornecida pelo filtro.

Segundo (Blum and Langley, 1997), um dos esquemas mais simples de filtragem é a avaliação de cada feature individualmente, baseada na sua correlação com o conceito meta, escolhendo as k features que fornecem o melhor valor. Esse método é comumente empregado em tarefas de categorização de textos, geralmente combinando o filtro com os algoritmos *Naive Bayes* (Mitchell, 1997) ou NN, os quais tem mostrado bons resultados empíricos.

O algoritmo RELIEF (Kira and Rendell, 1992) segue esse esquema de filtragem, mas incorpora uma função de avaliação de features mais complexa.

Figura 2.4.3: Abordagem *Filtro*

Já (Almuallim and Diettrich, 1991) descrevem uma abordagem filtro, o algoritmo FOCUS, o qual envolve um grau maior de busca no espaço de features. Esse algoritmo procura por combinações mínimas de atributos que descrevem as classes. Desse modo, esse método inicia a busca examinando cada feature em separado, depois examina pares de features, triplas e assim por diante, até encontrar uma combinação que gera partições puras do conjunto de treinamento, isto é, nas quais nenhuma instância tem diferentes classes.

Porém, a principal diferença entre esses dois algoritmos é que enquanto o FOCUS procura por um conjunto mínimo de features, o algoritmo RELIEF procura por todas as features relevantes, isto é, tanto as fortemente quanto as fracamente relevantes — Figura 2.4.4.

Embora ambos os algoritmos RELIEF e FOCUS realizem a seleção de features através da construção de árvores de decisão, outros métodos de indução podem também ser utilizados, como por exemplo no filtro descrito por (Cardie, 1993), o qual emprega o algoritmo NN.

2.4.2.3 *Wrapper*

Esta abordagem também ocorre externamente ao algoritmo básico de indução, porém utilizando tal algoritmo como uma caixa preta para analisar o conjunto de features fornecido — Figura 2.4.5. Em outras palavras, métodos *wrapper* geram um subconjunto candidato de features, executam o algoritmo de indução considerando apenas esse subconjunto de features selecionado do conjunto de treinamento, e utilizam a precisão resultante do classificador gerado para avaliar o subconjunto de features em questão. Esse processo é repetido para cada subconjunto de features até que o critério de parada determinado seja satisfeito.

Figura 2.4.4: Uma Visão da Relevância do Conjunto de Features

Um argumento utilizado com muita frequência para apoiar a utilização da abordagem *wrapper* é que o mesmo algoritmo de indução que vai usar o subconjunto de features selecionado deve prover uma estimativa melhor de precisão que um outro algoritmo, o qual pode possuir um *bias* de indução totalmente diferente (Kohavi and John, 1997).

Porém, a maior desvantagem dos métodos *wrapper* é o custo computacional, o qual resulta da execução do algoritmo de indução para avaliar cada subconjunto de features a ser considerado (Kohavi and John, 1997).

Na próxima seção é introduzida a segunda abordagem de Seleção de Features denominada Indução Construtiva ou Construção de Features.

2.5 Indução Construtiva

Algoritmos de aprendizado indutivo convencionais dependem de dados fornecidos pelo usuário para construir suas descrições. Linguagem de descrição ou espaço de representação inadequados podem tornar difíceis os problemas de aprendizado.

As features podem ser consideradas inadequadas para a tarefa de aprendizado quando são fracamente ou indiretamente relevantes, condicionalmente relevantes ou medidas de modo inapropriado (Lee, 1999; Baranauskas and Monard, 1999; Baranauskas et al., 1999). Se as features utilizadas para a descrição do conjunto de dados de treinamento são inadequadas, os algoritmos de aprendizado provavelmente criarão descrições imprecisas ou excessivamente complexas (Bloedorn and Michalski, 1998).

Porém, essas features, individualmente inadequadas, podem algumas vezes serem convenientemente combinadas gerando novas features que podem mostrar-se altamente representativas para a descrição de um conceito. O processo de construção de novas features é conhecido como Construção de Features ou Indução Construtiva — IC⁶ (Michalski,

⁶Os termos Aprendizado Construtivo ou Transformação de Features também são encontrados na literatura.

Figura 2.4.5: Abordagem *Wrapper*

1978).

Assim, IC é o processo de composição de features ditas primitivas⁷, produzindo-se novas features possivelmente relevantes para a descrição de um conceito. Em outras palavras, IC pode ser definido como:

A aplicação de operadores construtivos, *i.e.* operadores usados para compor features a partir de features existentes, resultando na definição de uma ou mais features.

É importante notar que, diferentemente da seleção de um subconjunto de features, na qual somente as features selecionadas são mostradas ao algoritmo, decrementando assim o espaço de busca de features, a Indução Construtiva aumenta o espaço de busca de features.

Indução Construtiva requer respostas para as seguintes questões:

Quais operadores construtivos utilizar? e

Quais os operandos (features primitivas) desses operadores?

Uma outra importante observação é que, em geral, o processo de construção de features é intratável, já que o número de features que podem ser construídas é uma função combinatória do número de features existentes multiplicado pelo número de operadores. Conseqüentemente, a construção de features, somente é viável, quando articulada a heurísticas que possibilitam a redução dos operandos e operadores construtivos a serem utilizados para construir novas features que simplifiquem a descrição do conceito em foco.

⁷Features pertencentes ao conjunto de dados original.

2.5.1 Abordagens para Indução Construtiva

Os métodos de Indução Construtiva podem ser agrupados de acordo com a informação que é utilizada para a busca do melhor espaço de representação (Bloedorn and Michalski, 1998; Wnek and Michalski, 1994; Wnek and Michalski, 1993). São eles:

1. IC guiada pelos dados (*data-driven constructive induction*), a qual é baseada na análise do conjunto de dados de treinamento;
2. IC guiada pela hipótese (*hypothesis-driven constructive induction*), a qual é baseada na análise da hipótese indutiva. Nessa abordagem, conceitos úteis podem ser extraídos, por exemplo regras, e usados na definição de novas features;
3. IC guiada pelo conhecimento (*knowledge-driven constructive induction*), a qual é baseada no conhecimento do domínio provido por um especialista e
4. IC multi-estratégico (*multistrategy constructive induction*), a qual combina dois ou mais dos métodos anteriores.

O processo de construção de features pode ser guiado e controlado pelo usuário/especialista ou pode ser conduzido automaticamente pelo sistema de aprendizado.

Nesse trabalho, é focada a Indução Construtiva guiada pelo conhecimento, a qual é baseada no conhecimento do domínio provido por um usuário/especialista.

2.5.2 Um Exemplo de Indução Construtiva

Para exemplificar o processo de construção de features, considere o exemplo clássico dos robôs amigos e inimigos. Nesse exemplo, as features *Cabeça*, *Corpo*, *Segura* e *Sorri* são utilizados para determinar se um robô é amigo ou inimigo. A Tabela 2.5.1 mostra o conjunto de exemplos.

Exemplos	Cabeça	Corpo	Sorri	Segura	Classe
E_1	quadrada	quadrado	sim	balão	amigo
E_2	quadrada	triangular	não	espada	inimigo
E_3	redonda	redondo	sim	bandeira	amigo
E_4	triangular	redondo	sim	espada	inimigo
E_5	triangular	triangular	sim	balão	amigo
E_6	redonda	quadrado	não	bandeira	inimigo

Tabela 2.5.1: Exemplos de Robôs Amigos e Inimigos

Construindo a árvore de decisão para esse conjunto de exemplos — Figura 2.5.1 — pode-se notar que das cinco features apresentadas, duas são consideradas para decidir se um robô é amigo ou inimigo, gerando uma árvore de dois níveis de profundidade.

Porém, construindo uma nova feature denominada *Mesma-forma* a partir da comparação dos valores das features *Cabeça* e *Corpo* (features primitivas) — Tabela 2.5.2 — pode-se observar que utilizando apenas essa nova feature é possível distinguir entre robôs amigos e inimigos.

Assim, como ilustra a árvore de decisão e as respectivas regras de decisão geradas após a composição dessa nova feature — Figura 2.5.2 — um robô é amigo apenas se sua cabeça

Se sorri = sim e
segura = espada
então inimigo.

Se sorri = sim e
segura = balão ou
bandeira
então amigo.

Se sorri = não
então inimigo.

Figura 2.5.1: Árvore e Regras de Decisão para o Exemplo de Robôs Amigos e Inimigos

Exemplos	Cabeça	Corpo	Sorri	Segura	Mesma-forma	Classe
E_1	quadrada	quadrado	sim	balão	v	amigo
E_2	quadrada	triangular	não	espada	f	inimigo
E_3	redonda	redondo	sim	bandeira	v	amigo
E_4	triangular	redondo	sim	espada	f	inimigo
E_5	triangular	triangular	sim	balão	v	amigo
E_6	redonda	quadrado	não	bandeira	f	inimigo

Tabela 2.5.2: Exemplos de Robôs Amigos e Inimigos depois da Construção de Features

e corpo têm o mesmo formato, caso contrário ele é inimigo. Pode-se notar que nenhuma das features primitivas foi explicitamente necessária para a indução de um conceito muito mais simples.

Se mesma_forma = v
então amigo.

Se mesma_forma = f
então inimigo.

Figura 2.5.2: Árvore e Regras de Decisão para o Exemplo de Robôs Amigos e Inimigos após a Construção de Features

2.6 Considerações Finais

A noção de relevância pode variar dependendo do objetivo a ser alcançado. Por essa razão, são encontradas na literatura diferentes definições de relevância de features. Algumas

dessas definições consideram, por exemplo, a relevância em relação ao conceito meta, ou a relevância em relação a probabilidade de distribuição enquanto outras consideram a relevância em relação a precisão da classificação alcançada utilizando-se um determinado subconjunto de features.

A procura por features relevantes, ou Seleção de Features pode ser tratada de duas formas: Seleção de um Subconjunto de Features e Indução Construtiva.

A busca heurística é um paradigma que representa de maneira conveniente as diversas abordagens de SSF, pois cada estado no espaço de busca é representado por um subconjunto de possíveis features. Nessa representação, qualquer método de SSF pode ser caracterizado por: ponto(s) de partida, organização da busca, estratégia de busca e critério de parada. Também, considerando a maneira como a estratégia de SSF interage com o algoritmo básico de indução, pode-se agrupar essa interação em: abordagem *embedded*, abordagem *filtro* e abordagem *wrapper*.

Ainda nesse capítulo, foi descrita de forma breve a construção de features. Nessa abordagem, contrariamente a seleção de um subconjunto de features, o conjunto original de features é incrementado através da composição de novas features a partir das features primitivas.

Um aspecto importante a ser ressaltado é a inexistência de uma análise matemática que permita a avaliação do desempenho de um método de aprendizado sobre um conjunto de dados. Isso faz com que estudos experimentais sejam muito importantes na área de Aprendizado de Máquina. Assim, também para a tarefa de Seleção de Features é importante que estudos empíricos sejam realizados sobre os conjuntos de dados de interesse, a fim de determinar que métodos de AM são mais apropriados, ou apresentam o melhor desempenho, para esses conjuntos de dados em particular. A importância desses estudos experimentais deve ser refletida na seriedade das execuções desses experimentos e na forma correta de relatar os resultados, os quais devem ser descritos detalhadamente de modo que outros pesquisadores possam repetir esses estudos.

Neste trabalho foram realizadas diversas avaliações empíricas de Seleção de Features. No próximo capítulo são descritos os algoritmos e as ferramentas utilizados no desenvolvimento desses estudos experimentais, bem como os conjuntos de dados utilizados nesses experimentos.