Meta-Aprendizado para Recomendação de Algoritmos

Bruno F. de Souza, Ricardo B. C. Prudêncio, André de Carvalho

Abstract

The amount of data generated from different sources has increased in large scales over the years. The need for automatically analyzing these data has demanded new methods of Machine Learning and Data Mining that can deal with the problems associated to this challenge. A difficulty observed in Machine Learning applications is to recommend the most adequate algorithms to acquire knowledge from a given dataset. The issue of algorithm recommendation is investigated in the research field called Meta-Learning. Different Meta-Learning techniques have been proposed and successfully adopted to perform recommendations for classic tasks of Machine Learning (e.g., pattern classification). This course presents the basic concepts and most common techniques for using Meta-Learning in real problems. As a result, the authors expect to form competent and autonomous researchers to deal with this topic.

Resumo

A quantidade de dados gerados por diferentes fontes tem crescido em escalas cada vez maiores. A necessidade de analisar tais dados de forma automática demanda o desenvolvimento de novos métodos de Aprendizado de Máquina e de Mineração de Dados capazes de lidar com os problemas associados a esse desafio. Uma das dificuldades encontradas em Aprendizado de Máquina é a recomendação dos algoritmos mais promissores para adquirir conhecimento útil a partir de um conjunto de dados específico. A questão de recomendação de algoritmos é investigada na linha de pesquisa denominada Meta-Aprendizado. Diferentes técnicas de Meta-Aprendizado foram propostas e adotadas com sucesso para realizar recomendações em tarefas clássicas de Aprendizado de Máquina (por exemplo, classificação de padrões). Este curso apresentará os conceitos básicos e as técnicas mais comuns para o uso de Meta-Aprendizado em problemas reais. Como resultado, espera-se a formação de pesquisadores que possam atuar com competência e autonomia nesse tema.

1.1. Introdução

Com o avanço do Aprendizado de Máquina (AM) [Carbonell et al. 1984, Simon 1984, Michalski 1986], vários métodos e algoritmos têm se tornado disponíveis. Para que ferramentas baseadas nessas abordagens resultem em soluções apropriadas às necessidades do usuário, elas devem ser cuidadosamente escolhidas. Segundo o teorema *No Free Lunch* (NFL) [Schaffer 1994], qualquer vantagem apresentada por um algoritmo sobre uma classe específica de problemas é compensada quando de sua aplicação em outra classe,

no caso em que outro algoritmo resultará mais adequado. Sendo assim, se todos os problemas forem igualmente possíveis, então os algoritmos tenderão a apresentar, na média, o mesmo comportamento preditivo. Claramente, o NFL tem caráter geral e, na prática, um algoritmo pode resultar superior em determinados problemas. Portanto, a definição do algoritmo mais satisfatório a cada aplicação deve ser feita de maneira pontual.

A abordagem mais comum para esta tarefa envolve um processo iterativo que combina a experiência do usuário e alguma forma de investigação empírica. Nesse caso, um subconjunto de algoritmos é inicialmente selecionado de acordo com o conhecimento de um especialista sobre os algoritmos e o problema. Tais métodos são então aplicados ao problema e uma medida de desempenho é calculada. Com base nas informações obtidas, decide-se por alterar a configuração do algoritmo, ou ele próprio, até que resultados satisfatórios sejam alcançados. Essa abordagem apresenta duas grandes desvantagens. A primeira é que o conhecimento do usuário geralmente tem baixo nível de confiança ou é muito limitado. A segunda é que a utilização de usuários especialistas é custosa e a experimentação é um processo demorado. Em vista disso, destaca-se a importância do desenvolvimento e do emprego de métodos eficientes que auxiliem o usuário nessa tarefa.

Conforme visto no Capítulo 2 de [Brazdil et al. 2009], o propósito de um sistema de recomendação de algoritmos, do ponto de vista de seu utilizador, é reduzir o número de algoritmos testados a fim de otimizar o tempo de experimentação com um mínimo de perda na qualidade dos resultados obtidos. Assim, dado um novo problema, o sistema deve ser capaz de sugerir os algoritmos mais apropriados, ou no mínimo razoáveis, para encontrar a sua solução. Caso haja recursos computacionais disponíveis, então outras alternativas potencialmente adequadas devem também ser fornecidas, assim como sua ordem de utilização. Desta forma, há uma necessidade por um entendimento maior entre as propriedades dos problemas e o desempenho dos algoritmos para que a seleção destes seja realizada de acordo com as características daqueles.

Ao encontro desse objetivo, a comunidade de AM introduziu as bases conceituais do meta-aprendizado, que fornece meios para esse fim. De maneira genérica, o meta-aprendizado pode ser entendido como a utilização de técnicas de AM para a construção de modelos que expliquem o relacionamento entre estratégias de aprendizado e problemas, segundo alguma perspectiva definida [Vilalta et al. 2005]. Sendo assim, ele consegue explorar conhecimento acumulado sobre diversas tarefas e aplicá-lo para a resolução de problemas semelhantes. Com isso, pretende-se determinar sob quais condições cada algoritmo é mais apropriado, possivelmente ampliando o entendimento do mesmo e levando a sugestões de uso mais adequadas. Do ponto de vista prático, o meta-aprendizado ainda tem sido pouco explorado, apesar de ter o potencial de transformar a forma como o AM é empregado em setores industriais, comerciais e governamentais.

O objetivo deste trabalho é apresentar os conceitos básicos e as técnicas mais comuns para o uso de meta-aprendizado para a recomendação de algoritmos. Dados o patente interesse no assunto e a maior disponibilidade de trabalhos relacionados (vide referências em [Brazdil et al. 2009]), o viés da exposição é para a seleção de algoritmos de classificação. Assim, na Seção 1.2, apresenta-se uma introdução a vários aspectos importantes de AM supervisionado, como os principais paradigmas, as medidas de desempenho e as estratégias de avaliação de classificadores, e as metodologias de comparação dos mesmos. Na Seção 1.3, apresenta-se o meta-aprendizado de maneira genérica e sua utilização para a recomendação de algoritmos. São abordados os tópicos de caracterização de conjuntos de dados, medidas de avaliação, formas de sugestão e construção da sugestão. As bases conceituais deste trabalho são desenvolvidas nessa seção. Na Seção 1.4, apresenta-se três estudos de caso desenvolvidos a fim de ilustrar o emprego de meta-aprendizado. O primeiro é no contexto de classificação de dados de expressão gênica e apresenta detalhamento maior. Os demais lidam com seleção de séries temporais e otimização combinatória. Por fim, na Seção 1.5, há a conclusão do trabalho e uma breve discussão sobre os novos desafios em meta-aprendizado.

1.2. Aprendizado de Máquina Supervisionado

O Aprendizado de Máquina (AM) (introdução abrangente sobre diversos tópicos no assunto está disponível em [Mitchell 1997]) pode ser considerado um campo de pesquisa fundamentado na Inteligência Artificial e na Estatística que estuda e modela as diversas facetas do processo de aprendizado. Seu surgimento foi motivado pela observação de que, em sistemas biológicos, a inteligência está intrinsicamente relacionada à capacidade de aprender. Os seres humanos, por exemplo, são capazes de adquirir conhecimentos, desenvolver novas habilidades e melhorar seu desempenho com a prática. Assim, as pesquisas em AM buscam compreender esses mecanismos naturais e/ou reproduzi-los em sistemas artificiais aptos para lidar com problemas reais.

Diversas estratégias de aprendizado podem ser utilizadas para desenvolver algoritmos de AM. A mais estudada atualmente baseia-se no conceito de indução, segundo o qual é possível obter-se conclusões genéricas a partir de fatos ou observações particulares. Esse tipo de inferência lógica caracteriza-se por extrapolar a informação contida nos dados a fim de modelar conceitos mais gerais. Quanto mais representativos do conceito geral forem os dados, melhor a qualidade da modelagem, embora não haja garantias de que as conclusões do raciocínio indutivo sejam sempre verdadeiras. Em todo caso, a indução representa uma importante ferramenta para a geração de novos conhecimentos.

O aprendizado indutivo tem sido tradicionalmente empregado para a concepção de abordagens de AM segundo duas vertentes básicas: supervisionada e não-supervisionada. Na primeira, o objetivo é induzir descrições gerais de conceitos utilizando exemplos específicos dos mesmos. Neste contexto, uma tarefa usual de algoritmos supervisionados é, dado um conjunto rotulado

de exemplos, encontrar uma função capaz de predizer a classe ou o valor associado a um novo exemplo, com base nos atributos descritores do mesmo. Na segunda, a meta é descobrir padrões e regras gerais que consigam explicar as observações. Comumente, algoritmos não-supervisionados são aplicados para analisar os dados e tentar identificar, caso haja, estruturas de grupos. Outra aplicação é buscar por padrões frequentes de associação entre os atributos de um conjunto de dados. Mais recentemente, algoritmos que utilizam conceitos de ambas as vertentes têm sido propostos segundo o paradigma de aprendizado semi-supervisionado. Nele, dados não rotulados são utilizados para modificar ou refinar os modelos construídos a partir de dados rotulados. Essa abordagem pode ser vantajosa em aplicações em que a rotulação de exemplos é difícil, custosa ou demanda muito tempo, e há grande disponibilidade de dados não rotulados. Um apreciação sobre esse novo tipo de algorimos pode ser encontrada em [Chapelle et al. 2006]

O enfoque desta seção é o AM supervisionado, dado sua relevância para o presente trabalho. Na Seção 1.2.1, são apresentados algumas de suas noções básicas. Na Seção 1.2.2, os paradigmas de classificação são brevemente comentados. Na Seção 1.2.3, algumas formas usuais de estimar o desempenho dos classificadores são descritas. Na Seção 1.2.4, os testes estatísticos comumente utilizados para aferir a diferença de desempenho entre os classificadores são abordados.

1.2.1. Noções básicas

Um conceito pode ser entendido como uma regra que particiona os objetos ou exemplos de um domínio de acordo com a obediência (ou não) deles a ela. A tarefa do AM supervisionado é induzir tais conceitos a partir de exemplos específicos dos mesmos. Os exemplos são rotulados por um supervisor ou professor, que detém o conhecimento do domínio e conhece a definição do conceito a ser aprendido. O supervisor fornece os exemplos na forma de pares entrada e saída desejada e o algoritmo aprende, então, uma relação entre as características das entradas e das saídas que seja consistente com os exemplos considerados e que possa ser utilizado para predizer saídas corretas para entradas não vistas anteriormente.

Mais formalmente, considere um conjunto S formado por n elementos dispostos em um espaço de m dimensões tal que $S = \{(\overrightarrow{x_1}, y_1), ..., (\overrightarrow{x_n}, y_n)\}$. Esse conjunto ou base de dados é composto por exemplos $\overrightarrow{x_i}$ com valores de atributos $x_{ij} \in X$ e de classes $y_i \in Y$, amostrados do domínio seguindo uma distribuição D fixa, desconhecida e arbitrária. Os exemplos em S são utilizados pelo algoritmo de AM para a construção de um classificador, chamado também de hipótese ou preditor, capaz de predizer os valores de y_i para novos $\overrightarrow{x_i}$. Esse processo de construção ou indução de um classificador é denominado treinamento.

Na prática, as hipóteses geradas durante o treinamento são comumente representadas por um mapeamento $h : X \mapsto Y$ que aproxime, idealmente, a

verdadeira função $f(X)$. Uma vez escolhida uma função de mapeamento h, definida no espaço de hipóteses H, pode-se classificar elementos não vistos durante a fase de treinamento, por meio do cômputo de $h(\overrightarrow{x_i})$, para um $\overrightarrow{x_i}$ qualquer. Esse mapeamento só é possível sob a condição de que os exemplos desconhecidos ao classificador (coletivamente referidos como conjunto de teste) sejam gerados pela mesma distribuição D do conjunto S.

Em AM, os exemplos $\overrightarrow{x_i}$, conhecidos ainda como casos, padrões, instâncias ou registros, representam um objeto particular do mundo real. Eles são descritos por tuplas de m atributos $x_{ij} \in R$, cada um indicando uma característica ou aspecto do exemplo. De maneira geral, há dois tipos de atributos: quantitativos e qualitativos. Os primeiros representam grandezas numéricas comumente resultantes de medições, enquanto os segundos correspondem a conceitos categóricos ou simbólicos de maior nível de abstração. Detalhes acerca propriedades de atributos podem ser encontrados no Capítulo 2 de [Jain e Dubes 1988].

A cada exemplo $\overrightarrow{x_i}$ é associado um valor especial y_i, que representa o fenômeno de interesse. Ele pode assumir valores contínuos ou discretos. No primeiro caso, tem-se $y_i \in Y = \{c_1, c_2, ..., c_{max}\}$, onde max indica o número de classes existentes. Problemas de aprendizado desse tipo são conhecidos como problemas de classificação. No segundo caso, os valores se apresentam de forma contínua, tal que $y \in Y = \Re$. A tarefa de aprendizado agora é conhecida como regressão.

A indução de um classificador a partir de um conjunto de dados pode ser vista como um problema de busca, em que o objetivo é encontrar a hipótese, entre todas que o algoritmo de AM é capaz de gerar, com a melhor capacidade de descrever o fenômeno a aprender. Como normalmente várias hipóteses são capazes de modelar o conceito, é necessário algum tipo de viés para guiar o processo de busca. O termo viés refere-se a qualquer critério de preferência do algoritmo por uma hipótese em relação a outra, dado que ambas sejam consistentes com os exemplos. De fato, aprendizado sem viés não é possível. Uma discussão sobre a importancia do viés para o aprendizado pode ser encontrada no Capítulo 2 de [Mitchell 1997].

Um viés inadequado ao problema ou um conjunto de dados pouco representativo pode afetar a nível de generalização do classificador, ou seja, sua habilidade em predizer corretamente as classes de exemplos do conjunto de teste. Dentre os problemas de generalização mais comuns, tem-se o *overfitting* e o *underfitting*. O *overfitting* ocorre quando a hipótese induzida é demasiado específica, ou seja, representa pormenores da amostra de dados usada para treinamento que não são característicos do fenômeno que os gera. Se tal acontecer, o desempenho do classificador, segundo uma métrica arbitrária de qualidade, é muito bom para exemplos conhecidos mas insatisfatório considerando a totalidade de exemplos da distribuição D. No caso do *underfitting*, o algoritmo de AM não consegue encontrar uma hipótese adequada que possa modelar devidamente os dados. Isso ocorre porque os dados não contêm in-

formação suficiente acerca do conceito a ser aprendido ou os parâmetros do algoritmo não estão corretamente ajustados para permitir o aprendizado ou o viés do algoritmo de AM é inadequado.

1.2.2. Paradigmas de AM Supervisionado

Há diversos algoritmos capazes de induzir a hipótese h utilizando uma base de dados. Para fins didáticos, é conveniente agrupá-los em paradigmas, segundo suas características. Embora vários autores tenham trabalhado no assunto, ainda não há uma taxonomia considerada consenso por eles. Neste trabalho, por simplicidade, optou-se por seguir a divisão apresentada por [Batista 2003], segundo a qual os paradigmas de AM supervisionado mais comuns são: simbólico, estatístico, baseado em exemplos e conexionista. Eles serão brevemente comentados a seguir.

As técnicas simbólicas operam construindo representações em alto nível de um conceito por meio da análise de exemplos desse conceito, de forma que elas possam ser facilmente interpretadas pelos seres humanos. Dentre os algoritmos mais utilizados, pode-se citar os algoritmos para indução de árvores de decisão [Quinlan 1993] e para indução de regras [Michalski 1969]. Os algoritmos do primeiro caso baseiam-se na estratégia de dividir para conquistar a fim de gerar a árvore de decisão. Genericamente, eles operam particionando os exemplos de dados de acordo com testes realizados nos atributos, tal que cada teste é representado por um nó na hierarquia gerada e os ramos correspondem aos possíveis valores dos atributos. O processo é repetido recursivamente até que folhas representando as classes do problema sejam obtidas. Para classificar um novo exemplo, é preciso apenas percorrer a árvore até uma folha e atribuir a classe correspondente. Por sua vez, os algoritmos mais comuns do segundo caso trabalham em um esquema de separar e conquistar. Nele, regras baseadas nos valores dos atributos são geradas para explicar as classes dos exemplos. Portanto, elas são capazes de abranger certos exemplos e excluir outros. Os exemplos que satisfazem as regras atuais são separados e o processo de aprendizado continua recursivamente sobre os demais, até que todos sejam abrangidos. Novos exemplos são classificados apresentando-os ao conjunto de regras gerado e observando-se a classes daquelas que são satisfeitas. Uma exposição detalhada acerca de algoritmos indutores de árvores e de regras pode ser encontrada nos Capítulos 3 e 10 de [Mitchell 1997], respectivamente.

Conceitos e ferramentas oriundos da estatística têm sido empregados para o desenvolvimento de técnicas capazes de aproximar a hipótese h. Dentre as abordagens de inspiração estatística comumente consideradas, destacam-se as *Support Vector Machines* (SVMs) [Boser et al. 1992] e o classificador *Naïve Bayes* [Kononenko 1990]. Os classificadores Bayesianos buscam minimizar o custo de classificação para um exemplo assinalando a ele a classe *a posteriori* mais provável dados seus atributos. Através da utilização do teorema de Bayes, o problema pode ser reformulado utilizando-se as probabilidades de

ocorrência de cada classe e as probabilidades condicionais de todos os atributos juntos dadas as classes. Ambos os termos podem ser estimados com base nos dados de treinamento. Entretanto, a estimação do segundo termo exige uma grande quantidade de exemplos, o que usualmente não está disponível. O *Naïve Bayes* contorna o problema considerando independentes os atributos e estimando diretamente dos dados as probabilidades necessárias. Embora essa solução pareça simplista para situações reais, ela tem apresentado bons resultados empíricos. Métodos mais sofisticados são ilustrados aqui pelas SVMs, que construem classificadores considerando tanto a complexidade do modelo quanto o desempenho durante o treinamento e implementam princípios matemáticos que minimizam limites teóricos do erro de generalização esperado. Na prática, as SVMs buscam o hiperplano que maximiza a margem de separação entre exemplos de classes distintas. Intuitivamente, percebe-se que quanto mais larga a margem, maior a confiança que o classificador tem em sua previsão. Assim, espera-se que ele consiga ser robusto também quando da apresentação de exemplos de teste. Caso os dados não sejam linearmente separáveis, eles podem ser projetados, de forma implícita, em um espaço de dimensão maior onde o hiperplano otimo possa ser encontrado. Excelente material introdutório sobre SVMs pode ser encontrado em [Burges 1998]. As abordagens Bayesianas para classificação são largamente discutidas no Capítulo 6 de [Mitchell 1997].

As abordagens baseadas em exemplos consideram informações locais sobre a vizinhança dos dados a fim de realizar sua classificação. O algoritmo mais comum desse paradigma é o *k Nearest Neighbors* (kNN) [Fix e Jr 1951]. Ele opera identificando a similaridade entre exemplos e realizando suas predições de acordo com algum esquema de votação/ponderação das classes dos vizinhos mais próximos. Um caso simples de seu funcionamento é ilustrado a seguir. Inicialmente, são calculadas sa distâncias entre um exemplo de teste e todos os dados de treinamento disponíveis. Tipicamente, utiza-se nesta etapa a distância Euclidiana. Em seguida, o método verifica a quais classes pertencem os k exemplos mais próximos, sendo k um parâmetro livre a ser definido de acordo com a aplicação. Por fim, a classe mais frequente entre os k exemplos é utilizada para classificar o novo caso. Esse esquema, apesar de simples, resulta adequado em diversos problemas. Variações mais sofisticadas do método básico podem ser empregadas. Por exemplo, pode-se considerar a influência de cada exemplo na classificação do novo caso de acordo com as distâncias calculadas, tal que exemplos mais próximos tenham maior peso na decisão final. Outra possibilidade é ponderar os atributos dos exemplos com base em sua capacidade preditiva esperada. Uma compilação sobre diversas extensões desse classificador pode ser encontrada em [Atkeson et al. 1997].

O paradigma conexionista considera modelos matemáticos inspirados no funcionamento do sistema neuronal biológico a fim de resolver problemas práticos de engenharia e aumentar o conhecimento acerca dos mescanimos cerebrais de aprendizado. A pesquisa nessa área resultou no desenvolvimento

de estruturas altamente inter-conectadas, chamadas Redes Neurais Artificiais (RNAs) [McCulloch e Pitts 1988], capazes de representar diversos tipos de conceitos a partir de exemplos. Nelas, as unidades de processamento básicas são os neurônios, que são organizadas em camadas. De acordo com a forma de conexão entre os neurônios nas camadas, tem-se diversas modalidades de RNAs. Uma das mais utilizadas corresponde à rede *Multilayer Perceptron* (MLP). Tipicamente, ela possui três tipos de camadas de neurônios. As do primeiro tipo realizam a captação dos dados, dispondo de um neurônio para cada atributo dos exemplos. As do segundo tipo contêm neurônios ocultos que possuem funções de ativação não lineares que controlam a magnitude das saídas dos neurônios. As do terceiro tipo apresentam os neurônios que exibem a saída da rede. O aprendizado em RNAs ocorre por meio dos ajustes dos pesos de conexão entre os neurônios de camadas distintas. Assim, os algoritmos de treinamento buscam determinar combinações de pesos que levem a uma minimização do erro entre as saídas obtidas pelos neurônios da última camada e as saídas dos exemplos. Para realizar essa tarefa, o conhecido algoritmo de retropropagação de erro trabalha em duas fases. Na primeira, os exemplos são apresentados à rede e seus efeitos sobre os pesos das conexões são propagados para frente. Na segunda, as últimas saídas obtidas são comparadas às saídas ideais e os pesos de toda a rede são ajustados para trás, de acordo com uma regra de correção de erro. Uma exposição sobre RNAs e referências relevantes do assunto podem ser obtidas no Capítulo 4 de [Mitchell 1997].

1.2.3. Desempenho de Classificadores

Após a aplicação de um algoritmo de AM sobre um conjunto de dados S, é importante avaliar a qualidade da hipótese induzida, a fim de determinar quão boa é sua capacidade de generalização. Em situações ideais, seria possível calcular o erro verdadeiro da hipótese com respeito à distribuição arbitrária D, definido como a probabilidade de que ela irá classificar erroneamente um exemplo amostrado aleatoriamente de D. Na prática, a distribuição D é desconhecida. Assim, faz-se mister utilizar estimativas de erro baseadas exclusivamente na amostra de dados disponível. Para tanto, dois aspectos devem ser definidos a priori: uma medida de desempenho dos modelos induzidos e uma estratégia para a avaliação dos mesmos. Ambos são discutidos a seguir.

1.2.3.1. Medidas de desempenho

Diversas medidas podem ser utilizadas para mensurar o desempenho dos algoritmos de classificação. Em um trabalho abrangente [Ferri et al. 2009], foram investigadas experimentalmente 18 medidas comumente consideradas pela comunidade de AM, tais como acurácia/erro, medida-F, AUC, etc, a fim de compreender melhor suas propriedades e relações. Para tanto, a análise empreendida apresentou duas vertentes. A primeira, de caráter qualitativo, propôs uma taxonomia baseada na sensibilidade das medidas em relação a

quatro propriedades, a saber: a escolha do limiar de classificação do algoritmo de AM, a calibração da saída do algoritmo, a ordenação das saídas do algoritmo e a distribuição dos exemplos entre as classes. A segunda, de caráter quantitativo, considerou a correlação entre os resultados obtidos pelas diversas medidas quando da aplicação de 6 algoritmos de classificação sobre 30 bases de dados em diversos cenários. Com esse estudo, os autores foram capazes de identificar tendências de comportamento entre as abordagens e destacar suas similaridades e diferenças, fornecendo orientação quanto à escolha da medida mais adequada para uma aplicação específica.

Na prática, a acurácia e o erro permanecem como as medidas padrão de desempenho. A taxa de erro $Erro(h)$ da hipótese h, relativa a um conjunto de teste T, de cardinalidade p, é definida pela Equação 1, que compara as classes reais dos exemplos $\overrightarrow{x_i}$ com aquelas preditas por h. A função $\delta(y_i, h(\overrightarrow{x_i}))$ é 1 se $y_i \neq h(\overrightarrow{x_i})$ e 0, caso contrário. O complemento da taxa de erro do classificador corresponde à acurácia.

$$Erro(h) = \frac{1}{p} \sum_{i=1}^{p} \delta(y_i, h(\overrightarrow{x_i})) \tag{1}$$

Alternativamente, pode-se mensurar o desempenho de classificadores através da utilização de análise de curvas do tipo *Receiver Operating Characteristics* (ROC), como ilustrado em [Swets 1988]. Seu uso tem sido realizado em problemas com classes potencialmente desbalanceadas e com custo distintos nos erros de classificação. Para auxiliar na avaliação, organização e seleção de algoritmos de AM, elas são estruturadas sobre um plano cartesiano onde cada modelo classificador é representado por um ponto. No eixo das abscissas, tem-se a taxa de verdadeiros positivos do modelo, ou seja, a proporção de exemplos da classe positiva classificados como tal, considerando-se um conjunto de testes. No eixo das ordenadas, tem-se a taxa de falsos positivos, que representa a proporção de exemplos erroneamente classificados como da classe positiva. A curva construída pela união dos pontos representa um compromisso entre acertos e erros obtidos, de acordo com diversos limiares de classificação pré-estabelecidos. Por meio de um exame conjunto desses resultados, pode-se ter uma descrição do comportamento preditivo do algoritmo e, por conseguinte, confrontar diversas abordagens de classificação. Notas e considerações práticas acerca de curvas ROC estão disponíveis em [Fawcett 2006].

A comparação entre algoritmos pode não ser trivial utilizando análise ROC diretamente, devido a sua concepção gráfica bidimensional. Para auxiliar na investigação, pode-se resumir a informação obtida pela curva em uma grandeza escalar única dada pela *Area Under an ROC Curve* (AUC). Ela é calculada pela Equação 2, relativa a uma hipótese h induzida por um algoritmo de AM e aplicada sobre um conjunto de testes T:

$$\hat{A} = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 \cdot n_1} \tag{2}$$

onde n_0 e n_1 são os números de exemplos positivos e negativos, respectivamente, e $S_0 = \sum r_i$, com r_i igual à posição do i-ésimo exemplo positivo em uma lista ordenada. $\hat{A}$ varia de 0 a 1, sendo que valores maiores são preferíveis, pois indicam uma maior área sob a curva.

1.2.3.2. Estratégia de avaliação

Para que a estimativa do erro seja confiável, a hipótese h deve ser testada em exemplos diferentes daqueles que foram usados para a induzir. Para definir que exemplos deverão integrar o conjunto T utilizado para calcular os $Erro(h)$ ou $\hat{A}$ definidos anteriormente, vários métodos podem ser utilizados. As principais alternativas baseiam-se na noção de amostragem, segundo a qual os exemplos são escolhidos aleatoriamente do conjunto de dados S. Teoricamente, o conjunto S representa uma distribuição aproximada D' da distribuição verdadeira D. Assumindo que D' represente o mundo real, pode-se simular o processo de amostragem sobre D e obter o conjunto de testes T. Na prática, os dados podem ser particionados em vários conjuntos de treinamento e teste para permitir uma estimativa mais confiável do desempenho do classificador. Os métodos mais utilizados para a construção de T são brevemente descritos a seguir. Uma apresentação mais detalhada sobre metodologias de avaliação de algoritmos está disponível em [Baranauskas 2001].

1. **Resubstituição** Este é o método mais simples para a definição de T e não faz uso de amostragem. Ele utiliza o próprio conjunto S como conjunto T, ou seja, os exemplos presentes na indução do classificador são exatamente os mesmos empregados para testá-lo. Nesta situação, a estimativa de erro é chamada de erro aparente e pode não aproximar fielmente o erro de generalização de h para conjuntos independentes de teste, na medida em que o próprio algoritmo de AM já busca minimizar o erro durante o treinamento. Assim, é argumentado na literatura de AM que a resubstituição tende a apresentar uma estimativa muito otimista do erro.
2. ***Holdout*** Neste método, S é dividido aleatoriamente em um conjunto de treinamento e outro de teste, de tamanhos previamente fixados. Usualmente, $2/3$ dos exemplos são utilizados para a construção da hipótese e $1/3$ para testá-la, embora outras proporções possam ser utilizadas. A estimativa do erro é calculada sobre o conjunto de teste. Para minimizar a variação dos resultados devido à divisão aleatória dos dados, pode-se realizar repetidos particionamentos e calcular a média das estimativas em cada caso. O *Holdout* tende a apresentar uma estimativa pessimista do error, devido à utilização de um conjunto de treinamento reduzido para a indução da hipótese h.
3. ***Cross-Validation*** Este método consiste em dividir aleatoriamente o conjunto S em k partições mutuamente exclusivas de exemplos, de tamanho

aproximadamente igual. Das k partições, $k-1$ são utilizadas para o treinamento do algoritmo de AM e a restante é utilizada para teste, aplicando-se as Equações 1 ou 2. O processo é repetido k vezes, em cada uma intercalando a partição utilizada no teste. O erro obtido para cada uma das k partições é ponderado, obtendo-se assim uma estimativa de desempenho do classificador. Duas variantes do *Cross-Validation* (CV) são comumente utilizadas: *Stratified Cross-Validation* e o *Leave-one-out* (LOO). Na primeira, ao gerar-se as partições, tem-se o cuidado de amostrar os exemplos tal que a mesma proporção de dados em cada uma das classes de S seja mantida nas partições. A segunda é um caso especial do CV, em que k é igual à quantidade de exemplos de S.

4. ***Bootstrap*** Neste método, são gerados L conjuntos de treinamento com exemplos amostrados uniformemente a partir de S, com reposição, tal que cada conjunto tenha a mesma cardinalidade n do conjunto original. Por esse esquema, a probabilidade de um exemplo não ser selecionado é dada por $(1-1/n)^n \approx e^{-1} \approx 0.368$. Os exemplos não amostrados integram os conjuntos de teste. O restante compõe os conjuntos de treinamento, cuja cardinalidade esperada é $0.632n$. Os l conjuntos de treinamento e teste são então utilizados para induzir as hipóteses e calcular suas taxas de erro ou AUC, respectivamente. A média das l taxas obtidas é utilizada para estimar o desempenho do classificador.

5. **.632** e **.632+**. Como o *boostrap* utiliza em média apenas 63% dos dados em cada conjunto de treinamento, sua estimativa de desempenho é pessimista. Para tentar corrigir o problema, foi criado o estimador .632 [Efron e Tibshirani 1993], que faz uma combinação linear do erro de resubstituição e o erro de *boostrap*, com pesos respectivos de 0.368 e 0.632. Com isso, estimativas otimista e pessimista são ponderadas para a construção de um estimador mais adequado. Como um melhoramento do .632, foi criado o .632+ [Efron e Tibshirani 1997], que não fixa a priori o peso dos erros de resubstituição e *bootstrap*, permitindo que eles sejam definidos dinamicamente. Assim, o .632+ tende a ser mais adequado a casos de *overfitting* severo. Usualmente, esta estratégia de avaliação é empregada em conjunto com a medida $Erro(h)$.

A comunidade científica ainda não estabeleceu um consenso sobre qual o melhor método de estimação de erro. Independente da abordagem adotada, é importante perceber que a estimativa obtida pode não ser acurada em relação ao verdadeiro erro esperado em quando há uma pequena quantidade de exemplos disponíveis. Situações onde isso ocorre são diversas em aplicações reais. Em [Isaksson et al. 2008], os autores realizaram um estudo simulando esse cenário e concluíram que, para um conjunto de dados único, não é possível determinar quão confiável é a estimativa do erro obtida através de *Cross-Validation* e de *bootstrap*. Os experimentos realizados ainda mostram que quanto menor o número de exemplos da base de dados, mais variável é a

estimativa do erro.

1.2.4. Comparação de Classificadores

Segundo [Wolpert 1996], não existe um único algoritmo que, em situações gerais, supere todos os demais sempre. Assim, estudos comparativos podem ser importantes para entender as qualidades e deficiências das abordagens e, desse modo, determinar qual a mais apropriada em uma dada situação. Na Seção 1.2.3, algumas formas de estimar o erro de uma hipótese induzida foram apresentadas. Comparar diretamente as estimativas de erros dos classificadores pode não ser suficiente para precisar qual o melhor, pois é necessário considerar os vários fatores de variabilidade dos dados e dos algoritmos, tais como a escolha dos conjuntos de treinamento e teste e a aleatoriedade interna de algumas abordagens de AM.

Uma forma de minimizar o efeito desses fatores é utilizar métodos estatísticos para aferir se a diferença de desempenho entre os classificadores é ou não significativa. Em [Dietterich 1998], o autor analisou vários testes capazes de comparar 2 algoritmos de AM em uma determinada base de dados. Embora todas as abordagens apresentem algum tipo de problema em relação aos fatores de variabilidade, o autor foi capaz de recomendar duas delas: o teste Nc-Nemar e o teste t pareado de 5x2CV. Quando vários algoritmos são comparados, é possível aplicar esses testes estátisticos com pares de algoritmos e em seguida corrigir o nível de significância global para considerar múltiplas comparações. Nessa situação, outras alternativa são os testes ANOVA e o teste binomial com correção de Bonferroni, conforme comentado em [Salzberg 1997]. No caso mais geral de vários algoritmos sendo comparados em várias bases ao mesmo tempo, o teste de Friedman seguido do teste de Nemenyi tem sido recomendado [Demšar 2006].

1.3. Meta-aprendizado

O meta-aprendizado estuda como os algoritmos de AM podem aumentar sua eficiência por meio da experiência [Vilalta e Drissi 2002]. De maneira geral, o objetivo é adquirir conhecimento acerca de como o próprio processo de aprendizado pode se tornar flexível de acordo com a natureza da tarefa considerada. Todos os algoritmos de AM funcionam adaptando seus parâmetros a um ambiente específico. O meta-aprendizado difere do aprendizado convencional, de base, no escopo de seu nível de adaptação. Enquanto o AM tradicional trabalha sobre um conjunto de dados por vez, o aprendizado no meta-nível é baseada no acúmulo de experiência do desempenho de múltiplas aplicações de um algoritmo de AM. De maneira geral, pode-se dizer que o meta-aprendizado tem interesse em focar na relação entre estratégias de aprendizado e problemas [Vilalta et al. 2005].

Dentre as aplicações mais comuns de meta-aprendizado, tem-se o problema de gerar regras capazes de relacionar o desempenho de algoritmos de AM com as propriedades dos conjuntos de dados. Em termos práticos, isso poderia ajudar na criação de sistemas que fornecessem ao usuário sugestões

sobre que algoritmos utilizar em determinadas situações. Tais sistemas, conforme visto em [Kalousis 2002], podem ser estudados segundo quatro critérios, que serão tratados no decorrer desta seção:

- Caracterização de bases de dados;
- Medidas de avaliação;
- Formas de sugestão;
- Métodos de construção de sugestão.

A seção é organizada como segue. Na Seção 1.3.1, o problema de recomendação de algoritmos é formalizado e sua relação com meta-aprendizado é estabelecida. Nela também se apresenta o vocabulário que será utilizado no decorrer do trabalho. Na Seção 1.3.2, as abordagens para a caracterização das bases de dados são discutidas. Na Seção 1.3.3, as medidas de desempenho dos classificadores, nas quais a sugestão pode ser baseada, são comentadas. Na Seção 1.3.4, as formas como a sugestão deve se apresentar ao usuário é abordada. Na Seção 1.3.5, é tratado o tema de métodos de construção, que aborda como realizar o mapeamento entre as propriedades das base de dados e o desempenho dos classificadores.

1.3.1. Recomendação de algoritmos

Com o avanço das pesquisas em AM, vários métodos e algoritmos têm se tornado disponíveis. Para que ferramentas baseadas nesses abordagens resultem em soluções apropriadas às necessidades do usuário, elas devem ser cuidadosamente escolhidas de acordo com a natureza da tarefa considerada. Isso ocorre porque cada algoritmo possui a chamada superioridade seletiva [Brodley 1995], segundo a qual ele pode desempenhar melhor que seus pares em uma determinada classe de tarefas. Assim, existe uma necessidade de se relacionar o viés indutivo de cada algoritmo de AM à morfologia dos dados em questão. Esse cenário foi modelado formalmente por [Rice 1976], que apresentou a seguinte definição para o problema de seleção de algoritmos:

> Para uma determinada instância de problema $x \in P$, com características $f(x) \in F$, encontre o mapeamento $S(f(x))$ no espaço de algoritmos A, tal que o algoritmo selecionado $\alpha \in A$ maximize o mapeamento de desempenho $y(\alpha(x)) \in Y$.

Considerando-se o espaço P de problemas, A de algoritmos e Y de desempenhos, a solução trivial para lidar com a situação emprega a interação de aconselhamento especializado com experimentação computacional custosa. Entre outras deficiências, essa abordagem não é capaz de determinar automaticamente o S, impedindo um aproveitamento sistemático de conhecimento acumulado no passado para a resolução de problemas semelhantes no futuro. Pela utilização de meta-aprendizado, é possível aprender tal mapeamento, propiciando a construção de sistemas de recomendação eficientes e efetivos.

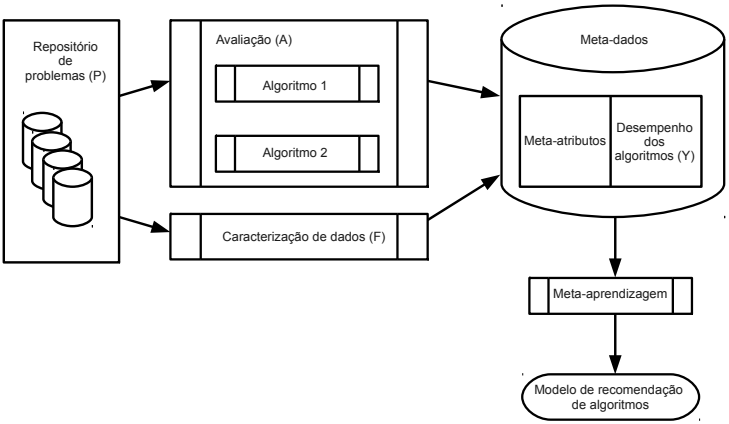

Figura 1.1. Processo de recomendação de algoritmos utilizando meta-aprendizado. Adaptado de [Brazdil et al. 2009]

O arcabouço genérico do processo de recomendação por meta-aprendizado é ilustrado na Figura 1.1. O processo inicia-se com a aquisição de um conjunto apropriado de problemas que sejam representativos daqueles para os quais a recomendação posterior será realizada. Ele representa o espaço P na definição de Rice. Em seguida, duas etapas são aplicadas a cada elemento de P: a avaliação dos algoritmos em A e a extração de características segundo as medidas em F. Idealmente, a caracterização dos problemas deve ser preditiva quanto ao comportamento dos algoritmos. Associando-se essas duas informações para cada problema, obtém-se um meta-exemplo, formado por meta-atributos de entrada e meta-atributos alvo (Y), respectivamente. Ao conjunto dos meta-exemplos disponíveis, dá-se o nome de meta-dados. Para induzir então o mapeamento S entre meta-atributos de entrada e meta-atributos alvo, aplica-se um algoritmo de AM, referido como meta-aprendiz. Por meio dele, pode-se utilizar o meta-conhecimento obtido do processo de aprendizado e realizar, por fim, a recomendação de algoritmos no contexto de meta-aprendizado.

As particularidades necessárias para a utilização adequada da abordagem ilustrada na Figura 1.1 serão apresentadas nas próximas subseções.

1.3.2. Caracterização de bases de dados

Caracterizar bases de dados consiste em identificar e extrair propriedades que, possivelmente, afetem o desempenho dos algoritmos de classificação. O objetivo da caracterização é fornecer informações morfológicas dos dados para a aplicação de técnicas de meta-aprendizado. Isto é possível devido ao

conhecimento *a priori* do comportamento dos algoritmos, sob um determinado aspecto. Por exemplo, alguns algoritmos não operam satisfatoriamente na presença de atributos irrelevantes, como o kNN, enquanto outros, como RNAs e SVMs, possuem mecanismos internos de seleção/ponderação de atributos, fazendo com que sejam mais robustos. Outro exemplo é o algoritmo *Naïve Bayes*, que pode não apresentar resultados satisfatórios em alguns problemas com atributos altamente redundantes. Com observações como essas, compreende-se a importância de considerar medidas que explorem as particularidades dos dados a fim de entender o desempenho dos algoritmos de classificação.

De acordo com [Soares et al. 2004], as medidas que caracterizam as bases de dados devem conter informação relevante para determinar o desempenho relativo entre os algoritmos de classificação e apresentar baixo custo computacional. Atualmente, a pesquisa em caracterização concentra-se em três áreas [Vilalta et al. 2005]:

- Caracterização direta;
- Caracterização baseada em *landmarking*;
- Caracterização via modelos.

A seguir, uma breve explanação acerca dessas abordagens é fornecida.

1.3.2.1. Caracterização direta

Um dos primeiros esforços sistemáticos e em larga escala para tentar relacionar as medidas que caracterizam as bases de dados e o desempenho dos algoritmos foi empreendido no projeto Statlog [Michie et al. 1994]. Entre outros objetivos, o projeto pesquisou porque certos algoritmos classificavam bem em alguns domínios e apenas regular em outros. Seus experimentos foram realizados utilizando 23 algoritmos e 21 bases de dados. As medidas, ou meta-atributos de entrada, utilizadas para caracterizar as bases de dados, foram divididas em três categorias: simples, estatísticas e baseadas na teoria da informação. Os meta-atributos simples incluem medidas gerais das bases de dados, como o número de atributos e o número de exemplos, entre outras. Os estatísticos aplicam conceitos como os coeficientes médios de assimetria e curtose, etc aos atributos numéricos. As medidas baseadas na teoria da informação são utilizadas para caracterizar os atributos nominais e sua relação com o atributo classe. A Tabela 1.1 informa todas as medidas consideradas no Statlog. Outras medidas relacionadas ao projeto foram posteriormente propostas por [Lindner e Studer 1999] e por [Sohn 1999].

Mais recentemente, o projeto METAL (`www.metal-kdd.org`) visou o desenvolvimento de ferramentas que auxiliem o usuário a selecionar uma combinação adequada de técnicas de pré-processamento, classificação e regressão. O projeto fomentou o desenvolvimento de várias abordagens relacionadas à

Tabela 1.1. Medidas utilizadas no Statlog, separadas por categoria.

Tipo	Descrição
Simples	Número de exemplos
	Número de atributos
	Número de classes
	Número de atributos binários
Estatísticas	Razão média entre desvio padrão dos atributos
	Correlação média absoluta entre atributos, por classe
	Primeira correlação canônica
	Proporção de variância explicada pelo 1o discriminante canônico
	Assimetria média absoluta dos atributos
	Curtose média dos atributos
Informação	Entropia normalizada das classes
	Entropia média dos atributos
	Informação mútua média entre classe e atributos
	Razão sinal/ruído

pesquisa em meta-aprendizado, especialmente em relação a formas apropriadas de meta-aprendizado, definição de meta-atributos para caracterizar os conjuntos de dados e o desempenho dos algoritmos, e estudos empíricos e teóricos sobre o tema. No contexto de caracteriação, uma das abordagens desenvolvidas consistiu em estender as medidas utilizadas no Statlog. As medidas extras utilizadas podem ser vistas na Tabela 1.2.

Tabela 1.2. Medidas extras utilizadas no METAL, separadas por categoria.

Tipo	Descrição
Simples	Número de atributos nominais
	Número de atributos numéricos
Estatísticas	Número de atributos com *outliers*
	Estatística M de Box
	Graus de liberdade da Estatística M
	Valor de Lambda de Wilk
	Estatística V de Barlett
Informação	Entropia conjunta de classe e atributos

Outras contribuições promissoras para a caracterização direta podem ser encontradas em [Kalousis 2002, Soares 2004].

1.3.2.2. *Landmarking*

Em [Pfahringer et al. 2000], os autores introduziram uma técnica de caracterização de bases de dados baseada no desempenho de algoritmos de classificação. Segundo os autores, cada algoritmo opera satisfatoriamente em uma determinada área de competência, ou seja, junto a um certo tipo de dados. Assim, por meio da aplicação de algoritmos simples, chamados *landmarkers*,

seria possível obter informação importante sobre a natureza do domínio em que eles são aplicados. Portanto, uma base de dados poderia ser descrita pela coleção de áreas de competência às quais ela pertence. *Landmarking* consiste da aplicação de *landmarkers* a bases de dados a fim de localizá-las em um espaço onde áreas de competências estão inseridas, o espaço de competências.

O *landmarking* é utilizado para determinar a proximidade de uma base de dados em relação a outras, através da similaridade de desempenho dos *landmarkers*. Com isso, forma-se uma vizinhança de áreas de competência, onde bases de dados podem ser representadas. Espera-se que bases de dados de natureza semelhante pertençam às mesmas áreas de competência e, por conseguinte, sejam adequadas à aplicação dos mesmos algoritmos de classificação. Aos algoritmos de meta-aprendizado cabe explorar quão bem as informações dos *landmarkers* podem ser utilizadas para localizar as bases de dados no espaço de competências.

Um exemplo de utilização de *landmarking* como técnica de caracterização de dados em meta-aprendizado pode ser visto no trabalho de [Bensusan e Giraud-Carrier 2000]. No estudo, os autores selecionaram sete algoritmos simples com diferentes mecanismos de funcionamento para atuar como *landmarkers* e dez algoritmos complexos para servir de sugestão à classificação final dos dados. Os *landmarkers* foram utilizados como meta-atributos por um esquema de meta-aprendizado para aprender e sugerir o algoritmo complexo mais apropriado para aplicação em uma base de dados. Os experimentos do trabalho compararam o *landmarking* e diferentes medidas envolvendo a caracterização direta, apresentada na subseção anterior, e indicaram o potencial da nova abordagem.

1.3.2.3. Caracterização via modelos

Uma forma alternativa de representar bases de dados utilizando algoritmos de classificação é fornecida pela caracterização via modelos [Bensusan et al. 2000]. Diferente da abordagem anterior, *landmarking*, a caracterização via modelos não considera diretamente medidas de desempenho do classificador induzido e sim a estrutura do próprio classificador, conhecida como a hipótese induzida ou modelo. Segundo [Vilalta et al. 2005], há diversas vantagens neste tipo de caracterização, dentre as quais destacam-se: a base de dados é resumida em uma estrutura que contém informações sobre a complexidade e desempenho do modelo e; a representação dos dados nessa forma pode servir de base para explicar o desempenho do algoritmo de aprendizado.

A utilização de modelos para a caracterização de bases de dados realiza uma mudança no espaço de busca do algoritmo de meta-aprendizado, passando do espaço de exemplos para o espaço de hipóteses do algoritmo utilizado para a caracterização. Como esse algoritmo, idealmente, é capaz de realizar uma busca eficiente em seu rico espaço de hipóteses, espera-se que a utilização da hipótese induzida comprima a base de dados original de forma

a oferecer meta-atributos mais informativos ao algoritmo de meta-aprendizado.

Dentre os algoritmos de classificação, as árvores de decisão são as mais utilizadas para realizar a caracterização de bases de dados. De acordo com [Bensusan et al. 2000], há evidências empíricas que sugerem importantes conexões entre as propriedades das bases de dados e as estruturas de árvores de decisão não podadas. Além disso, elas têm sido muito estudadas e apresentam comportamento determinístico, facilitando sua aplicação e entendimento. Como meta-atributos, diversas medidas de uma árvore de decisão podem ser utilizadas, como, por exemplo: o número de nós por atributo, a profundidade máxima da árvore, a sua forma, o grau de balanceamento, etc. Extensões das medidas de árvores são fornecidas por [Peng et al. 2002].

1.3.3. Medidas de avaliação

Para determinar qual algoritmo de AM utilizar em um determinado conjunto de dados, é necessário especificar as medidas de desempenho consideradas, para que uma lista de preferência dos algoritmos possa ser estabelecida. Diversas medidas podem ser utilizadas para este fim. Em estudo recente [Ferri et al. 2009] (vide a Seção 1.2 para mais detalhes), os autores analisaram empiricamente o comportamento de diversas medidas comumente empregadas em AM. Consideraram-se, por exemplo, acurácia/erro, medida-F, AUC, etc. Sua conclusão é que as medidas de fato tendem a mensurar aspectos diferentes do comportamento preditivo dos algoritmos. Isto enfatiza a necessidade de uma cuidadosa escolha de que medida utilizar.

Além de medidas relacionadas com as predições feitas pelos classificadores, em alguns domínios outras medidas podem ser interessantes, como, por exemplo, o tempo requerido para o algoritmo de AM construir um classificador, o tempo requerido para o classificador rotular um exemplo, a quantidade de memória requerida pelo algoritmo e a simplicidade e interpretabilidade dos classificadores construídos, entre outras. Assim, embora a utilização de apenas uma medida de avaliação seja a prática mais comum em meta-aprendizado, é importante que os sistemas de recomendação baseados em meta-aprendizado consigam lidar com abordagens multi-objetivas, em que duas ou mais medidas são combinadas. Por exemplo, o usuário pode estar interessado no compromisso entre tempo de treinamento e acurácia, preferindo algoritmos mais rápidos mesmo que discretamente menos acurados. Nesse caso, pode-se mapear os valores das medidas em um único valor, a fim de definir a lista de preferência entre os algoritmos. Em [Soares e Brazdil 2000], por exemplo, os autores utilizaram como critério uma medida chamada *Adjusted Ratio of Ratios*, que combina acurácia e tempo de execução. Uma abordagem mais flexível foi proposta por [Nakhaeizadeh e Schnabl 1997]. Ela permite a incorporação de qualquer quantidade de medidas de avaliação. Refefências para métodos alternativos multi-critérios podem ser encontrados no Capítulo 3 de [Brazdil et al. 2009].

1.3.4. Formas de sugestão

De acordo com [Kalousis 2002], de maneira geral, há três abordagens para sugerir algoritmos para a apreciação do usuário. A primeira abordagem consiste em fornecer apenas o melhor algoritmo, ou seja, aquele que produza, supostamente, o modelo mais apropriado para uma dada tarefa ou conjunto de dados de dados, segundo algum critério. Exemplos da utilização desse tipo de sugestão podem ser observados em [Bensusan e Giraud-Carrier 2000, Bensusan et al. 2000]. Uma crítica a essa abordagem refere-se à eventual não disponibilidade do algoritmo em um dado momento, impossibilitando sua aplicação. Outro aspecto negativo é que não há garantias de que o algoritmo sugerido produzirá, de fato, o resultado mais adequado. Assim, a recomendação de algoritmos alternativos seria desejável.

A segunda abordagem é mais flexível, indicando, dentre os algoritmos considerados, aqueles mais promissores para um dado problema. A definição de que algoritmos compõem esse grupo depende, geralmente, do desempenho relativo dos mesmos. Uma alternativa é fornecer como sugestão ao usuário o algoritmo com melhor desempenho esperado e também aqueles cujos desempenhos não sejam inferiores ao melhor por uma dada margem. Outra forma de decidir os componentes do grupo recomendado é através da utilização de testes estatísticos para comparar a significância das diferenças entre os desempenhos dos algoritmos. Assim, um algoritmo será sugerido caso ele não seja significantemente inferior ao melhor algoritmo de AM. No projeto Statlog, as sugestões eram apresentadas de acordo com essa abordagem.

A terceira abordagem exibe os algoritmos em ordem de preferência com relação ao conjunto de dados. O critério de ordenação pode ser simplesmente a acurácia dos classificadores ou medidas mais complexas, que envolvem múltiplos objetivos, tais como tempo de execução do algoritmo ou a interpretabilidade do modelo gerado, conforme visto na subseção anterior. A apresentação da sugestão na forma de *rankings* é interessante pois ela permite que o sistema de meta-aprendizado seja desenvolvido sem o conhecimento prévio de quantos algoritmos serão experimentados pelo usuário. Assim, havendo disponibilidade de tempo e recursos, esse número pode ser ampliado seguindo a ordem fornecida pelo *ranking*. Uma desvantagem dessa abordagem é a ausência de informações absolutas quanto ao desempenho dos algoritmos, pois apenas a ordem dos mesmos é apresentada. Não obstante, ela tem sido aplicada em diversos trabalhos (vide referências em no Capítulo 3 de [Brazdil et al. 2009]).

1.3.5. Construção de sugestão

Uma das principais tarefas do meta-aprendizado é relacionar as características dos conjuntos de dados ao desempenho dos algoritmos de AM, de forma a poder sugerir que algoritmos utilizar em determinados casos. De maneira geral, essa tarefa pode ser vista como um problema de AM, situado em um nível superior de abstração. Nele, os meta-exemplos compostos por meta-atributos de entrada e meta-atributo alvo são utilizados pelo meta-aprendiz

para induzir um meta-modelo capaz de aprender a associação desejada. Para ilustrar esse processo, três formas da utilização de meta-aprendizado são discutidas a seguir.

No projeto Statlog, o objetivo era determinar quais algoritmos de AM supervisionado, de um total de n, são aplicáveis a determinados conjuntos de dados. Para isso, n meta-problemas são construídos. Cada um deles é formado por meta-exemplos associados a uma das duas classes: aplicável e não-aplicável, dependendo se um dado algoritmo de AM apresentam desempenho alto ou baixo, em termos de acurácia, naquela base de dados. Os meta-classificadores aplicados aos meta-problemas são sistemas de regra de decisão. Para determinar quais algoritmos de AM são aplicáveis a uma nova base de dados, os n meta-classificadores são executados e a resposta é obtida.

Outra formulação do problema de meta-aprendizado considera $\binom{n}{2}$ meta-problemas, onde cada um corresponde a uma comparação pareada dos algoritmos de AM disponíveis. Os meta-exemplos dos meta-problemas são as descrições das bases de dados, mas agora a classe associada indica qual dos dois algoritmos é mais apropriado àquela base de dados específica. O objetivo aqui é construir meta-classificadores que descrevem sob que circunstâncias qual algoritmo de AM é preferível em relação a outro. Essa abordagem foi utilizada em [Pfahringer et al. 2000].

A forma de meta-aprendizado mais simples consiste em considerar um único meta-problema. Nele, a classe de cada meta-exemplos é o algoritmo de AM que apresenta maior acurácia naquela base de dados. Esta abordagem foi utilizada em [Bensusan e Giraud-Carrier 2000]. Nesse trabalho, eles consideraram diversos algoritmos de AM para construir meta-classificadores e verificaram o desempenho de cada um.

Técnicas mais sofisticadas e interessantes produzem diretamente *rankings* de algoritmos. Por exemplo, em [Soares e Brazdil 2000], os autores desenvolveram o *Zoomed-Ranking*, que opera em 2 fases. Na primeira, é selecionado um sub-conjunto de conjuntos de dados similares à nova base de dados. Esta seleção é feita com o uso do algoritmo kNN, cuja função de distância utiliza os meta-atributos de entrada dos conjuntos de dados para selecionar os k conjuntos de dados mais similares. Na segunda fase, o *ranking* é gerada, considerando o *Adjusted Ratio of Ratios*, comentado anteriormente.

1.4. Estudos de caso em meta-aprendizado

Conforme discutido na Seção 1.3, abordagens de meta-aprendizado podem ser empregadas para a elaboração de sistemas de recomendação de algoritmos. De maneira resumida, eles operam nas seguintes etapas básicas: geração de meta-exemplos a partir dos problemas disponíveis, indução de um meta-modelo capaz de aprender a relação entre os meta-atributos de entrada e o meta-atributo alvo, e a aplicação do meta-modelo para a construção de recomendações que dêem suporte à seleção de algoritmos para novos problemas. Utilizando esse arcabouço de meta-aprendizado, a recomen-

dação pode ocorrer em diversas áreas interdisciplinares. Em uma revisão abrangente [Smith-Miles 2009b], a autora destacou o uso de meta-aprendizado em domínios de classificação, regressão, previsão de séries, ordenação, satisfação de restrições e otimização. Apesar de as pesquisas em classificação serem preponderantes, estudos importantes nos outros domínios têm sido realizados. Uma listagem de alguns trabalhos representativos de cada área é fornecida naquele artigo.

A fim de exemplificar como a recomendação é realizada no contexto de meta-aprendizado, apresentam-se a seguir três estudos de caso. O primeiro é de classificação de dados de expressão gênica. O segundo trata da seleção de modelos para a previsão de séries temporais. O terceiro refere-se a um problema de otimização combinatorial, o caixeiro viajante. Esses casos representam classes de aplicações que englobam grande parte das pesquisas em meta-aprendizado para seleção de algoritmos, sendo, portanto, úteis para ilustrar o potencial dos métodos discutidos no presente trabalho. Além disso, o fato de os autores e seus grupos de pesquisa estarem investigando instâncias desses três grandes casos permite que conhecimento de cunho prático seja repassado.

1.4.1. Classificação de dados de expressão gênica

1.4.1.1. Contexto

Em abordagens tradicionais, a classificação de tumores é baseada primariamente em sua aparência morfológica e no tecido onde a doença se originou. Entretanto, não há garantias de que tumores semelhantes tenham o mesmo desenvolvimento clínico e, por conseguinte, exijam o mesmo tipo de tratamento. Assim, o estudo de técnicas que permitam determinar a variedade correta de câncer de um tecido, com uma taxa mais elevada de acerto, é essencial.

Neste cenário, para prover um maior entendimento do fenômeno estudado e tentar promover uma distinção mais acurada de tumores, em [Golub et al. 1999], os autores utilizaram um esquema simples de voto ponderado aplicado sobre dados de *microarrays*. Em seus experimentos, o metabolismo de células pertencentes a dois tipos de leucemia foi analisado e um padrão pôde ser aprendido. Com isso, mostrou-se ser possível a classificação de novas amostras de tecidos considerando-se apenas seus níveis de expressão gênica.

Após o esforço pioneiro de Golub e colaboradores, observou-se um notável crescimento no número de pesquisas envolvendo análise de dados de *microarrays* [Schena et al. 1995] por métodos de AM supervisionado. Tal interesse suscitou a necessidade de se investigar quais dessas técnicas de classificação seriam mais apropriadas. Em [Asyali et al. 2006], os autores destacaram algumas abordagens comumente consideradas nesta tarefa, com diferentes graus de adequação, e relataram que diversas propriedades de cada uma delas podem impactar seu desempenho. Isto pôde ser constatado através de trabalhos comparativos realizados com classificadores em dados de expressão gênica.

Alguns desses trabalhos são destacados em [Boulesteix et al. 2008].

Uma análise em conjunto dos resultados obtidos nesses estudos revela que não há um único método que ofereça um desempenho superior aos outros para todos os problemas considerados. Portanto, a definição de que algoritmo utilizar para a obtenção dos resultados mais satisfatórios em cada aplicação deve ser feita de maneira pontual. Para lidar com a situação de maneira eficiente e eficaz, em [de Souza 2010], o autor analisou como meta-aprendizado pode ser utilizado para a seleção de classificadores no domínio de expressão gênica. As pesquisas ali desenvolvidas resultaram em métodos capazes de realizar recomendações sobre *rankings* de algoritmos mais apropriados a bases de dados com determinadas características, oferencendo potencial suporte a cientistas da área médica na tarefa de escolher soluções adequadas para seus aplicações. A seguir, apresenta-se parte dos experimentos conduzidos em [de Souza 2010].

1.4.1.2. Materiais e métodos

Nesta seção, descrevem-se os conjuntos de dados, os algoritmos de AM, as medidas de caracterização e de desempenho utilizadas para a construção dos meta-exemplos, e os métodos aplicados para realizar o meta-aprendizado.

Conjuntos de dados

Os dados numéricos de expressão gênica gerados pelos experimentos de *microarrays* são dispostos, geralmente, em matrizes, tal que suas colunas representem as amostras de dados (por exemplo, tecidos de diferentes pacientes) e as linhas representem os genes utilizados para caracterizar as amostras. Cada entrada da matriz representa o nível de expressão de um gene específico em uma dada amostra. A Tabela 1.3 exibe um exemplo de matriz $n \times m$, onde $x_{i,j}$ é o valor de expressão gênica.

Tabela 1.3. Matriz genérica de expressão gênica

Genes	Tecido 1	Tecido 2	...	Tecido m
1	$x_{i,j}$	$x_{i,j}$		$x_{i,j}$
2	$x_{i,j}$	$x_{i,j}$		$x_{i,j}$
3	$x_{i,j}$	$x_{i,j}$		$x_{i,j}$
$\vdots$				
n	$x_{i,j}$	$x_{i,j}$		$x_{i,j}$

Seguindo o formato geral apresentado na Tabela 1.3, vários estudos têm disponibilizado seus dados na Internet para apreciação pública. Alguns deles são bem conhecidos e têm ajudado no entendimento do fenômeno de interesse, através de diversos tipos de análises feitas pela comunidade científica. Aqui, foca-se em problemas relacionados à classificação de tecidos com câncer. Assim, para os experimentos reportados em seções posteriores, foram

utilizadas 49 bases de dados de *microarrays*. As descrições das principais características deste conjunto e das operações de pré-processamento realizadas são fornecidas em [de Souza 2010].

Algoritmos base
Com base em estudos envolvendo classificação em dados de expressão gênica (vide referências em [Boulesteix et al. 2008]), selecionaram-se os seguintes algoritmos de AM para induzir os classificadores base do meta-aprendizado aqui utilizados: *Diagonal Linear Discriminat Analysis* (DLDA), *Diagonal Quadratic Discriminant Analsysis* (DQDA), *Prediction Analysis for Microarrays* (PAM), *k Nearest Neighbors* (kNN), *Support Vector Machines* com *kernel* Linear (SVM-L), *Support Vector Machines* com *kernel* Radial (SVM-R) e *Random Forests* (RF). Eles foram executados com os parâmetros padrão disponíveis nos pacotes R que os implementam ou na literatura usual de AM. A Tabela 1.4 exibe os pacotes R utilizados, assim como os valores utilizados nos parâmetros dos algoritmos, quando aplicável.

Tabela 1.4. Pacotes R utilizados e parâmetros dos algoritmos de AM

Algoritmo	Parâmetro(s)	Pacote
DLDA	Não aplicável	`sfsmisc`
DQDA	Não aplicável	`sfsmisc`
PAM	$\Delta = 1$	`pamr`
kNN	$k = 3$	`class`
SVM-L	$C = 1$	`e1071`
SVM-R	$\sigma = 1/\#Atributos, C = 1$	`e1071`
RF	$ntree = 500, mtryFactor = 1$	`randomForest`

Construção de meta-dados
Na Seção 1.3, destacaram-se algumas abordagens para caracterizar as bases de dados utilizadas em meta-aprendizado. Elas são empregadas para a geração dos meta-atributos de entrada constituintes dos meta-exemplos. Neste estudo de caso, focou-se em um conjunto de medidas designado STATLOG. Seus componentes são apresentados na Tabela 1.5. Eles correspondem às medidas desenvolvidas durante o projeto Statlog adequadas à natureza numérica e contínua dos dados de expressão gênica.

Desde a execução dos experimentos iniciais deste estudo de caso, observou-se que as medidas MDP, COR, CAN e PRO apresentadas na Tabela não eram adequadas a dados de alta dimensionalidade, pois ou exigiam o cálculo de determinantes de matrizes de covariância muito grandes ou de correlações par a par entre todos os atributos [Kalousis 2002], fazendo com que valores nulos fossem retornados em seu cômputo. Assim, o conjunto STATLOG pode se beneficiar da aplicação de abordagens de redução de dimensionalidade de dados. Devido a sua simplicidade, a seu baixo custo computacional e a seu bom desempenho em aplicações envolvendo expressão gênica (como, por

Tabela 1.5. Medidas de caracterização

Medida	Descrição
NEX	Logaritmo do número de exemplos
NAT	Logaritmo do número de atributos
NCL	Logaritmo do número de classes
MDP	Razão média entre desvio padrão dos atributos
COR	Correlação média absoluta entre atributos, por classe
CAN	Primeira correlação canônica
PRO	Proporção de variância explicada pelo 1ž discriminante canônico
ASS	Assimetria média absoluta dos atributos
CUR	Curtose média dos atributos
ENT	Entropia normalizada das classes

exemplo, no trabalho de [Boulesteix e Strimmer 2007]), o *Partial Least Squares* (PLS) [Wold et al. 2001] foi empregado aqui. PLS é um método de extração de atributos capaz de gerar combinações lineares ortogonais dos mesmos, tal que a covariância entre essas combinações e a variável de saída (classe) seja maximizada. O único parâmetro livre do método, o número de componentes do espaço reduzido de dimensões, foi selecionado entre $p = 2, ..., 15$ de forma a mininizar a estatística *Predicted Residual Error Sum of Squares* (PRESS), como realizado em [Souza et al. 2008].

Os meta-exemplos são também constituídos de meta-atributos alvo. No contexto deste estudo, eles correspondem ao *ranking* dos algoritmos base quando aplicados a uma determinada base de dados, tal que classificadores com melhores desempenhos são dispostos em posições menores nos *rankings*. A determinação do melhor método para estimar o erro dos métodos quando a classificação ocorre em dados de *microarrays* é assunto controverso e depende dos objetivos de cada estudo [Boulesteix et al. 2008]. Nos experimentos conduzidos aqui, tal tarefa é realizada utilizando-se o estimador .632+, com 50 amostras de *bootstrap*. Assim, a geração dos *rankings* ideais dos 49 meta-exemplos utilizados é baseada no .632+. Sucintamente, ele opera através da ponderação de dois termos: o erro aparente e o erro de *bootstrap*, tal que a estimativa otimista do primeiro é compensada pela estimativa pessimista do segundo. Desta forma, tem-se uma maior robustez da estimativa de erro para diferentes abordagens de classificação. Um detalhamento do método, incluindo as equações utilizadas em sua implementação, está disponível em [Efron e Tibshirani 1997].

Método de meta-aprendizado

Em geral, um *ranking* representa uma função de preferência sobre um conjunto de itens. Neste estudo de caso, os itens são os algoritmos de AM considerados para a classificação dos dados de expressão gênica, enquanto a função de preferência expressa o desempenho esperado de um algoritmo em uma base de dados. Assim, se um algoritmo desempenha melhor que outro em uma base particular, ele deve ser posicionado melhor no *ranking*. No contexto de meta-

aprendizado estudado aqui, o objetivo é relacionar as características das bases de dados à ordem relativa de desempenho dos algoritmos de classificação. Isso pode ser feito, entre outros, por meio de um método de *ranking* baseado em vizinhos mais próximos, como descrito abaixo.

O kNN foi adaptado para a tarefa de aprender *rankings* e foi aplicado a problemas de meta-aprendizagem com relativo sucesso (vide Capítulo 3 de [Brazdil et al. 2009]). Em classificação com atributo alvo único, ele opera selecionando, no conjunto do treinamento, os k exemplos mais similares a um exemplo de teste qualquer e combinando suas classes para prover a nova predição. Quando o atributo alvo apresenta-se na forma de *ranking*, os k *rankings* dos exemplos selecionados necessitam ser agregados para formar o *ranking* do exemplo de teste. Uma abordagem simples para isto é considerar o método *Average Ranks* (AR). Seja $R_{i,j}$ a posição no *ranking* do algoritmo a_j ($j = 1, \ldots, n$) na base de dados i, onde n é o número de algoritmos. A posição média no *ranking* para cada a_j é dada pela Equação 3. O *ranking* predito é obtido ordenando-se os valores de $\bar{R}_j$ e ajustando suas posições de acordo.

$$\bar{R}_j = \frac{\sum_{i=1}^{k} R_{i,j}}{k} \tag{3}$$

Uma extensão do método de *ranking* por kNN foi desenvolvida em [de Souza 2010]. Ela baseia-se na intuição de que, entre os k exemplos do conjunto de treinamento vizinhos do exemplo de teste, os mais próximos devam ter maior influência na formação do *ranking*. Esta extensão é uma adaptação do *weighted k-Nearest Neighbor* (wKNN) [Hechenbichler e Schliep 2006], que foi proposto para classificação com atributo alvo único, para lidar com atributo alvo do tipo *ranking*. O wkNN pondera a influência de cada vizinho na classificação do exemplo de teste transformando as distâncias até ele em medidas de similaridade, que por sua vez são utilizadas como pesos. Esta transformação é realizada pela utilização de funções de *kernel* $K(\cdot)$, que operam sobre uma distância d entre o exemplo de teste e um vizinho. Ela atinge valor máximo quando $d = 0$ e decresce à medida que d cresce. A seguir, tem-se algumas das funções $K(\cdot)$ mais comuns:

$$\begin{aligned}
\text{Retangular} &: \frac{1}{2} \cdot I(|d| \leq 1) \\
\text{Triangular} &: (1 - |d|) \cdot I(|d| \leq 1) \\
\text{Epanechnikov} &: \frac{3}{4}(1 - d^2) \cdot I(|d| \leq 1) \\
\text{Gauss} &: \frac{1}{\sqrt{2\pi}} e^{(-\frac{d^2}{2})} \cdot I(|d| \leq 1)
\end{aligned} \tag{4}$$

Pode-se perceber, pela definição dos quatro *kernels* considerados, que os valores de d necessitam estar dentro de um certo intervalo para que as similaridades produzidas sejam comparáveis. Isto é obtido pela normalização das

distâncias d em relação à distância do $k+1$-ésimo vizinho mais próximo do exemplo de teste. Assim, as distâncias $d_i, i = 1, \ldots, k$ são transformadas em $\hat{d}_i, i = 1, \ldots, k$, fechadas no intervalo $[0, 1]$, pela Equação 5:

$$\hat{d}_i = \frac{d_i}{d_{k+1}} \tag{5}$$

Utilizando-se as idéias do wkNN, desenvolveu-se o método de ponderação de *rankings* denominado wAR (do inglês *weighted Average Rank*). De maneira similar ao AR, ele utiliza as médias das posições do *ranking* dos k vizinhos selecionados para compor o *ranking* predito. A diferença para o wAR é que este utiliza médias ponderadas para tal tarefa. Assim, seja $R_{i,j}$ a posição no *ranking* do algoritmo a_j $(j = 1, \ldots, n)$ na base de dados i, onde n é o número de algoritmos. A posição média ponderada no *ranking* para cada a_j é dada pela Equação 6. O *ranking* predito é obtido ordenando-se os valores de $w\bar{R}_j$ e ajustando suas posições de acordo.

$$w\bar{R}_j = \frac{\sum_{i=1}^{k} w_i * R_{i,j}}{\sum_{i=1}^{k} w_i} \tag{6}$$

1.4.1.3. Resultados

Nesta seção, discutem-se os resultados da recomendação de algoritmos obtida pelo método de *rankings* baseado em vizinhos mais próximos. Os experimentos foram realizdos segundo um esquema de validação cruzada. Especificamente, adotou-se o LOO comentado na Seção 1.2.3.2, dada à pequena quantidade de meta-exemplos disponíveis. Nele, cada meta-exemplo é utilizado como teste do meta-modelo gerado pelo conjunto de treinamento. Em cada iteração, a acurácia de predição em relação ao *ranking* ideal conhecido é medida pelo coeficiente de Spearman, que tem sido comumente empregada para medir a acurácia de *rankings* gerados por métodos de meta-aprendizado [Brazdil et al. 2009]. Ele é calculado pela Equação 7, para *rankings* arbitrários R e Q de itens $i = 1, \ldots, n$:

$$r_S = \frac{\sum_{i=1}^{n}(R_i - \overline{R})(Q_i - \overline{Q})}{\sqrt{(\sum_{i=1}^{n}(R - \overline{R})^2 \sum_{i=1}^{n}(Q - \overline{Q})^2)}} \tag{7}$$

onde R_i (Q_i) corresponde à posição do item i em R (Q) e $\overline{R}$ ($\overline{Q}$) representa a média dessas posições. Por sua definição, percebe-se que o r_S avalia a monotonicidade entre os R e Q, ou seja, se suas variações estão relacionadas. Caso os valores nos *rankings* apresentem tendência em crescer ou decrescer juntos, então há uma correlação positiva, com máximo em 1. Se os valores de um *ranking* crescem enquanto os do outro decrescem, então há uma correlação negativa, com mínimo em -1. Uma correlação de 0 indica que os *rankings* não estão correlacionados.

Para estabelecer se a estimativa de desempenho de uma abordagem específica pode ser considerada adequada ou não para o conjunto de meta-exemplos em estudo, é importante ter uma abordagem de comparação. Em AM, geralmente estratégias de predição ingênuas, como classificar um novo exemplo segundo a classe majoritária no conjunto de treinamento (conhecida como classe padrão), são empregadas para avaliar métodos mais complexos. No caso de *rankings*, uma abordagem similar consiste em resumir a informação dos meta-atributos de saída de todos os meta-exemplos de treinamento em um único *ranking*. Isto é obtido utilizando-se o método *Average Ranks* (AR) apresentado na Equação 3. Desta forma, obtém-se o chamado *ranking* padrão (RP), que serve de base para avaliar a qualidade do método de *ranking*.

Limites do kNN

Anteriormente, apresentou-se o RP como base de comparação para kNN. Uma análise complementar de seu desempenho pode ser realizada considerando os limites superior e inferior do método. No arcabouço de LOO, o limite superior do kNN é obtido selecionando-se os k vizinhos no conjunto de treinamento que produzam as maiores correlações com o *ranking* ideal de cada meta-exemplo de teste, independente das distâncias entre os meta-exemplos no espaço de meta-atributos de entrada, e aplicando-se sobre eles o AR para a geração das predições. Desta forma, obtém-se a máxima acurácia média do kNN quando empregado sobre um determinado grupo de problemas. Para calcular o limite inferior, escolhe-se, para cada valor de k, meta-vizinhos aleatórios no conjunto de treinamento, novamente desconsiderando as distância para o meta-exemplo de teste, e constrói-se os *rankings* utilizando o AR. Com isso, simula-se a situação em que o desempenho do kNN depende tão somente da agregação fortuita de *rankings* de k meta-exemplos. Por meio desses dois limites, a intenção é contextualizar o desempenho do kNN, apreciando quão bom o método é e quão melhor (teoricamente) ele ainda pode ser.

A Figura 1.2 exibe os limites inferior e superior do kNN variando-se $k = 1, \ldots, 30$. Os valores apresentados são as médias de acurácia (utilizando o r_S) dos *rankings* obtidos no decorrer do LOO para os 49 problemas considerados. O limite superior é dado pela linha tracejada. Percebe-se que o desempenho do kNN inicia superior a 0.9 e vai decrescendo com o número de vizinhos. Isso é natural, dado que os vizinhos neste limite são selecionados segundo sua correlação com os meta-exemplos de teste. Então, à medida que o AR vai sendo empregado com valores de k crescentes, espera-se uma redução de desempenho, pois meta-exemplos com *rankings* ideais cada vez mais heterogêneos vão sendo utilizados na construção das predições. Quando todos os 48 meta-exemplos de treinamento são aplicados ao AR, o limite superior coincide com o RP (linha completa na figura).

No caso do limite inferior (*boxplots*[1] obtidos a partir de 100 amostragens

[1] Os 5 traços horizontais de cada *boxplot* representam, respectivamente: o mínimo, o primeiro quartil, a mediana, o terceiro quartil e o máximo da amostra de observações

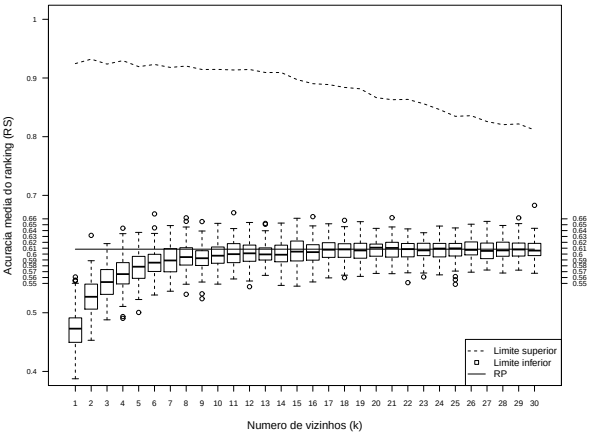

Figura 1.2. Limites superior e inferior do método kNN.

randômicas), o emprego de poucos meta-exemplos resulta em baixa acurácia em relação ao RP, pois os *rankings* considerados não apresentam uma grande correlação com os *rankings* dos meta-exemplos de teste. A situação pode ser explicada pela aleatoriedade da escolha dos meta-exemplos. Este comportamento é suavizado com o aumento de k. Assim, com a utilização de mais meta-exemplos na composição pelo AR, a probabilidade de que melhores *rankings* sejam utilizados cresce, o que leva, gradativamente, a acurácias próximas à obtida pelo RP. De fato, a partir de $k = 17$, a mediana dos *boxplots* varia em torno de 0.608, a acurácia média de RP.

Desempenho do kNN

A Figura 1.3 exibe os resultados da aplicação do kNN e de sua extensão wkNN para a predição de *rankings* nos 49 problemas em uso. As acurácias médias do LOO utilizando o r_S são mostradas. São utilizadas 4 funções de *kernel* típicas para o método: Retangular, Triangular, Epanechnikov e Gauss. O *kernel* Retangular pondera igualmente todos os vizinhos e corresponde exatamente ao método kNN ordinário. Com o número de vizinhos variando em $k = 1, \ldots, 30$, os desempenhos de todas as versões do wkNN aumentam com o valor de k, inicialmente. À medida que a vizinhança ultrapassa um certo limiar, os desempenhos vão se deteriorando, pois vizinhos de menor relevância

consideradas. Os círculos indicam observações usuais da amostra, considerando ela segue uma distribuição normal.

aos meta-exemplos de teste vão sendo considerados na construção dos *rankings*. Esta degradação ocorre primeiro com o wkNN Retangular. A partir de $k = 4$, ele exibe um desempenho errático em direção ao RP, indicando que o método é bastante sensível aos valores de k. Este comportamento é compartilhado, em certo grau, com o *kernel* Gauss, embora este resulte em acurácias médias levemente superiores no decorrer de todo o intervalo de k.

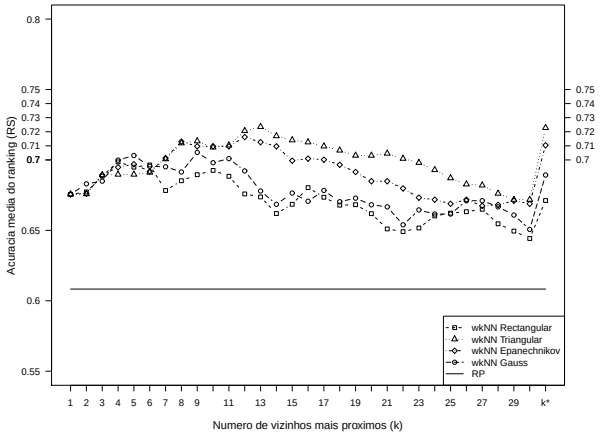

Figura 1.3. Desempenho do método wkNN com caracterização de dados utilizando o conjunto STATLOG e 4 funções de *kernel*

Os métodos wkNN Triangular e Epanechnikov apresentam vantagens mais notáveis em relação ao wkNN Retangular. Para qualquer $k > 6$, ele são superiores, com primazia do primeiro *kernel*. Seus melhores resultados são, respectivamente: 0.723 ($k = 13$) e 0.716 ($k = 12$). Para $13 < k \leq 30$, eles resultam em um suave decaimento na acurácia média. Para entender seus comportamentos, pode-se inspecionar a Figura 1.4, que ilustra a forma dessas 2 funções de acordo com as definições apresentadas pela Equação 4. Elas operam transformando as distâncias normalizadas entre o meta-exemplo de teste corrente e seus vizinhos em similaridades, que são então utilizadas como pesos pelo wkNN. Nota-se que com o *kernel* Triangular, o peso de vizinhos rapidamente se atenua com sua distância, fazendo com que aqueles mais distantes logo se tornem menos influentes na composição do *raking*. No caso do *kernel* Epanechnikov, o detrimento da influência dos vizinhos é menos acentuado. Para o *kernel* Gauss, tal declínio é ainda mais suave, dada sua forma mais próxima a do *kernel* Retangular.

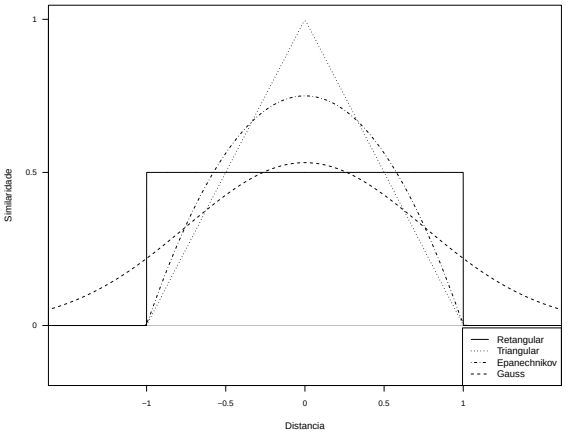

Figura 1.4. Ilustração da forma das funções de *kernel* Retangular, Triangular, Epanechnikov e Gauss

A fim de lidar com a necessidade de escolha de um valor apropriado para k, aplicou-se um procedimento interno de LOO aos meta-exemplos de treinamento para selecionar um número de vizinhos específico para cada problema, sobre $k = 1, \ldots, 15$. O k de melhor desempenho foi escolhido. Tal valor é representado pela entrada k^* na Figura 1.3. Essa abordagem selecionou em média (com desvio padrão) 5.645 (1.862), 12.604 (1.21), 10.687 (1.925) e 8.062 (1.767) vizinhos para os wkNN com *kernel* Retangular, Triangular, Epanechnikov e Gauss, respectivamente, resultando em acurácias médias de 0.671, 0.722, 0.710 e 0.689. Aplicando-se os testes de Friedman e Dunn (comentados no Capítulo 2 de [Brazdil et al. 2009]) para comparar estatisticamente os desempenhos dos quatro métodos, não se observaram diferenças significantes entre eles. Quando comparados com o RP, todos apresentaram desempenho estatisticamente superior.

1.4.1.4. Comentários finais

O foco da pesquisa apresentada parcialmente nesse estudo de caso foi investigar a utilização de meta-aprendizagem como ferramenta de suporte à classificação de dados de expressão gênica. Tal empreendimento visou contribuir em duas áreas. Da perspectiva da aplicação, pretendeu-se desenvolver métodos que pudessem auxiliar o usuário não especialista em sistemas de classificação, tipicamente biólogos e cientistas da área médica, na

tarefa de escolher as soluções mais apropriadas para seu problema. Isso facilitaria a análise dos experimentos realizados, fazendo com que a utilização de *microarrays* no diagnóstico de doenças se torne mais acessível. Do ponto de vista da meta-aprendizagem, a intenção foi expandir sua prática corrente em recomendação de algoritmos utilizados para classificação de dados. Conforme pode ser observado a partir da literatura especializada (vide o Capítulo 3 de [Brazdil et al. 2009]), métodos de sugestão baseados nessa abordagem têm exibido considerável sucesso quando aplicados a problemas oriundos de âmbitos diversos, como os representados pelos conjuntos de dados disponíveis em repositórios de propósito geral (por exemplo, o da UCI, de endereço `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`). Entretanto, sua adequação a problemas reais de natureza semelhante tem recebido menos atenção. Em situações como esta, o desempenho dos algoritmos de AM e os dados tendem a se apresentar mais homogêneos, tornando incerta a adequação dos métodos tradicionais de meta-aprendizagem ao problema de recomendação. Os resultados obtidos forneceram evidências significantes da viabilidade da aplicação de meta-aprendizado para a recomendação de algoritmos de AM no contexto de dados de expressão gênica.

1.4.2. Seleção de Modelos de Previsão

1.4.2.1. Contexto

Uma série temporal é um conjunto de observações de um fenômeno ordenadas no tempo [Box e Jenkins 1970]. Pode-se citar como exemplos de séries temporais o consumo mensal de energia elétrica de uma casa, registrado durante um ano, ou a temperatura de uma reação química observada a cada segundo durante uma hora, ou ainda as vendas diárias de um produto no decorrer de um mês. A previsão de séries temporais é fundamental para diminuir os riscos na tomada de decisões nos contextos onde as séries estão inseridas, e para antecipar quadros futuros que servirão para o planejamento atual de estratégias. Segundo [Chatfield 2001], os métodos de previsão podem ser classificados em três tipos:

1. Métodos subjetivos: são baseados no julgamento subjetivo, na intuição ou no conhecimento de um ou mais especialistas do domínio da previsão.
2. Métodos univariados: onde a série é prevista usando apenas os valores passados da própria série, relacionados através de modelos matemáticos.
3. Métodos multivariados: onde a previsão de uma série depende de, no mínimo, uma ou mais variáveis adicionais, chamadas de variáveis preditoras ou explanatórias, relacionadas através de modelos matemáticos.

Diversos modelos de séries temporais já foram propostos na literatura. De acordo com [Chatfield 2001], a literatura em séries temporais tem se concentrado em como implementar modelos particulares de previsão, enquanto que

os usuários necessitam muito mais de estratégias para a previsão, ou seja, determinar quando usar um modelo e como definir parâmetros do modelo. De fato, assim como ocorre nos problemas clássicos de AM, em previsão de séries temporais tem-se uma grande variedade de modelos candidatos para realizar a previsão. Além disso, não se pode apontar um determinado modelo como sendo o melhor independentemente do problema em questão. A seleção de modelos de previsão depende das características das séries sendo previstas e dos critérios usados na avaliação dos modelos. O desempenho de um modelo pode variar bastante dependendo da série. Esses aspectos, que incluem grande quantidade e diversidade de modelos, fazem da seleção de modelos de previsão de séries temporais um bom candidato para a aplicação de sistemas de meta-aprendizado.

Os primeiros trabalhos que exploraram a idéia de usar algoritmos de aprendizado para seleção de modelos de previsão datam ainda da década de 90 [Chu e Widjaja 1994, Arinze 1994, Venkatachalan e Sohl 1999]. Embora possam ser definidos atualmente dentro do contexto de meta-aprendizado, esses trabalhos ainda não tinham conexão direta com o tema. De uma forma geral, eles tratam a seleção de modelos como um problema de classificação em que o atributo classe representa o melhor modelo para prever uma dada série, e os atributos preditores são as características da série temporal. Cada exemplo de treinamento nesse contexto armazena as características descritivas de uma dada série temporal e é associado a um atributo classe (melhor modelo de previsão para a série). O valor do atributo classe para uma dada série é tipicamente definido de forma experimental, realizando um torneio *out-of-sample* entre os modelos candidatos. A série é associada à classe relativa ao modelo candidato que obtiver o melhor desempenho no torneio. A partir de um conjunto desses exemplos de treinamento, um algoritmo de aprendizado será capaz de associar características de novas séries temporais fornecidas como entrada aos melhores modelos de previsão.

Em [Arinze 1994], os autores usaram o algoritmo ID3 para indução de árvores de decisão usadas para selecionar entre seis modelos de previsão. Como exemplos de treinamento, os autores usaram 67 séries reais extraídas de domínios econômicos. Cada série foi descrita por seis características: nível de agregação (trimestral ou anual), número de turning points da série, autocorrelações, presença de tendência, coeficiente de determinação e o erro obtido pelo modelo de regressão linear.

Em [Chu e Widjaja 1994], um sistema baseado em redes neurais é usado para selecionar entre seis modelos de alisamento exponencial. Inicialmente, uma rede MLP é usada para identificar o tipo de padrão da série (estacionário, com tendência linear, sazonal,...) a partir dos valores das autocorrelações da série. Em seguida, uma outra rede MLP é usada para selecionar o melhor modelo de previsão a partir do padrão da série identificado anteriormente, do horizonte de previsão e da indústria a que a série pertencia. Nesse trabalho, os autores usaram séries simuladas para gerar o conjunto de exemplos de

treinamento.

Em [Sohl e Venkatachalam 1995], a seleção do melhor modelo para uma série fornecida como entrada era feita em dois passos. Os 9 modelos de previsão considerados nesse trabalho foram divididos em três grupos de acordo com as similaridades entre os modelos. No primeiro passo, uma rede MLP era responsável inicialmente por definir o melhor grupo de modelos para a série fornecida como entrada. Em seguida, uma outra rede MLP selecionava um dos modelos pertencentes ao grupo. Para cada grupo de modelos, os autores treinaram uma rede MLP. Separando os modelos em grupos e realizando a seleção de modelos em níveis, os autores fazem uso indireto de um conhecimento *a priori*, que é a existência de similaridades entre os modelos. Para gerar o conjunto de exemplos de treinamento, os autores usaram uma amostra aleatória das séries da M-Competition, descritas com as seguintes características: tamanho da série, período entre observações (mensal, trimestral e anual), tipo dos dados (econômicos, industriais,...), tendência básica, tendência recente e variabilidade da série. Uma descrição mais detalhada dessa proposta pode ser encontrada em [Venkatachalan e Sohl 1999], um trabalho posterior dos mesmos autores.

Alguns trabalhos como [Lee e Jhee 1999, Reynolds et al. 1995, Chenoweth et al. 2000] usam algoritmos de aprendizado (especificamente redes neurais artificiais) para auxiliar a análise de funções de autocorrelação. Nesses trabalhos, as características descritivas das séries temporais são as autocorrelações seriais e os modelos candidatos são os diferentes modelos da classe de Box-Jenkins.

Pode-se citar ainda trabalhos correlatos que usaram técnicas oriundas da estatística para a seleção de modelos. Em [Shah 1997], os autores usaram diferentes métodos de análise discriminante para selecionar entre três modelos de previsão candidatos. Nesse trabalho, o conjunto de exemplos foi gerado usando as 203 séries trimestrais da M-Competition que eram descritas usando 26 características. A técnica ANOVA é usada em [Zhang et al. 2001] e [Zhang 2001] para identificar diferenças no desempenho de redes neurais para previsão. [Sanders e Manrodt 2003] usa ANOVA para auxiliar a seleção entre modelos quantitativos e métodos qualitativos de previsão.

O primeiro trabalho que adotou explicitamente o uso do termo meta-aprendizado no contexto de previsão de séries temporais e a estabelecer uma conexão entre as áreas foi o trabalho de Prudêncio e Ludermir [Prudêncio e Ludermir 2004a]. Esse trabalho desenvolveu dois estudos de caso. No primeiro estudo de caso, um algoritmo de aprendizado simples foi usado para selecionar entre dois modelos de séries temporais: Alisamento Exponencial Simples e a *Time Delay Neural Network* (TDNN) [Lang e Hinton 1988], usados para prever 99 séries estacionárias coletadas do repositório *Time Series Data Library* (TSDL) [Hyndman e Muhammad 2004]. No segundo estudo de caso, a abordagem NOEMON [Kalousis e Theoharis 1999] foi usada para gerar um *ranking* e selecionar três modelos usados para prever as 645 séries

anuais da M3-Competition [Makridakis e Hibon 2000]. A abordagem NOEMON é um método desenvolvido pela comunidade de meta-aprendizado capaz de fornecer um *ranking* de algoritmos através da combinação de diferentes algoritmos de aprendizado.

Em [Santos et al. 2004], os autores propuseram o uso de outra abordagem original, desenvolvida dentro da área de meta-aprendizado, para ordenar e recomendar modelos de previsão. Nesse trabalho, a abordagem de *Zoomed-Ranking* [Brazdil et al. 2009] foi aplicada para ordenar três modelos neurais de previsão: (1) o modelo TDNN; (2) o modelo *Time-Lagged* RBF [Haykin 1998]; e (3) o modelo *SOM with Temporal Activity Diffusion* (SOMTAD) [Principe et al. 2002]. Um diferencial do *Zoomed-Ranking* é que ele é capaz de recomendar algoritmos levando em consideração tanto o desempenho como o custo de treinamento dos algoritmos. No caso dos modelos de previsão, esses critérios corresponderam ao erro de previsão e ao tempo de estimação de parâmetros dos modelos. Experimentos foram realizados usando 80 séries temporais mensais extraídos do repositório TSDL.

Em [Prudêncio e Ludermir 2004b], os autores aplicaram a abordagem de meta-regressão para combinar dois modelos usados na previsão das séries da M3-Competition. Nessa abordagem, uma rede MLP é usada para definir pesos numéricos para os modelos candidatos de acordo com as características da série. A previsão combinada é a média das previsões individuais dos modelos ponderada pelos pesos associados.

Mais recentemente, em [Wang et al. 2009] os autores aplicaram meta-aprendizado usando inicialmente uma rede SOM para mapear o desempenho de quatro modelos de previsão. Em seguida, um algoritmo de indução de regras foi usado para relacionar as características das séries temporais com o desempenho dos modelos. Um ponto interessante desse trabalho é a vizualização do desempenho dos modelos através das redes SOM. Dessa forma, foi possível adquirir insights interessantes para a aplicação dos modelos. Finalmente em [Lemke e Gabrys 2010], um estudo de meta-aprendizado foi desenvolvido com um número mais expressivo de modelos candidatos, incluindo métodos de combinação de modelos. Nesse trabalho os autores usaram como meta-aprendiz uma rede neural feedforward, árvores de decisão, *Support Vector Machines* e o método de *Zoomed-Ranking*. Um extensão do métodos de *Zoomed-Ranking* foi aplicada para definir pesos para combinação linear dos modelos candidatos (e não apenas para ordenar e selecionar um único modelo).

A fim de exemplificar como meta-aprendizado pode ser utilizada para a seleção de séries temporais, apresenta-se a seguir parte dos experimentos realizados em [Prudêncio e Ludermir 2004a]. Nesse estudo de caso, a abordagem NOEMON foi adotada para formar *rankings* de três modelos de previsão: *Random Walk* (RW), Alisamento Exponencial Linear de Holt (HL) e o modelo AutoRegressivo (AR). Esses modelos foram usados para realizar previsões a um passo de séries anuais da M3-Competition.

1.4.2.2. Materiais e métodos

Nesta seção, descrevem-se as bases de dados, as medidas de caracterização e de desempenho utilizadas para a construção dos meta-exemplos, e o método aplicado para realizar o meta-aprendizado.

Conjuntos de dados
Um aspecto que favorece a aplicação de meta-aprendizado para seleção de modelos de séries temporais é a disponibilidade de dados (pelo menos no caso de séries univariadas). De fato, podem ser citados repositórios como a TSDL e ainda bases de dados usadas em competições como a M3-Competition e outras competições de previsão de séries temporais. A quantidade de problemas disponíveis nesses repositórios é em geral muito maior que a quantidade de bases de problemas de classificação e regressão usados nos trabalhos usuais de meta-aprendizado.

Nesse estudo de caso, foi investigada a seleção de modelos para prever as séries anuais da M3-Competition (dados disponíveis em `http://www.forecasters.org/data/m3comp/m3comp.htm`). Essas séries são relacionadas a domínios econômicos e demográficos e representam uma amostra conveniente para realização de experimentos em previsão de séries temporais [Shah 1997]. De fato, as séries da M3-Competition foram usadas em diferentes trabalhos como amostra padrão para avaliar estratégias de seleção de modelos [Shah 1997, Venkatachalan e Sohl 1999]. Aqui, foram utilizadas as 645 séries anuais da competição.

Construção de meta-dados
Os meta-exemplos foram gerados a partir da descrição das 645 séries e da aplicação dos três modelos de previsão para coleta das informações de desempenho. Cada meta-exemplo foi associado a uma série temporal específica e armazenou: (1) cinco medidas descritivas como meta-atributos de entrada e (2) o erro de previsão obtido por cada modelo na série em questão, que são utilizados para a definição dos meta-atributos alvo. As medidas de caracterização utiilizadas nesse estudo de caso são apresentadas na Tabela 1.6. As primeiras quatro características são estatísticas descritivas calculadas diretamente a partir dos dados da série. A característica TIPO por sua vez é uma informação contextual fornecida pelos autores da M3-Competition.

Tabela 1.6. Medidas de caracterização

Medida	Descrição
TAM	Tamanho da série
TEN	Tendência da séris pela estatística t da inclinação do modelo de regressão linear
PTP	Proporção de *Turning Points*
PCA	Primeiro Coeficiente de Autocorrelação
TIP	Tipo da série (*micro*, *macro*, *industry*, *finances*, *demographich* ou *others*)

Para gerar as informações de desempenho dos modelos para uma dada

série, foi aplicado uma avaliação *out-of-sample* com os dados da série. Nessa avaliação, os últimos seis anos da série foram usados para testar os modelos e os dados restantes iniciais foram usados para calibrar os parâmetros dos modelos (e realizar as previsões). Esse número de pontos de teste foi definido seguindo as regras da M3-Competition. A medida de Erro Absoluto Médio (EAM) foi usada com informação sobre o comportamento preditivo do modelo.

Método de meta-aprendizado

Nesse estudo de caso, a abordagem NOEMON foi utilizada a fim de construir a recomendação dos algoritmos no meta-aprendizado. Basicamente, ela é capaz de gerar um *ranking* de modelos para cada série temporal fornecida como entrada, de acordo com as características da série e de uma função de erro (no caso o EAM) que se deseja minimizar.

Para gerar um *ranking* de 3 modelos, o NOEMON usa $3 * (3 - 1)/2 = 3$ aprendizes (classificadores), cada um associado a um par específico de modelos. Nessa implementação de NOEMON, redes MLP foram usados como classificadores. Cada classificador recebeu como entrada a base de meta-exemplos apresentada nas seções anteriores com a diferença que o atributo alvo da classificação corresponde ao melhor modelo considerando o par de modelos associado ao classificador (RW ou HL, AR ou HL, RW ou AR). Uma vez treinados os classificadores, a abordagem gera um *ranking* de modelos combinando as respostas dos classificadores individuais através do número de votos de cada modelo.

1.4.2.3. Resultados

Nesta seção, avalia-se a qualidade dos *rankings* gerados pelo NOEMON. Para isso, foi realizado um experimento de validação cruzada com os 645 meta-exemplos do estudo de caso. Desta forma, uma parte dos meta-exemplos foi utilizado para treinamento dos classificadores e outra parte para avaliar a qualidade das recomendações.

A qualidade de um *ranking* gerado para uma dada série foi avaliada medindo a similaridade com o *ranking* ideal, que representa a ordem correta dos modelos levando em conta o erro de previsão EAM obtido. Nos experimentos conduzidos, foi usado o coeficiente de correlação de Spearman (r_S) entre os *rankings*. Ele assume valores entre -1 e 1, tal que valores maiores indicam mais similaridade entre o *ranking* sugerido e o *ranking* ideal para a série, sendo, portanto, preferidos.

O meta-aprendiz NOEMON foi comparado com um método de *ranking* padrão (RP), onde o mesmo *ranking* é sugerido para todas as séries. Esse *ranking* foi definido observando o número de vezes em que cada modelo de previsão foi o melhor para o conjunto de séries em questão. Nesse estudo de caso, o RP foi HL-RW-AR. Na Tabela 1.7, mostra-se a média de r_S para os

rankings gerados por NOEMON e pelo ranking default sobre as execuções do experimento de validação cruzada. Como pode ser visto, os *rankings* gerados pelo meta-aprendiz NOEMON foram em média mais correlacionados com o ranking ideal, gerando um ganho na medida SRC de 0.21 pontos em relação ao método RP.

Tabela 1.7. Média do coeficiente Spearman obtidos por NOEMON e RP para as 215 séries de teste. A terceira linha mostra o ganho obtido com o uso de NOEMON.

Método	SRC
NOEMON	0.36
RP	0.15
Ganho Obtido	0.21

1.4.2.4. Comentários finais

Conforme visto anteriormente, diferentes trabalhos aplicaram os conceitos e idéias de meta-aprendizado para a seleção de modelos de séries temporais. Diversos pontos permanecem em aberto. Em sua totalidade, as pesquisas têm focado apenas em modelos univariados de previsão. Entretanto, outras abordagens, em especial os modelos multivariados, são bastante importantes na prática. Nessa linha, um dos desafios é definir meta-atributos adequados para descrever os problemas de previsão com múltiplas séries. Os meta-atributos usuais descrevem séries univariadas, geralmente identificando características de sazonalidade, tendência e ruído nas séries. Embora os resultados obtidos com essa caracterização tenham se mostrado interessantes, a aplicação desses meta-atributos em problemas multivariados não é direta. Assim, mais estudos para lidar com esse problema são requeridos.

Outra direção de pesquisa é o uso meta-aprendizado para combinação de modelos, o que foi explorando em poucos trabalhos (por exemplo, [Lemke e Gabrys 2010]). Aspectos importantes dessa aplicação que podem ser estudados é como selecionar ou filtrar os modelos a serem combinados. Também pode-se utilizar meta-aprendizado para identificar modelos pouco correlacionados a fim de combiná-los.

Destaca-se ainda a necessidade de se estudar a seleção dinâmica de modelos para múltiplos horizontes de previsão. As características de uma série temporal podem mudar ao longo do tempo, especialmente em contextos mais dinâmicos. Nesses casos, o método de meta-aprendizado deveria ser capaz de perceber as mudanças dos conceitos e dos processos que geraram a série. Os trabalhos anteriores fazem recomendações estáticas, ou seja, que não são

capazes de se adaptar à medida que novos pontos da série são observados. Além disso, a seleção de modelos levando em consideração múltiplos horizontes de previsão em geral não tem recebido uma análise mais cuidadosa.

Finalmente, pode-se estudar a integração de meta-aprendizado com conhecimento especialista. Embora, o conhecimento especialista seja, em geral, difícil de adquirir e elicitar, na literatura de análise de séries temporais existe uma gama de resultados (em especial para modelos lineares mais simples) que podem ser integrados ao processo de seleção. Por exemplo, alguns modelos de previsão partem de premissas como a existência de sazonalidade ou tendência na série, ou ainda se diferenciam pelo tipo de sazonalidade (aditiva ou multiplicativa), dentre outros aspectos. Assim, dependendo da série, alguns modelos poderiam ser descartados como modelos candidatos para uma posterior seleção por parte de meta-aprendizado. Como outro exemplo, em [Venkatachalan e Sohl 1999], modelos são agrupados a partir de conhecimento *a priori* e um meta-aprendiz é construído para seleção de modelos dentro de cada grupo. Outras formas de integração de meta-aprendizado com conhecimento *a priori* podem ser propostas no futuro.

1.4.3. Problema do caixeiro viajante

1.4.3.1. Contexto

Meta-aprendizado também tem sido utilizado com sucesso em problemas de otimização combinatorial. Tipicamente, em aplicações desse tipo, procura-se pela melhor solução dentre um conjunto de possíveis soluções. A qualidade delas é estimada por uma função denominada função objetivo. Assim, a solução buscada em um problema de otimização pode ser aquela que minimize o valor da função objetivo (problema de minimização) ou aquela que o maximize (problema de maximização). A solução ótima é obtida encontrando a melhor combinação de valores para as variáveis do problema. Um exemplo clássico estudado em otimização é o problema do caixeiro viajante (TSP, do inglês *Traveling Salesman Problem*). Notas históricas sobre seu desenvolvimento podem ser encontradas em [Schrijver 2005].

Diversas variações do TSP podem ser encontradas na literatura. Na versão tradicional do problema, um caixeiro viajante deve visitar N cidades em sua área de vendas. O caixeiro começa de uma cidade base, visita cada uma das N cidades uma única vez e retorna à cidade base no final. A cada viagem entre duas cidades esta associado um custo proporcional à distância percorrida. O caixeiro deve percorrer a rota com a menor distância total. É fácil encontrar vários problemas reais de otimização similares ao TSP. Dentre eles, pode-se mencionar como exemplos (vide o Capítulo 8 de [Goldbarg e Luna 2005] para outras aplicações do TSP): roteamento de veículos, planejamento de produção, agendamento de tarefas, e corte e empacotamento.

Pra fins práticos, o TSP pode ser representado por um grafo ponderado em que cada nó está associado a uma cidade e o valor do peso associado a uma aresta que conecta dois nós indica a distância entre as cidades conectadas.

Quando todas os pares de nós estão conectados por uma aresta, o grafo é dito completamente conectado. Caso contrário, o grafo é dito parcialmente conectado. A Figura 1.5 ilustra um exemplo de TSP por um grafo ponderado parcialmente conectado, indicando que alguns pares de cidade não estão diretamente ligadas.

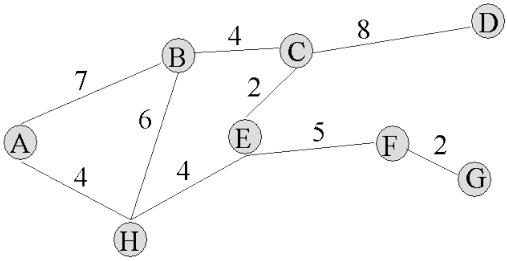

Figura 1.5. Exemplo do problema do caixeiro viajante

Apesar de aparentemente simples, o TSP é um problema NP-Difícil, com seu computacional apresentando um crescimento fatorial em função do número de cidades. Para um TSP com n cidades completamente interconectadas, existem $(n-1)!$ possíveis rotas cujo cálculo de custo por um método exato demanda $n!$ operações de soma até que sejam encontradas todas as possíveis soluções. Como exemplo, para calcular todas as possibilidades de caminho em um TSP com 50 cidades, seriam calculadas aproximadamente 10^{64} operações de soma, que terminariam após 10^{45} séculos em um computador com capacidade de processamento de um milhão de operações por segundo. Assim, o TSP é um dos problemas mais estudados nas áreas de otimização e busca. Até hoje, não foi encontrada um método exato eficiente para o caso geral. Quem resolver este problema para o caso geral receberá um prêmio de $US\$$ 1.000.000 do Clay Mathematics Institute, nos Estados Unidos.

Por ser impraticável encontrar a melhor solução por meio do cálculo de todas as possíveis soluções para a grande maioria de exemplos de TSP, várias heurísticas têm sido propostas para procurar boas soluções locais em um intervalo de tempo razoável. Uma heurística é uma técnica que procura boas soluções com um custo computacional razoável [Voß2001]. Ela pode ser formada por uma regra ou conjunto de regras. É esperado que ela encontre uma boa solução, mas não existem garantias de que ela encontre a solução ótima. Em geral, uma heurística primeiro encontra uma boa solução inicial factível e depois procura melhorá-la. No contexto de TSP, tem-se heurísticas construtivas ou de refinamento. Enquanto as primeiras inserem uma cidade à solução em cada iteração, as segundas procuram por soluções melhores na vizinhança da solução construída.

Uma meta-heurística, por sua vez, é uma estratégia mais elaborada, que

guia uma heurística durante a solução de um dado problema [Voß2001]. Elas utilizam estratégias que buscam explorar o espaço de soluções factíveis de forma a evitar a convergência do algoritmo para ótimos locais. Em No Capítulo 8 de [Goldbarg e Luna 2005], são listadas as principais meta-heurísticas utilizadas em exemplos de TSP, dentre as quais podem ser citadas: Algoritmos Genéticos (AG), Busca Tabu (BT), *Simulated Annealing* (SA) e *Greedy Randomized Adaptive Search Procedure* (GRASP).

Como as diversas variações do TSP apresentam diferentes características, é difícil, dado um problema de TSP, definir qual o algoritmo de otimização mais apropriado. De maneira geral, não existe um algoritmo que apresente o melhor desempenho para todos os tipos de problema. Quando um algoritmo é superior a outro(s) em uma classe de problemas, ele será inferior em outra(s) classe(s). Para a escolha do algoritmo mais adequado, uma alternativa é testar experimentalmente um grupo de algoritmos candidatos e escolher aquele que apresentar melhor desempenho. Se o número de algoritmos a serem avaliados for grande, o custo computacional dessa alternativa pode ser muito elevado. Se, por outro lado, o grupo for formado por poucos algoritmos, algoritmos que apresentariam um bom desempenho podem não fazer parte do grupo. Uma outra alternativa que tem se mostrado promissora é associar características ou propriedades de uma classe de problemas aos algoritmos mais promissores para resolver os problemas dessa classe. Nessa alternativa, meta-aprendizado tem sido utilizado em em diversos trabalhos para estimar uma função que faça essa associação [Smith-Miles 2009a, Kanda et al. 2010, Smith-Miles et al. 2010, Kanda et al. 2011].

Para ilustrar o emprego de meta-aprendizado para a escolha da meta-heurística mais adequada para um problema de TSP específico, vai-se apresentar a seguir um subconjunto dos experimentos conduzidos em [Kanda et al. 2010, Kanda et al. 2011]. O texto dessa aplicação é baseado nesses trablhos.

1.4.3.2. Materiais e métodos

Nesta seção, descrevem-se as medidas de caracterização e de desempenho utilizadas para a construção dos meta-exemplos, e o método aplicado para realizar o meta-aprendizado.

Construção de meta-dados

Para a utilização de meta-aprendizado para selecionar os algortimos mais promissores, o primeiro passo é a definição dos meta-atributos. Diferente da utilização do meta-aprendizado para classificação de dados, em que os meta-atributos de entrada são características dos conjuntos de dados, em aplicações de otimimização, eles correspondem a características dos problemas a serem otimizados. No caso particular do TSP, pode-se utilizar características do grafo que representa os exemplos de TSP. Um vez definidos os meta-atributos de en-

trada, uma abordagem semelhante à utilizada para a recomendação de algoritmos de indução de classificadores pode ser empregada para a recomendação de algoritmos de otimização. A Tabela 1.8 ilustra um conjunto de características que podem ser extraídas para compor os meta-atributos de entrada. Outras características do TSP também podem ser utilizadas, como, por exemplo, medidas provenientes da área de redes complexas. Mais alternativas são discutidas em [Smith-Miles et al. 2010].

Tabela 1.8. Meta-atributos para TSP.

Medida	Descrição
NOV	Número de vértices (cidades) do grafo
NOA	Número de caminhos (arestas) ligando pares de cidades
MIP	Menor valor dentre os pesos das arestas
MAP	Maior valor dentre os pesos das arestas
MEP	Média dos pesos das arestas
DPP	Desvio-padrão dos pesos das arestas
MDP	Mediana dos pesos das arestas
LIP	Soma dos V menores pesos (limite inferior para soluções obtidas)
LSP	Soma dos V maiores pesos (limite superior para soluções obtidas)
MHP	Razão entre o número de arestas e a soma do inverso dos valores dos pesos
MEV	Médias dos pesos associados aos vértices
DPV	Desvio padrão dos pesos associados aos vértices

Para a composição dos meta-exemplos, é necessário ainda definir o meta-atributo alvo adequado. No presente estudo de caso, ele corresponde ao algoritmo de otimização, dentre um conjunto de candidatos, que melhor resolve o problema de TSP. Assim, a cada instância de TSP, associa-se o algoritmo que encontrou a menor distância total percorrida no grafo.

Método de meta-aprendizado

O sistema de recomendação para a seleção de algoritmos de otimização no contexto de TSP é conceitualmente simples. Ele é semelhante a abordagens que realizam classificação de base. Sua operação acontece como segue. Inicialmente, um conjunto de meta-exemplos é apresentado a um meta-classificador. Esse conjunto contém informações sobre as descrições dos problemas de TSP e a meta-heurística de melhor desempenho para cada caso. O meta-classificador, que pode ser induzido, por exemplo, por qualquer algoritmo AM visto na Seção 1.2, gera então um meta-modelo capaz aprender a relação entre os meta-atributos de entrada e alvo. Por fim, os meta-exemplos não vistos durante a etapa de treinamento podem ter a meta-herística adequada prevista.

1.4.3.3. Resultados

Para validar o uso de meta-aprendizado para recomendação de algoritmos de otimização para variações do TSP, foram realizados experimentos em que 535 problemas foram gerados. Quatro meta-heurísticas foram avaliadas para

cada variação (AG, BT, SA e GRASP). Os meta-dados foram criados tal que para cada variação do TSP, foi indicada a meta-heurística que obteve o desempenho médio em 50 execuções. Essa meta-heurística corresponderá ao meta-atributo alvo associada a cada meta-exemplo. Quando duas ou mais meta-heurísticas apresentaram o melhor desempenho médio, os meta-atributos alvo terão mais de um valor, tornando o conjunto de dados um conjunto multirrótulo [de Carvalho e Freitas 2009].

Esse conjunto de dados foi utilizado para a indução de classificadores a serem utilizados para prever a meta-heurística mais promissora para novas variações do TSP. Foi comparada a acurácia preditiva de quatro algoritmos de indução de classificadores: *k nearest neighbors* (kNN) , Árvore de Decisão (AD), *Support Vector Machines* (SVM) e *Naïve Bayes* (NB). Diversas alternativas para lidar com dados multirrótulo foram avaliadas. Para os experimentos, o conjunto de meta-dados foi particionado utilizando *10 Cross-Validation*.

A Tabela 1.9 ilustra os resultados obtidos para três medidas de desempenho de classificãção (acurácia, medida-F e área sob a curva ROC, AUC). De acordo com esses resultados, as ADs apresentaram a melhor acurácia preditiva para a predição da meta-heurística mais promissora, com taxas de acurácia, medida-F e AUC elevadas. Os outros algoritmos também apresentaram bons resultados. Os resultados apresentados nessa tabela mostram o potencial do uso de meta-aprendizado para a previsão da melhor meta-heurística para novas variações do TSP.

Tabela 1.9. Desempenho de classificação

Medidas	kNN	AD	SVM	NB
Acurácia	0.62 ± 0.07	0.73 ± 0.07	0.56 ± 0.07	0.30 ± 0.11
Medida-F	0.65 ± 0.08	0.76 ± 0.07	0.62 ± 0.07	0.34 ± 0.11
AUC	0.91 ± 0.02	0.95 ± 0.02	0.79 ± 0.01	0.82 ± 0.04

1.4.3.4. Comentários finais

Os principais problemas encontrados na utilização de meta-aprendizado para a recomendação de técnicas de otimização para novas instâncias do TSP são a definição de meta-dados correspondentes a instâncias representativas das possíveis variações do TSP, a determinação da melhor forma de caracterizar o problema e a escolha das técnicas de otimização. Diferente dos dois outros estudos de caso, não se tem aqui um conjunto de dados a partir do qual meta-atributos são extraídos, e sim instâncias de um problema de otimização. Isso leva a novos desafios para a caracterização dos dados. A maneira encontrada para extrair meta-atributos foi transformar cada instância do problema em um grafo e utilizar características do grafo como meta-atributos. Outra particularidade deste estudo de caso diz respeito a como avaliar o desempenho de cada meta-heurística. Ao invés de medidas de acurácia e taxas de erro,

foi considerada a menor distância percorrida. Diferente da literatura para problemas de classificação e previsão, em que são conhecidas as técnicas mais adequadas para determinados domínios, para o TSP, cada trabalho, em geral, propõe e investiga técnicas diferentes, sem uma clara definição de quais devam ser preferidas. As quatro meta-heurística aqui empregadas foram selecionadas pelo entendimento de que elas são elas podem desempenhar bem em variedade de problemas de TSP.

1.5. Conclusão e novos desafios

Meta-aprendizado tem um potencial de utilização em diversos domínios de aplicação. De fato, como apresentado em [Smith-Miles 2009b], meta-aprendizado já foi adotado para seleção de algoritmos em tarefas diversas. Entretanto, ainda existe uma grande variedade de domínios de aplicação que podem ser explorados. Segundo [Smith-Miles 2009a], meta-aprendizado pode ser aplicado em qualquer domínio que compartilhe os seguintes aspectos: (1) disponibilidade de instâncias de problemas com diferentes níveis de complexidade; (2) variedade de algoritmos a escolher, com comportamento diverso; (3) métricas objetivas para avaliação de desempenho; e (4) métricas (meta-atributos) usados para descrição de problemas. Cada aspecto mencionado pode gerar trabalhos de pesquisa interessantes. Por exemplo, que meta-atributos utilizar para descrição de problemas é raramente um consenso dentro de um domínio de aplicação. Além disso, nem sempre é claro definir quando um meta-atributo é de fato relevante e reflete a complexidade dos problemas em questão.

De fato, cada domínio de aplicação possui características particulares que diferencia e adiciona dificuldades específicas o uso de meta-aprendizado. Por exemplo, em relação a disponibilidade de instâncias para a produção de meta-exemplos, em alguns domínios (como no estudo de caso de otimização de TSP apresentado na Seção 1.4.3), há pouca disponibilidade de instâncias de problemas reais. Desta forma, instâncias de problemas são geradas de forma artificial, o que pode introduzir algum viés para o meta-aprendizado. De fato, suposições subjacentes fortes podem eventualmente ser impostas durante a geração dos dados artificiais, o que pode levar a meta-exemplos pouco representativos em relação aos problemas reais. Em outros domínios (como no caso de classificação de dados de expressão gênica), as bases públicas de dados reais usadas para geração de meta-exemplos, apesar de estarem se popularizando, ainda são em número limitado. Desta forma, a produção de uma base significativamente grande de meta-exemplos não é trivial.

Em relação aos algoritmos candidatos a recomendação, em geral os estudos de meta-aprendizado são limitados a um número relativamente pequeno de algoritmos, tanto para diminuir o custo de produção de meta-exemplos, como facilitar o problema de aprendizado no nível meta. Em alguns casos, como na seleção de séries temporais, conhecimento a priori pode ser incorporado para uma escolha prévia dos algoritmos candidatos. Por exemplo, modelo

de previsão sazonal podem ser escolhidos previamente como modelos candidatos dado que as séries de interesse apresentem sazonalidade. Em outros domínios, incorporar conhecimento *a priori* pode ser mais difícil.

Em relação às métricas usadas como meta-atributos, alguns domínios possuem características em comum de forma que meta-atributos usados em um domínio podem ser aproveitados tambem em outros domínios. Por exemplo, no estudo de caso de expressão gênica, os meta-atributos foram definidos previamente por autores que investigaram meta-aprendizado para seleção de algoritmos de classificação. Por outro lado, meta-atributos podem ser bastante específicos do domínios (por exemplo, tipo de *microarray* usado na realização dos experimentos biológicos).

Como trabalhos futuros, pode-se considerar como direção de pesquisa promissora dentro de meta-aprendizado a sua aplicação para projeto de algoritmos, em vez de simplesmente seleção de algoritmos. Dentro dessa linha, pode-se incluir, por exemplo, a definição de parâmetros de algoritmos, como a seleção de parâmetros para SVMs, conforme realizado em [Soares et al. 2004]. Podemos considerar ainda nesse contexto a seleção de técnicas de pré-processamento de dados (como as técnicas de seleção de atributos) e de estratégias para combinação de algoritmos.

Outra direção de pesquisa dentro de meta-aprendizado é procurar utilizar o meta-conhecimento adquirido para propor novos algoritmos. Nesse contexto, meta-aprendizado pode fornecer informações úteis para entender a limitação dos algoritmos avaliados, que podem então ser usados pelos pesquisadores para propor novas estratégias para resolução de problemas e refinamentos de algoritmos existentes.

Referências bibliográficas

[Arinze 1994] Arinze, B. (1994). Selecting appropriate forecasting models using rule induction. *Omega-International Journal of Management Science*, 22(6):647–658.

[Asyali et al. 2006] Asyali, M. H., Colak, D., Demirkaya, O. e Inan, M. S. (2006). Gene expression profile classification: A review. *Current Bioinformatics*, 1(1):55–73.

[Atkeson et al. 1997] Atkeson, C., Moore, A. e Schaal, S. (1997). Locally weighted learning. *AI Review*, 11:11–73.

[Baranauskas 2001] Baranauskas, J. A. (2001). *Extração automática de conhecimento por múltiplos indutores*. Tese, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos - SP.

[Batista 2003] Batista, G. E. A. P. A. (2003). *Pré-processamento de dados em aprendizado de máquina supervisionado*. Tese de doutorado, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos, SP.

[Bensusan e Giraud-Carrier 2000] Bensusan, H. e Giraud-Carrier, C. (2000).

Casa batlo is in passeig de gracia or landmarking the expertise space. In *Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pp. 29–47. ECML'2000.

[Bensusan et al. 2000] Bensusan, H., Giraud-Carrier, C. e Kennedy, C. (2000). A higher-order approach to meta-learning. In *Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pp. 109–117. ECML'2000.

[Boser et al. 1992] Boser, B. E., Guyon, I. M. e Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, New York, NY, USA. ACM Press.

[Boulesteix e Strimmer 2007] Boulesteix, A.-L. e Strimmer, K. (2007). Partial least squares: A versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics*, 8(1):32–44.

[Boulesteix et al. 2008] Boulesteix, A.-L., Strobl, C., Augustin, T. e Daumer, M. (2008). Evaluating microarray-based classifiers: An overview. *Cancer Informatics*, 4:77–97.

[Box e Jenkins 1970] Box, G. E. e Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. Holden-Day.

[Brazdil et al. 2009] Brazdil, P., Giraud-Carrier, C., Soares, C. e Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Cognitive Technologies. Springer.

[Brodley 1995] Brodley, C. E. (1995). Recursive automatic bias selection for classifier construction. *Mach. Learn.*, 20(1-2):63–94.

[Burges 1998] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167.

[Carbonell et al. 1984] Carbonell, J. G., Michalski, R. S. e Mitchell, T. M. (1984). *Machine Learning: An Artificial Intelligence Approach*, capítulo An Overview of Machine Learning, pp. 3–23. Springer, Berlin, Heidelberg.

[Chapelle et al. 2006] Chapelle, O., Schölkopf, B. e Zien, A., editores (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

[Chatfield 2001] Chatfield, C. (2001). *Time Series Forecasting*. Chapman and Hall.

[Chenoweth et al. 2000] Chenoweth, T., Hubata, R. e Louis, R. S. (2000). Automatic arma identification using neural networks and the extended sample autocorrelation function: a reevaluation. *Decision Support Systems*, 29(1):21–30.

[Chu e Widjaja 1994] Chu, C.-H. e Widjaja, D. (1994). Neural network system for forecasting method selection. *Decision Support Systems*, 12(1):13–24.

[de Carvalho e Freitas 2009] de Carvalho, A. e Freitas, A. (2009). A tutorial on multi-label classification techniques. *Studies in Computational Intelligence*, 205:177–195.

[de Souza 2010] de Souza, B. F. (2010). *Meta-aprendizagem aplicada à classificação de dados de expressão gênica*. Tese de doutorado, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, São Carlos, SP.

[Demšar 2006] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

[Dietterich 1998] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

[Efron e Tibshirani 1997] Efron, B. e Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92:548–560.

[Efron e Tibshirani 1993] Efron, B. e Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York.

[Fawcett 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

[Ferri et al. 2009] Ferri, C., Hernández-Orallo, J. e Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38.

[Fix e Jr 1951] Fix, E. e Jr (1951). Discriminatory analysis: Nonparametric discrimination: Consistency properties. Relatório Técnico Project 21-49-004, Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas.

[Goldbarg e Luna 2005] Goldbarg, M. e Luna, H. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. Campus, Rio de Janeiro, 2nd edição.

[Golub et al. 1999] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D. e Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.

[Haykin 1998] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[Hechenbichler e Schliep 2006] Hechenbichler, K. e Schliep, K. (2006). Weighted k-nearest-neighbor techniques and ordinal classification. In *Discussion Paper 399, SFB 386*.

[Hyndman e Muhammad 2004] Hyndman, R. e Muhammad, A. (2004). Time series data library, http://www-personal.buseco.monash.edu.au/ hynd-

man/tsdl.

[Isaksson et al. 2008] Isaksson, A., Wallman, M., Göransson, H. e Gustafsson, M. G. (2008). Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recogn. Lett.*, 29(14):1960–1965.

[Jain e Dubes 1988] Jain, A. e Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, N.J.

[Kalousis 2002] Kalousis, A. (2002). *Algorithm Selection via Meta-Learning*. Tese de doutorado, Centre Universiteire d'Informatique, Université de Genève, Geneva, Suíça.

[Kalousis e Theoharis 1999] Kalousis, A. e Theoharis, T. (1999). Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337.

[Kanda et al. 2010] Kanda, J., Carvalho, A., Hruschka, E. e Soares, C. (2010). Using meta-learning to classify traveling salesman problems. *In Proceedings of the 2010 Brazilian Symposium on Artificial Neural Network*, pp. 73–78.

[Kanda et al. 2011] Kanda, J., Carvalho, A., Hruschka, E. e Soares, C. (2011). Selection of algorithms to solve traveling salesman problems using meta-learning. *International Journal of Hybrid Intelligent Systems*.

[Kononenko 1990] Kononenko, I. (1990). *Current trends in knowledge aquisition*, capítulo Comparison of inductive and naïve Bayesian learning approaches to automatic knowledge aquisition. IOS Press, Amsterdam.

[Lang e Hinton 1988] Lang, K. J. e Hinton, G. E. (1988). A time-delay neural network architecture for speech recognition. Relatório Técnico CMU-DS-88-152, Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[Lee e Jhee 1999] Lee, J. K. e Jhee, W. C. (1999). A two stage neural network approach for arma model identification with esacf. *Decision Support Systems*, 11:461–479.

[Lemke e Gabrys 2010] Lemke, C. e Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, 73(10-12):2006–2016.

[Lindner e Studer 1999] Lindner, G. e Studer, R. (1999). Ast: Support for algorithm selection with a cbr approach. In *PKDD '99: Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 418–423, London, UK. Springer-Verlag.

[Makridakis e Hibon 2000] Makridakis, S. e Hibon, M. (2000). The M3-competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476.

[McCulloch e Pitts 1988] McCulloch, W. S. e Pitts, W. (1988). *A logical calculus of the ideas immanent in nervous activity*, pp. 15–27. MIT Press, Cambridge, MA, USA.

[Michalski 1969] Michalski, R. S. (1969). On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3 (Switching Circuits), pp. 125–128, Bled, Yugoslavia.

[Michalski 1986] Michalski, R. S. (1986). Understanding the nature of learning: Issues and research directions. In Michalski, R. S., Carbonell, J. G. e Mitchell, T. M., editores, *Machine Learning: An Artificial Intelligence Approach: Volume II*, pp. 3–25. Kaufmann, Los Altos, CA.

[Michie et al. 1994] Michie, D., Spiegelhalter, D. J., Taylor, C. C. e Campbell, J., editores (1994). *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA.

[Mitchell 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

[Nakhaeizadeh e Schnabl 1997] Nakhaeizadeh, G. e Schnabl, A. (1997). Development of multi-criteria metrics for evaluation of data mining algorithms. In *KDD*, pp. 37–42.

[Peng et al. 2002] Peng, Y., Flach, P. A., Soares, C. e Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In *DS '02: Proceedings of the 5th International Conference on Discovery Science*, pp. 141–152, London, UK. Springer-Verlag.

[Pfahringer et al. 2000] Pfahringer, B., Bensusan, H. e Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning, ICML'2000*, pp. 743–750. Morgan Kaufmann.

[Principe et al. 2002] Principe, J., Euliano, N. e Garania, S. (2002). Principles and networks for self-organization in space-time. *Neural Networks*, 15:1069–1083.

[Prudêncio e Ludermir 2004a] Prudêncio, R. B. C. e Ludermir, T. B. (2004a). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137.

[Prudêncio e Ludermir 2004b] Prudêncio, R. B. C. e Ludermir, T. B. (2004b). Using machine learning techniques to combine forecasting methods. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, volume 3339, pp. 1122–1127.

[Quinlan 1993] Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Reynolds et al. 1995] Reynolds, S. B., Mellichamp, J. e Smith, R. (1995). Box-jenkins forecast model identification. *AI Expert*, pp. 15–28.

[Rice 1976] Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65–118.

[Salzberg 1997] Salzberg, S. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Min. Knowl. Discov.*, 1(3):317–328.

[Sanders e Manrodt 2003] Sanders, N. R. e Manrodt, K. (2003). The efficacy of using judgmental versus quantitative forecasting methods in practice. *Omega-International Journal Management Science*, 31(6):511–522.

[Santos et al. 2004] Santos, P., Ludermir, T. e Prudêncio, R. (2004). Selection of time series forecasting models based on performance information. In *4th International Conference on Hybrid Intelligent Systems*, pp. 366–371.

[Schaffer 1994] Schaffer, C. (1994). A conservation law for generalization performance. In *ICML*, pp. 259–265.

[Schena et al. 1995] Schena, M., Shalon, D., Davis, R. W. e Brown, P. O. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270(5235):467–470.

[Schrijver 2005] Schrijver, A. (2005). *Handbook of Discrete Optimization*, capítulo On the history of combinatorial optimization (till 1960), pp. 1–68. Elsevier, Amsterdam.

[Shah 1997] Shah, C. (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting*, 13:489–500.

[Simon 1984] Simon, H. A. (1984). *Machine Learning: An Artificial Intelligence Approach*, capítulo Why should machines learn?, pp. 25–37. Springer, Berlin, Heidelberg.

[Smith-Miles 2009a] Smith-Miles, K. (2009a). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41:6:1–6:25.

[Smith-Miles et al. 2010] Smith-Miles, K., van Hemert, J. I. e Lim, X. Y. (2010). Understanding tsp difficulty by learning from evolved instances. In *LION*, pp. 266–280.

[Smith-Miles 2009b] Smith-Miles, K. A. (2009b). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41:6:1–6:25.

[Soares 2004] Soares, C. (2004). *Learning Rankings of Learning Algorithms: Recommendation of Algorithms with Meta-Learning*. Tese de doutorado, Departamento de Ciência da Computação, Faculdade de Ciências da Universidade do Porto, Porto, Portugal.

[Soares e Brazdil 2000] Soares, C. e Brazdil, P. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In *PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 126–135, London, UK. Springer-Verlag.

[Soares et al. 2004] Soares, C., Brazdil, P. B. e Kuba, P. (2004). A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, 54(3):195–209.

[Sohl e Venkatachalam 1995] Sohl, J. E. e Venkatachalam, A. R. (1995). A neural network approach to forecasting model selection. *Information & Management*, 29:297–303.

[Sohn 1999] Sohn, S. Y. (1999). Meta analysis of classification algorithms for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11):1137–1144.

[Souza et al. 2008] Souza, B. F., de Carvalho, A. e Soares, C. (2008). Metalearning for gene expression data classification. In *HIS '08: Proceedings of the 2008 Eighth International Conference on Hybrid Intelligent Systems*, pp. 441–446, Washington, DC, USA. IEEE Computer Society.

[Swets 1988] Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293.

[Venkatachalan e Sohl 1999] Venkatachalan, A. R. e Sohl, J. E. (1999). An intelligent model selection and forecasting system. *Journal of Forecasting*, 18:167–180.

[Vilalta e Drissi 2002] Vilalta, R. e Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artif. Intell. Rev.*, 18(2):77–95.

[Vilalta et al. 2005] Vilalta, R., Giraud-Carrier, C., e Brazdil, P. (2005). *Data Mining and Knowledge. Discovery Handbook: A Complete Guide for Practitioners and Researchers*, capítulo Meta-Learning: Concepts and Techniques, pp. 1–17. Kluwer Academic Publishers.

[Voß2001] Voß, S. (2001). Meta-heuristics: The state of the art. In *Proceedings of the Workshop on Local Search for Planning and Scheduling-Revised Papers*, ECAI '00, pp. 1–23. Springer-Verlag.

[Wang et al. 2009] Wang, X., Smith-Miles, K. e Hyndman, R. (2009). Rule induction for forecasting method selection - meta-learning the characteristics of univariate time series. *Neurocomputing*, 72(10-12):2581–2594.

[Wold et al. 2001] Wold, S., Sjostrom, M. e Eriksson, L. (2001). Pls-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58:109–130.

[Wolpert 1996] Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390.

[Zhang 2001] Zhang, G. P. (2001). An investigation of neural networks for linear time-series forecasting. *Computers & Operation Research*, 28:1183–1202.

[Zhang et al. 2001] Zhang, G. P., Patuwo, B. E. e Hu, M. Y. (2001). A simultation study of artificial neural networks for nonlinear time-series forecasting. *Computers & Operation Research*, 28:381–396.