UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2009.1

ESTUDO DO USO DA INFORMAÇÃO MÚTUA NA SELEÇÃO DE ATRIBUTOS PARA O TREINAMENTO DE REDES NEURAIS

TRABALHO DE GRADUAÇÃO

Paulo Fagner Tenório Barros de Morais

Recife-PE, junho de 2009.

UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2009.1

ESTUDO DO USO DA INFORMAÇÃO MÚTUA NA SELEÇÃO DE ATRIBUTOS PARA O TREINAMENTO DE REDES NEURAIS

TRABALHO DE GRADUAÇÃO

Paulo Fagner Tenório Barros de Morais

Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Professor Paulo Jorge Leitão Adeodato

Recife-PE, junho de 2009.

Agradecimentos

Agradeço a Deus e a minha família pelo amor incondicional, em especial à minha mãe Maria de Fátima e seu exemplo nos estudos, na vida e na educação que nos proporcionou. Ao meu pai Paulo e meus irmãos Alysson e Samuel que sempre me apoiaram e muito contribuíram na minha formação pessoal e profissional.

À minha noiva Fabiane, que sempre esteve ao meu lado em todos os momentos deste curso e em muitas outras ocasiões da minha vida. Uma pessoa que foi peça vital para o decorrer da graduação e enfrentamento das dificuldades do curso, depositando mais confiança em minhas capacidades do que eu mesmo.

Ao professor Paulo Adeodato, pela sua motivação contagiante, envolvimento e seriedade e com que encara o assunto, e que é hoje um exemplo para mim. Agradeço por ter me apoiado e ter depositado sua confiança em mim para a concepção deste trabalho.

A todos os meus amigos do CIn, em especial aos demais membros da eterna equipe Brayner (Jolu'son, Thiago Brayner e Rafael Menelau). E a todos com quem convivi e compartilharam suas amizades no decorrer deste curso, tais como Vinícius, d'Oleron, Hugo Azevedo, Augusto Lupin, Aluísio Rodrigo (Bombeiro), Bruno Leonardo, Gleibson Rodrigo, Rodrigão, Vivi, Theógenes Ferreira, André Xico, Icamaan dentre tantos outros. Obrigado a todos.

Aos meu amigos do BNB de Recife e Salvador, pelo contínuo incentivo na conclusão deste curso.

Resumo

As redes neurais artificiais são hoje em dia uma dos exemplos mais usuais de classificador não-linear. Suas capacidades facilitaram sua divulgação e aceitação nos meios acadêmicos e comerciais. Uma das principais dificuldades enfrentadas pelos problemas de classificação é encontrar um subconjunto de atributos significativos para compor o vetor de entrada da rede. Este subconjunto estará relacionado com a redução da complexidade do problema, bem como com a redução do custo computacional do modelo construído, de forma que se busca manter o conjunto tão reduzido quanto o possível, sem que ocorra degradação do desempenho do classificador. Usualmente se objetiva manter os atributos significativos em relação às classes de saída do problema, ao passo que se busca minimizar a redundância existente na base de dados. O presente trabalho realiza um estudo sobre uma forma de seleção de atributos baseada no conceito de entropia.

Palavras-chave: Redes Neurais, Seleção de Atributos, Entropia, Informação Mútua, MIFS.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

Com uso bastante difundido nos meios acadêmicos e industriais, as redes neurais artificiais, mais comumente chamadas redes neurais, são sistemas paralelos distribuídos, compostos por unidades de processamento denominadas neurônios, cujo funcionamento baseia-se no cálculo de funções matemáticas. O conhecimento da rede é armazenado, na maioria dos casos, na forma de pesos nas conexões entre neurônios. Com uma estrutura inspirada no funcionamento do cérebro humano, sua principal faculdade é a generalização, termo oriundo da psicologia que faz referência ao fato de a rede neural gerar saídas adequadas para entradas que não se encontravam presentes durante o processo de treinamento (HAYKIN, S., 1999).

Para o processo de aprendizagem de redes neurais existem os seguintes paradigmas: i) o supervisionado, onde a cada vetor apresentado à rede, uma saída desejada é também apresentada, de forma a comparar o resultado gerado com o desejado para posterior ajuste dos parâmetros da rede; ii) o de aprendizagem por reforço, muitas vezes considerado um caso particular do paradigma supervisionado, no qual o aprendizado se dá pelo reforço das ações corretas realizadas pela rede, sem que seja calculado um valor de *score* para cada saída gerada e iii) o não-supervisionado, onde o aprendizado se dá pela existência de regularidade e redundância nos padrões apresentados à rede, uma vez que o padrão desejado não é conhecido de antemão (BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B., 2007).

Como regra geral, o processo de aprendizagem faz uso de uma massa de dados, composta por registros, ou instâncias, sendo estes, por sua vez, formados por atributos, ou variáveis, que descrevem as características dos registros da amostra. Desta forma, tem-se que a cada registro está associado um vetor de atributos e, nos casos de paradigma supervisionado de aprendizagem, um valor de saída desejado.

Em problemas de classificação, um dos principais desafios da etapa de modelagem e treinamento da rede é a escolha dos atributos que comporão o seu vetor de entrada. Este processo é de fundamental importância para a melhoria da capacidade de generalizar de um classificador (BATTITI, R., 1992). Na prática, o conjunto ótimo de atributos é geralmente desconhecido, de forma que se torna comum a presença de atributos irrelevantes ou redundantes na massa de dados. Assim, busca-se manter o número de atributos tão pequeno quanto o possível para reduzir o custo computacional de treinamento de um classificador, bem como sua complexidade (ESTÉVEZ, P. A. *et al.*, 2009).

Objetivos

Verifica-se, portanto, a necessidade de seleção de atributos para a modelagem de uma rede neural de forma a evitar a utilização de atributos irrelevantes ou redundantes, bem como prover melhoria na capacidade de generalização do classificador, além da redução do custo e complexidade computacionais. O objetivo do presente trabalho é realizar um estudo sobre a utilização da Informação Mútua como parâmetro para o processo de seleção de atributos para o treinamento e modelagem de uma rede MLP (*Multilayer Perceptron*). Segundo Battiti (1992), no contexto da Teoria da Informação, o problema consiste em encontrar um subconjunto de atributos *S* com *k* atributos, contido no conjunto inicial *F* com *n* atributos, que minimiza a incerteza do conjunto das classes de saída *C* dado o subconjunto *S*, e maximiza a informação mútua entre *C* e *S*.

O presente trabalho descreve a implementação e análise do algoritmo proposto por Battiti, o MIFS (*Mutual Information for Feature Selection*), bem como o algoritmo MIFS-U (*Mutual Information for Feature Selection Under Uniform Information Distribution*), proposto como uma melhoria sobre o original (KWAK, N.; CHOI, C.H., 2002). O desempenho foi exaustivamente testado sobre

os atributos de duas bases de dados públicas envolvendo problemas de classificação, e comparado com outra técnica de seleção, bem como com as bases originais completas. O desempenho das técnicas de seleção de atributos será mensurado: i) a partir dos custos computacionais associados ao processo de seleção de atributos e ao treinamento da rede construída com estes atributos de entrada e ii) pelos desempenhos das redes neurais modeladas e treinadas com os atributos selecionados, desempenhos estes medidos pelos métodos de geração da curva ROC (FAWCETT, 2003) e pelo teste estatístico de Kolmogorov-Smirnov (KS) (CONNOVER, W. J., 1999).

Organização

O presente trabalho contempla ainda, além desta seção introdutória, os seguintes capítulos:

- Capítulo 2: Trata sobre as redes neurais artificiais, conceitos, suas aplicações e o modelo MLP, utilizado no trabalho.

- Capítulo 3: Implicações da Multidimensionalidade, Técnicas Empregadas na Seleção de Atributos e breve revisão da literatura sobre o tema.

- Capítulo 4: Trata sobre os conceitos e métodos de estimação da informação mútua. Apresenta uma descrição do algoritmo MIFS, o processo de implementação e suas variações.

- Capítulo 5: Apresenta os resultados gerados a partir dos testes realizados sobre as bases de dados.

- Capítulo 6: Resume o trabalho realizado destacando as principais contribuições, resultados atingidos e principais desafios para trabalhos futuros.

2. Redes Neurais

Como já exposto na seção anterior, Redes Neurais são sistemas paralelos distribuídos, compostos por unidades de processamento, denominadas neurônios, dispostas em camadas e cujo funcionamento baseia-se no cálculo de funções matemáticas. As unidades, entre as camadas, são interconectadas e, na maioria dos modelos, estão associadas a pesos, os quais armazenam o conhecimento adquirido na forma de valores numéricos. Trata-se então de uma abordagem alternativa à computação algorítmica, com uma forma de computação inspirada no funcionamento do cérebro humano.

Devido a sua capacidade de generalização, as redes neurais são amplamente utilizadas na resolução de problemas do mundo real envolvendo grandes massas de dados. A natureza destes problemas é em geral de: i) classificação, *i.e.* atribuição a um padrão desconhecido uma entre várias classes de conhecidas; ii) categorização, *i.e.* o agrupamento em categorias bem definidas, obtidas a partir da redundância existente entre os registros apresentados à rede; iii) aproximação, em geral aproximação de funções e iv) previsão, *i.e.* prever situações futuras a partir dos dados atuais (BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B., 2007).

No decorrer deste capítulo será apresenta um histórico das redes neurais e uma explanação sobre o modelo utilizado no presente trabalho, o MLP (*Multilayer Perceprton*), e seu algoritmo de aprendizagem mais comumente utilizado: o *Backpropagation*.

2.1 Breve Histórico das Redes Neurais

A inspiração para o funcionamento das redes neurais é baseada no funcionamento do cérebro humano, área aprofunda pelo estudo pioneiro de

Ramón Y Cajál (HAYKIN, S., 1999), que introduziu a idéia dos neurônios tais como são conhecidos hoje. Os neurônios são entre 5 e 6 vezes mais lentos que as portas lógicas de silício (HAYKIN, S., 1999), porém trabalhando de forma maciçamente paralela, o cérebro compensa esse aparente retardo na velocidade de funcionamento das células nervosas.

McCulloch e Pitts (MCCULLOCH, W. S.; PITTS, W., 1943) foram os primeiros a formular um modelo matemático simples para um neurônio. McCulloch um psiquiatra, Pitts, matemático, associaram-se um ano antes da publicação do trabalho conjunto. O trabalho é em essência uma unificação de estudos da neurofisiologia com a lógica matemática, que propõe conexões entre os neurônios ajustadas corretamente e operando de forma síncrona para a computação de funções. Neste modelo, a saída de um dado neurônio assume valor 1 caso o campo local induzido (*i.e.* uma combinação linear entre seus valores de entrada acrescida de um valor de bias) é não-negativo e assume valor 0 na situação contrária. Este modelo utiliza uma função degrau para a condição de disparo, tal função é chamada de função de ativação sendo também referida como função restritiva, pois restringe a faixa de saída para um intervalo menor ou conjunto reduzido de valores. O trabalho de McCulloch e Pitts foi um marco para a área, pois mostrou que era possível fazer computação de uma forma inspirada no funcionamento do cérebro.

Como marco posterior há o livro de Hebb em 1949 (HAYKIN, S., 1999), intitulado *The Organization of Behavior*, que propõe que a conectividade no cérebro é continuamente modificada a partir do aprendizado de novas tarefas funcionais criando agrupamentos neurais. Seu trabalho, entretanto, foi pouco influente na comunidade de engenharia, exercendo grande impacto entre psicólogos (HAYKIN, S., 1999). O trabalho lança o *postulado da aprendizagem* que afirma que o reforço em uma conexão entre dois neurônios deve ser reforçado em razão de uma constante ativação de um dos dois neurônios a partir da conexão entre eles.

O trabalho de Hebb, em grau de relevância para a área, é sucedido pelo trabalho de Frank Rosenblat em1958 (BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B., 2007) que cria o modelo Perceptron, baseado no modelo de McCulloch-Pitts. Seu modelo é baseado na variação da intensidade entre as conexões inter-neurônios, ou seja, sinapses ajustáveis, como uma forma de prover treinamento à rede para classificar certos tipos de padrões. Seu modelo, entretanto, era capaz de resolver apenas problemas linearmente separáveis, não sendo possível lidar com não-linearidades.

O fato de o Perceptron não resolver problemas não-linearmente separáveis aliado à observação de Minsky e Papert em 1969 (HAYKIN, S., 1999), afirmando que o modelo não garantia a convergência para mais de uma camada, resultou em um adormecimento nas pesquisas em redes neurais durante os anos 1970 (BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B., 2007), época onde alguns poucos pesquisadores continuaram seus estudos na área. Os efeitos pessimistas em relação à capacidade das redes neurais causados pelo livro de Minsky e Papert só foram revertidos com o advento do algoritmo *Backpropagation* (RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J., 1986) que mostrou que o modelo Perceptron era capaz de resolver problemas não-lineramente separáveis. O *Backpropagation* utiliza múltiplas camadas, baseia-se no método do gradiente descendente sendo bastante utilizado no modelo *Multilayer Perceptron*, descrito a seguir.

2.2 Perceptron de Múltiplas Camadas

No presente trabalho foi utilizada o Perceptron de Múltiplas Camadas, ou rede MLP (*Multilayer Perceptron*), que adota o paradigma supervisionado de aprendizagem, caracterizado pela existência de um “supervisor” externo, cujo papel central é fornecer as entradas à rede e comparar as saídas geradas com um resultado esperado, para posterior ajuste de parâmetros, a partir dos erros

gerados. A minimização do erro decorre de maneira incremental até a convergência, definida por um critério de parada pré-estabelecido. Assim, o vetor contendo os dados a serem processados é apresentado à rede a partir da camada de entrada, gerando estímulos que são propagados adiante nas demais camadas. À exceção da última camada (camada de saída), as saídas de cada neurônio de uma dada camada comporão as entradas da camada subseqüente. Desta forma, por cada conexão entre neurônios, transcorrem estímulos que são ponderados por seus respectivos pesos ω_j, determinando, a partir de uma função de ativação, os estímulos que chegam aos neurônios conectados a sua saída. Seguindo este procedimento, tem-se então uma resposta final na camada de saída, correspondente ao vetor apresentado.

O aprendizado na rede MLP transcorre em duas fases, sendo seu algoritmo de treinamento mais popular o *backpropagation* (HAYKIN, S. 1999: 183). Na primeira fase, denominada *forward*, é gerada a saída da rede correspondente ao padrão apresentado. Na segunda fase, denominada *backward*, a saída desejada e a saída de fato fornecida pela rede são comparadas, e um erro é calculado. Este erro é então utilizado como parâmetro para a atualização dos pesos da camada de saída, sendo em seguida retro-propagado para a camada anterior a partir de sua multiplicação pelo peso das conexões entre as camadas. Desta maneira, cada camada escondida (*i.e.* toda camada localizada entre a camada de entrada e a camada de saída) recebe sua parcela de contribuição no erro gerado na camada de saída e atualiza seus próprios pesos.

2.2.1 Arquitetura do Modelo MLP

As redes MLP adotam múltiplas camadas, alimentadas para frente (*feedforward*), sendo a primeira a camada de entrada, sucedida pelas camadas escondidas (ou intermediárias) e, por fim, uma camada de saída. A Figura 1

ilustra um modelo MLP com duas camadas escondidas. A arquitetura da rede pode ser mais bem descrita pelo número de camadas, a função de ativação e o número de neurônios por camada.

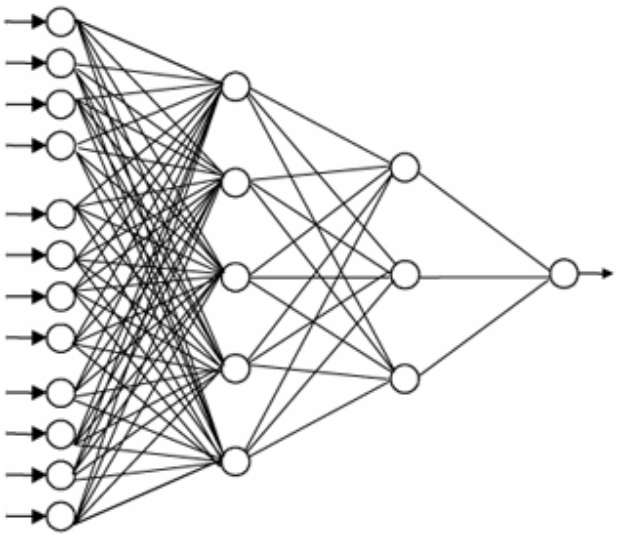

Figura 1 - Modelo esquemático de uma rede MLP com duas camadas intermediárias.

O Número de Camadas

A escolha do número de camadas é dada pela natureza do problema em questão e em cada camada ocorre uma fração do processamento como um todo, onde o espaço é dividido em regiões de decisão. Desta forma a rede é capaz de trabalhar com problemas que não sejam linearmente separáveis.

Papel importante cabe às camadas intermediárias, que criam uma codificação interna para os padrões apresentados na entrada. Um número suficientemente grande de camadas é possível formar representações para qualquer conjunto de entrada (BRAGA, A. P.; CARVALHO, A. P. L. F.;

LUDERMIR, T. B., 2007). Entretanto a grande maioria dos problemas raramente necessita de mais de uma camada intermediária.

Funções de Ativação

A rede emprega em cada unidade de processamento uma função de ativação. A função de ativação utilizada nas camadas intermediárias deve ser uma função não-linear, para que seja incorporado o tratamento de não-linearidade à rede. Caso as funções sejam puramente lineares, o *Perceptron* de Múltiplas Camadas terá funcionamento equivalente a uma única camada, visto que sucessivas transformações lineares equivalem a uma única transformação linear (BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B., 2007). As funções de ativação mais utilizadas no modelo MLP são funções sigmóides, tais como a função logística e a função tangente hiperbólica, que devem estar presentes em pelo menos uma das camadas escondidas. As funções sigmóides são contínuas, diferenciáveis em qualquer ponto, monotônicas estritamente crescentes, possuindo certa similaridade com funções lineares, o que torna essa classe de funções ideal para o MLP. As funções sigmóides se aproximam assintoticamente de seus valores de saturação. A Figura 2 ilustra um exemplo de função sigmóide, a função logística binária, com a imagem definida no intervalo [0,1]. A função logística é definida por:

$$f(x) = 1 / (1 + \exp(-kx))$$, sendo k uma constante positiva.

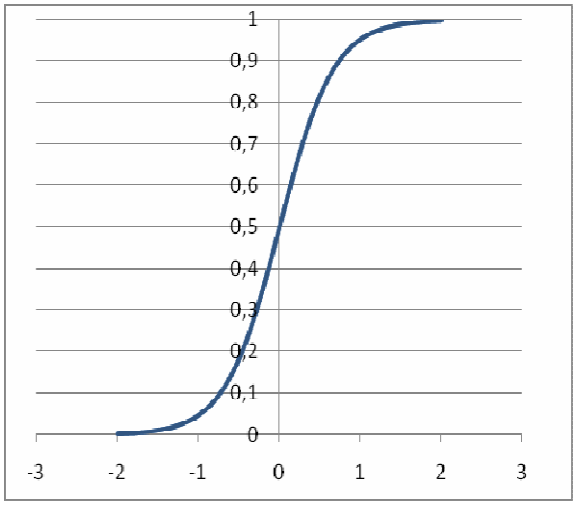

Figura 2 - Gráfico da função logística binária para o domínio [-2, 2] e *k* igual a 3.

A constante *k* é utilizada como um fator de inclinação da função. O valor de *K* também atua como um controle automático de ganho (BEALE, R.; JACKSON, T., 1990), visto que para valores de entrada pequenos a inclinação é bastante íngreme e as saídas aumentam rapidamente em relação ao domínio e; para valores grandes de entrada a inclinação varia em menor intensidade. O termo *k* possibilita então um balanceamento para as saídas desta função, dada a amplitude do domínio.

A função logística também pode ser bipolar, caso este em que a imagem da função é definida no intervalo [-1,1]. Define-se como mostrado abaixo:

$f(x) = (1 - \exp(-kx)) / (1 + \exp(-kx))$, sendo k uma constante positiva.

No presente trabalho, as redes utilizadas utilizam a função logística binária como função de ativação.

Definição do número de neurônios por camada

Não há uma regra geral bem explicitada na literatura sobre a definição do número de neurônios para a resolução de um determinado problema. Esta

modelagem deverá ser definida em função da complexidade do problema, de maneira que quanto maior a complexidade, um maior número de neurônios é exigido, atentando sempre para que a rede não perca sua capacidade de generalização caso um número excessivo de neurônios escondidos seja utilizado. Um ponto relevante neste caso é que a complexidade do problema a ser tratado não é conhecida de antemão, e o modelo a ser utilizado deve ter complexidade proporcional ao problema.

Desta forma, nos testes realizados no presente trabalho, a quantidade de neurônios na camada escondida foi definida empiricamente, a partir da realização de sucessivos testes. Para a camada de entrada foram utilizados tantos nós quanto atributos selecionados da base e, por fim, dois neurônios na camada de saída, pois se tratam de problemas binários de classificação.

O algoritmo de Aprendizagem

No presente trabalho foi utilizado o algoritmo padrão *Backpropagation*, brevemente descrito anteriormente nesta seção e esquematicamente detalhado neste tópico. A seguir a descrição do algoritmo:

Passo 1: Os pesos são inicializados com valores pequenos e aleatórios, e algumas vezes retirados de uma distribuição pré-definida. Caso a implementação faça uso de um *threshold* este também deverá ser inicializado neste passo. Convém ressaltar que utilizar um *threshold* é equivalente a adicionar um bias ao vetor de entrada com valor 1.

Passo 2: São apresentados o vetor de entrada (X_j, j = 0, 1, 2,..., n) e a saída desejada (t). Cada unidade de entrada transmite o sinal recebido para a primeira camada intermediária. Os passos 2 e 3 são repetidos enquanto a condição de parada não for alcançada.

Passo 2.1: Em cada camada intermediária uma combinação linear é realizada em cada unidade de processamento. Esta combinação é dada pela soma ponderada de todos os valores recebidos dos nós da camada anterior multiplicados pelos pesos de suas respectivas conexões. Assim, cada unidade da camada intermediária (Z_i, i = 0, 1, 2,..., p) calcula $z_j = \sum_{i \epsilon j} \omega_{ij} x_i$, e em seguida processa a função de ativação com o parâmetro z_j, ou seja $z\_saida_j = f(z_j)$. Este último sinal é então propagado à camada seguinte.

Passo 2.2: Para cada unidade de saída (O_k, k = 0, 1, 2,..., m) é realizado um procedimento semelhante ao passo anterior. Calcula-se $o\_entrada_k = \sum_{i \epsilon p} \omega_{jk} x_j$ e em seguida calcula-se a saída da função de ativação (o_k) usando o parâmetro $o\_entrada_k$.

Passo 3: Fase de retroprogação do erro. Cada unidade de saída calcula o erro correspondente da seguinte forma: $\delta_k = (t_k - o_k)(f'(o\_entrada_k))$. Partindo da camada de saída, procede-se com o ajuste dos pesos a partir do cálculo:

$$\omega_{ij}(t + 1) = \omega_{ij}(t) + \eta \, \delta_k \, o_k,$$

Onde o termo $\omega_{ij}(t + 1)$ corresponde ao novo peso após a iteração e η é a taxa de aprendizagem, um valor pertencente ao conjunto dos Reais e usualmente no intervalo (0, 1]. O termo δ de erro varia de acordo com a camada: para a camada de saída é calculado como já descrito e para as camadas intermediárias é dado por $\delta_j = f'(o\_entrada_k) \sum_{k \, \epsilon \, m} (\delta_k \, \omega_{jk})$.

O passo 4 testa a condição de parada.

2.3 Implicações da Multidimensionalidade

É intuitivo perceber que quanto maior a quantidade de neurônios na camada intermediária no modelo MLP maior a complexidade do modelo

construído. Dado que a quantidade de neurônios de entrada e a quantidade de neurônios de saída são obtidas em função do problema, a complexidade da rede estará determinada pelo número de nós escondidos.

Independentemente da modelagem da rede, um aspecto relevante a ser analisado é a dimensão do vetor de características (vetor de entrada ou vetor de atributos), que contém informação sobre a complexidade do problema. Este aspecto será abordado no capítulo seguinte, onde são tratados de temas como a Maldição da Dimensionalidade e VC *Dimension*.

2.4 Considerações Finais

Diante do exposto, o presente trabalho utiliza o modelo MLP, dada suas características e capacidades. Sua faculdade de generalização, ampla aceitação nos meios comerciais e acadêmicos e a capacidade de resolução de problemas não-linearmente separáveis foram fatores decisivos para a escolha deste classificador para a execução dos testes neste trabalho.

Deve-se ter em mente também, que a redução da complexidade é um dos objetivos para a melhoria do desempenho e tempo de treinamento da rede. Quanto maior o vetor de características, maior será a complexidade do modelo e, por conseguinte, maior o tempo de treinamento e as possibilidades de falhas na classificação. Buscando-se a redução do vetor de características, contribui-se para a melhoria do modelo, sem perdas na sua capacidade de generalização.

3. Redução da Dimensionalidade e Técnicas de Seleção de Características

Como exposto na seção anterior, as redes neurais são sistemas computacionais capazes de resolver problemas não-lineares e aprender com dados do passado para que o conhecimento armazenado seja a base da generalização de respostas para situações futuras. Os modelos, entretanto, dada a natureza do problema, exigem complexidade equivalente para o aprendizado efetivo, o que nem sempre é algo mensurável. Este capítulo explanará sobre alguns motivos para que se mantenha a dimensionalidade do problema tão pequena quanto o possível e as implicações positivas desta preocupação no treinamento e aprendizado das redes neurais.

Dentre outros motivos, há dois principais para a redução da dimensionalidade: (i) redução do custo computacional, o que torna o classificador mais rápido e com menor consumo de memória e (ii) melhoria do desempenho de classificação. Importante lembrar que estes objetivos devem ser alcançados sem que se deteriore a capacidade de um classificador.

Dois conceitos relacionados com a redução da dimensionalidade, a Maldição da Dimensionalidade e VC *Dimension*, são discutidos nas subseções seguintes.

3.2 A Maldição da Dimensionalidade e o Fenômeno do Pico

A maldição da dimensionalidade foi introduzida por Bellman (BELLMAN, R., 1961) em seu livro *Adaptative Control Processes: A Guided Tour*, e faz referência ao fato de que há um aumento exponencial da complexidade de um problema devido ao aumento de sua dimensionalidade. Dado um vetor de

entrada *X* de dimensão *m* e uma partição do espaço de entradas com *n* registros (ou padrões), a densidade da amostragem é proporcional a $n^{1/m}$, de forma que um aumento de dimensão fará a densidade cair de maneira exponencial. Assim, tomando o exemplo de uma função complexa *F* a ser aproximada, é necessário tomar amostras de dados densas para aproximá-la de maneira satisfatória.

Verifica-se que o número necessário de instâncias de treinamento para que o classificador tenha um desempenho satisfatório é função exponencial da dimensionalidade do vetor de características e que na prática, segundo Jain, Duin e Mao (2000), a adição de novas características pode degradar o desempenho de um classificador se o número de padrões é relativamente pequeno em relação à dimensão do vetor de características. Tipicamente a adição de características diminui o erro de classificação e logo em seguida, com a contínua adição de características, o erro é elevado, gerando então um pico. Este fenômeno é conhecido como *Fenômeno do Pico*, sendo conseqüência direta da maldição da dimensionalidade.

3.3 VC *Dimension*

VC *Dimension* (Vapnik-Chervonenkis *Dimension*), ou Dimensão VC, no contexto da teoria da aprendizagem computacional, é uma medida de capacidade de um sistema de classificação, definida como a cardinalidade do maior conjunto de pontos que o algoritmo pode realizar uma operação conhecida como *particionamento*. O particionamento é realizado quando dado um conjunto de instâncias *S*, para cada dicotomia (*i.e.* uma dada função de classificação binária) deste conjunto existe alguma hipótese num conjunto de hipóteses *H* consistente com esta dicotomia (MITCHELL, T. M., 1997). Em outras palavras, VC *Dimension* mede a complexidade de um conjunto de hipóteses *H* não por sua cardinalidade, mas pelo número de diferentes instâncias de um dado conjunto *X* que podem ser descriminadas usando *H*. Ou ainda: o número

máximo de exemplos de treinamento que podem ser aprendidos pela máquina sem erro para todas as rotulações possíveis das funções de classificação (HAYKIN, S., 1999).

A Figura 3 ilustra um exemplo intuitivo de dimensão VC. Seja *H* o conjunto de todas as possíveis superfícies de decisão linear no plano, ou seja, *H* é o espaço de hipóteses correspondente a um perceptron simples e o conjunto de instâncias *X* é composto por elementos das classes '+' e '-'. Para que a condição de *particionamento* seja alcançada, para cada dicotomia de *X* deve haver alguma hipótese em *H* consistente com a dicotomia; ou seja, todo conjunto de instâncias deve ser separado por reta pertencente a *H*. Desta maneira, verifica-se que para as situações *a*, *b* e *c* na Figura 3 a situação é satisfeita. Em *d* não há nenhuma reta capaz de capaz de estabelecer a dicotomia. Assim, ao menos que os pontos sejam colineares, a VC *dimension* para *H*, *VC(H) = 3.*

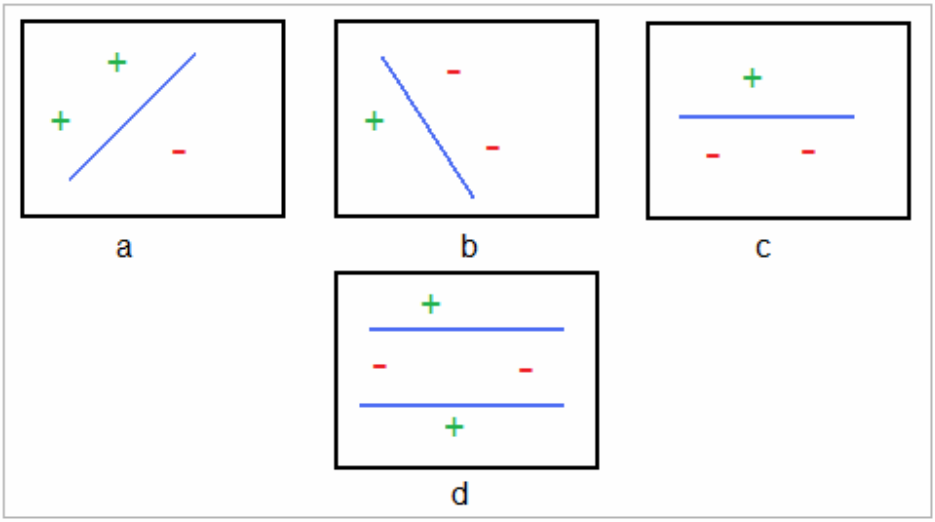

Figura 3 – Exemplo de VC *dimension* no espaço 2-D para superfície única de decisão linear. Decisões possíveis em *a*, *b* e *c.* Em *d* não é possível. A VC *dimension* é 3.

A quantidade de elementos necessários para que um classificador aprenda uma classe de exemplos é proporcional à dimensão VC daquela classe, fato que determina a importância da estimação da dimensão VC.

Para o Perceptron de Múltiplas Camadas, Koiran e Sontag (1997) mostraram em seu trabalho que uma rede utilizando funções de ativação contínuas, como a função logística, possui dimensão VC proporcional a w^2 onde w é o número de parâmetros livres da rede (pesos). A dimensão VC para o MLP possui complexidade computacional de ordem $O(w^2)$.

3.4 A Seleção de Características

Visando redução de custo computacional e melhoria no desempenho de classificadores, um vasto campo de pesquisa se formou em torno do estudo de técnicas de seleção de características. Desta forma, o estudo da dimensão VC de um classificador e a análise da maldição da dimensionalidade mostram o importância da redução da dimensionalidade do problema.

Seleção de características é a etapa de pré-processamento de dados responsável pelo processo de escolha de um subconjunto de atributos extraído de um conjunto original, visando uma melhor descrição do conjunto de dados (LIU, H. *et al.*, 2008). Visa além da redução da dimensionalidade, a remoção de atributos redundantes e atributos irrelevantes, a redução da quantidade de dados necessária para o aprendizado do algoritmo e tornar o modelo a ser construído mais compreensível.

3.4.1 Abordagens

Dentro deste campo da aprendizagem de máquina, há dois propósitos relacionados com a redução da dimensionalidade: extração de características e

seleção de características (JAIN, A. K.; DUIN, R. P.; MAO, J., 2000). De acordo com Jain, Duin e Mao (2000), extração de características é o conjunto de métodos utilizados na criação de novos atributos a partir de combinações e transformações entre os existentes no conjunto original. Seleção de características, abordagem estudada no presente trabalho, é a seleção de subconjuntos ótimos a partir do conjunto original. A seleção de características também pode ocorrer a partir de um *ranking* de atributos da base.

Para o processo de seleção de atributos (características) existem, em geral, duas distintas metodologias: o uso de *filters* e o uso de *wrappers* (KOHAVI, J.; PFLEGER, K., 1994). *Filters* são métodos de seleção de características empregados na etapa de pré-processamento de dados, que usam determinadas técnicas de avaliação do conjunto original, buscando identificar relevância e redundância, de maneira completamente independente do algoritmo de aprendizagem (Liu *et al.*, 2008) (ESTÉVEZ *et al.*, 2009), onde a seleção de atributos ocorre sem nenhuma intervenção do classificador. Desta maneira, os métodos que usam esta abordagem são bastante populares em razão de seu reduzido custo computacional. Um exemplo de método que usa está abordagem é o CFS (*Correlation-based Feature Selection*), disponível no software WEKA – *Waikato Environment for Knowledge Analysis* (WEKA, 2008) e descrito em Hall (1999). Alguns *filters* geram subconjuntos de atributos, enquanto outros geram um *ranking* de atributos por ordem de relevância.

Os *wrappers* utilizam o desempenho do classificador para avaliar a relevância de um conjunto de atributos (ESTÉVEZ *et al.*, 2009), o que os fazem gerar melhores resultados que os *filters*, porém com um custo computacional mais elevado, uma vez que esta categoria necessita re-treinar o classificador para cada diferente combinação de atributos. Devido ao fato desta classe utilizar um algoritmo específico de seleção, seu conjunto de atributos gerado é menos genérico, o que poderá gerar respostas diferentes de classificação a partir da mudança do classificador.

O presente trabalho realiza um estudo sobre seleção de atributos usando uma determinada família de *filters*.

3.4.2 Objetivos da Seleção de Características

Uma explanação sobre divisão de um conjunto de atributos é encontrada em Yu e Liu (2004) que realiza uma decomposição da seguinte maneira: i) atributos irrelevantes; ii) atributos redundantes; iii) atributos fracamente relevantes, porém não redundantes e iv) atributos altamente relevantes. Para Yu e Liu o conjunto ótimo deve conter todos os atributos altamente relevantes e um subconjunto dos fracamente relevantes, porém não redundantes. É interessante observar que outras abordagens podem ser adotadas. Dada a natureza do problema, é interessante a eliminação de atributos que apesar de bastantes relevantes são também altamente redundantes em relação a outros atributos presentes na base de dados.

Como regra geral, busca-se manter atributos relevantes e relativamente relevantes, ao passo em que se eliminam os irrelevantes e/ou altamente redundantes.

3.4.3 Principais Técnicas Empregadas

Uma técnica bastante empregada na extração de atributos é o método PCA (*Principal Component Analysis*) que transforma atributos em outros. O ponto negativo desta técnica é a manutenção dos dados, que requer novo processamento caso novos dados sejam adicionados. Além do mais, a técnica é sensível a transformações feitas sobre a base de dados.

Para a seleção de atributos, métodos baseados em correlação estatística são bastante usados, porém não garantem que sejam capturadas correlações

não-lineares entre atributos. Não há certeza da existência de dados puramente lineares nos problemas do mundo real.

Uma grande contribuição foi feita com o uso de árvores de decisão (KWAK, N.; CHOI, C. H., 2002), a exemplo da seleção híbrida de atributos proposta por Setiono e Lui (1997) que exclui as características de entrada uma por uma e re-treina a rede repetidamente. O método é eficiente na escolha dos atributos, porém requer um tempo relativamente alto para re-treinar a rede para cada nova combinação de atributos. Um outro método baseado em árvores faz uso do conceito de informação mútua, que será a seguir introduzido. Este último método, proposto por Agrawal, Imielinski e Swami (1993) consome uma grande quantidade de memória, pois mapeia todos os possíveis pares de entrada e saída do problema.

O presente trabalho foca atenção no algoritmo proposto por Battiti (1992) e no algoritmo proposto por Kwak e Choi (2002) como uma melhoria sobre o original de Battiti. O algoritmo de Battiti, o MIFS é baseado na informação mútua, que leva a vantagem de trabalhar com relações não-lineares entre variáveis. O algoritmo será apresentado no capítulo seguinte, bem como a teoria envolvida em sua concepção.

4. Entropia, Informação Mútua e o MIFS

Neste capítulo são discutidos os algoritmos MIFS e o MIFS-U e os conceitos chaves relacionados a eles, tais como entropia e informação mútua. O capítulo trata ainda das implementações realizadas para a estimação da informação mútua a partir dos dados, a exemplo do algoritmo de Fraser (FRASER, A. M; SWINNEY, H. L., 1986) e o método de otimização de histogramas proposto por Shimazaki e Shinomoto (2007).

4.1 Entropia e Informação Mútua

A teoria da informação introduzida por Shannon (1948) em seu clássico trabalho *A Mathematical Theory of Communication* apresenta alguns conceitos matemáticos aplicados à transmissão de sinais. Um dos temas centrais do trabalho descreve os conceitos de Entropia e Incerteza.

A entropia de Shannon é uma medida da informação capaz de quantificar a incerteza de variáveis aleatórias bem como a quantidade de informação compartilhada entre elas. A entropia é, portanto, uma medida do nível de incerteza em uma distribuição. A entropia de uma dada classe *C* é definida por:

$$H(C) = -\sum_{c=1}^{N_c} P(c) \log P(c)$$

Onde a probabilidade para as valorações possíveis de *C* é dada por P(c), c = 1,..., N.

Duas considerações são relevantes acerca da entropia de uma variável aleatória:

1. Quando a entropia é 0, a incerteza é mínima. Assim a distribuição deve apresentar um único elemento com probabilidade de ocorrência 1 e todos os demais com probabilidade 0. Não há surpresa no resultado, que já é sabido de antemão.
2. A entropia em uma dada distribuição será máxima quando todos os elementos possuírem igual probabilidade de ocorrência. Assim o "nível de surpresa" na ocorrência de um evento é máximo.

A entropia condicional é a incerteza restante em uma distribuição após se conhecer outra. Matematicamente para o caso discreto:

$$H(C|F) = -\sum_{f=1}^{N_f} P(\boldsymbol{f}) \left(\sum_{c=1}^{N_c} P(c|\boldsymbol{f}) \log P(c|\boldsymbol{f}) \right)$$

Onde *F* é a classe conhecida de antemão com N_f possíveis valorações. A entropia conjunta é definida por:

$$H(X,Y) = -\sum_{y \in Y} \sum_{x \in X} p(x,y) \log p(x,y)$$

A partir destes conceitos, pode-se definir a Informação Mútua *I* entre duas classes:

$$I(C;F) = H(C) - H(C|F)$$

Esta expressão pode ser reduzida para:

$$I(C;F) = I(F;C) = \sum_{c,f} P(c,\boldsymbol{f}) \log \frac{P(c,\boldsymbol{f})}{P(c)P(\boldsymbol{f})}$$

A informação mútua é definida desta forma como a quantidade de incerteza que é reduzida em uma dada classe a partir do conhecimento provido

por outra (BATTITI, R., 1992). A Figura 4 seguinte ilustra o conceito de Informação Mútua bem como sua relação com a entropia anteriormente definida.

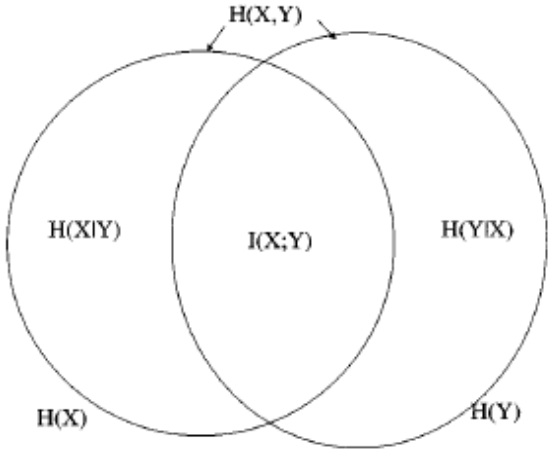

Figura 4 - A informação mútua *I* é dada pela interseção das entropias das classes *X* e *Y*.

A informação mútua entre duas variáveis será baixa quando elas forem fracamente correlacionadas e, alta, na situação contrária. Quando a informação mútua é 0, significa que as duas variáveis são independentes, sem nenhuma correlação entre si. Kwak e Choi (2002) apresentam outras relações entre a informação mútua e a entropia:

$I(X; Y) = I(Y; X)$ e, portanto $I(X, Y) = H(Y) – H(Y \mid X)$

$I(X; Y) = H(X) + H(Y) – H(X, Y)$

$I(X; X) = H(X)$ uma vez que nenhuma informação adicional sobre *X* é obtida dada a própria classe *X.*

Para variáveis contínuas, a entropia pode ser representada pela integral:

$$H = -\int_{-\infty}^{\infty} p(x) \log p(x)\, dx$$

Onde *p(x)* é a função de densidade de probabilidade. Para uma distribuição *n* dimensional tem-se:

$$H = -\int \cdots \int p(x_1, \ldots, x_n) \log p(x_1, \ldots, x_n)\, dx_1 \cdots dx_n$$

E, portanto, para classes duas classes:

$$H(x,y) = -\iint p(x,y) \log p(x,y)\, dx\, dy$$

De forma que o cálculo da entropia para as variáveis contínuas depende do sistema de coordenadas adotado. Portanto transformações lineares realizadas sobre as funções de densidade resultarão em novos valores de entropia (SHANNON, C. E., 1948) (BATTITI, R., 1992).

A informação mútua para os sistemas contínuos é definida:

$$I(X;Y) = \int \int p(x,y) \log \frac{p(x,y)}{p(x)p(y)}\, dxdy$$

E caso um dos atributos seja discreto, a exemplo de uma classe de saída, o somatório substituirá a respectiva integral:

$$I(C;F) = I(F;C) = \sum_c \int P(c,\boldsymbol{f}) \log \frac{P(c,\boldsymbol{f})}{P(c)P(\boldsymbol{f})} d\boldsymbol{f}$$

4.2 O Algoritmo MIFS

Uma vez que em alguns problemas pode-se chegar a um subconjunto de atributos, contido no conjunto inicial, capaz de resolver o problema de classificação de maneira satisfatória e com baixa ambigüidade, Battiti (1992) define o problema de seleção de características (FRn-k) da seguinte forma:

(FRn-k): “Dado um conjunto inicial de *n* atributos, encontrar um subconjunto com *k* < *n* atributos que é ‘maximamente informativo’ sobre a classe.”

E no contexto da Teoria da Informação, Battiti reformulou o problema da seguinte maneira:

(FRn-k): Dado um conjunto inicial *F* com *n* atributos, encontrar um subconjunto *S* contido em *F* com *k* atributos que miniminiza *H(C|S) i.e.* que maximiza a informação mútua *I(C; S).*

Para a resolução do problema proposto, caso a quantidade de atributos seja alta, torna-se computacionalmente inviável o cálculo da informação mútua para ordens elevadas de *C* e *S*. Caso se opte por extrair todos os subconjuntos e treiná-los diretamente na rede neural, a quantidade de tempo necessária seria ainda maior, visto que o total de subconjuntos gerados é uma combinação de *n* elementos tomados *k* a *k* com *k* variando entre 1 e *n*.

Desta forma, Battiti propôs em seu algoritmo duas aproximações para a resolução do problema FRn-k: i) a informação mútua entre o vetor de variáveis é aproximada pela informação mútua entre pares de variáveis e ii) a análise de todos os possíveis subconjuntos dá lugar a um algoritmo guloso que seleciona as variáveis que maximizam uma dada expressão. Assim, ao invés de se computar *I*(*F; C*), a informação mútua entre um conjunto de atributos e o conjunto das classes de saída, o MIFS computa apenas *I*(*f, C*) e *I*(*f, f’*), onde *f* e *f’* são dois atributos da base de dados.

A seguir é descrito o algoritmo MIFS:

1) (Inicialização) Defina $F \leftarrow$ “conjunto inicial de n atributos” e $S \leftarrow$ “conjunto inicialmente vazio”.

2) (Cálculo da informação mútua em relação à classe de saída) Para cada atributo f pertencente a F computar $I(C; f_i)$.

3) (Escolha do primeiro atributo) Encontrar um atributo f_i que maximiza $I(C; f_i)$; Fazer $F \leftarrow F \setminus \{ f_i \}$; $S \leftarrow \{ f_i \}$

4) (Seleção gulosa) Repetir até que $|S| = k$:

a) (Cálculo da informação mútua entre variáveis) para todo par de variáveis (f_i, f_s) com f_i pertencente a F e f_s pertencente a S computar $I(f_i; f_s)$ caso ainda não computado.

b) (Escolha do próximo atributo) escolher o atributo f_i que maximiza $I(C; f_i) - \beta \Sigma_{s \text{ em } S} I(f_i; f_s)$; Fazer $F \leftarrow F \setminus \{ f_i \}$; $S \leftarrow S \cup \{ f_i \}$.

5) Retornar o conjunto S contendo os atributos selecionados.

O termo β regula a relevância dos atributos já selecionados no processo de escolha de atributos. Caso seja 0 apenas a informação mútua entre cada atributo e a classe de saída é considerada. Com o incremento de β aumenta-se o nível de influência da redundância entre os atributos já selecionados, como fator de escolha do próximo atributo. Battiti afirma que na pratica a escolha de β no intervalo $0.5 \leq \beta \leq 1$ é apropriado para a maioria dos problemas de classificação.

Percebe-se, portanto, que o algoritmo MIFS usa a informação mútua entre pares de variáveis como uma heurística para a aproximação do cálculo da informação mútua entre vetores de ordem mais elevada e as classes de saída. A

seguir será descrito o algoritmo MIFS-U (*Mutual Information for Feature Selection Under Uniform Information Distribution*) (KWAK, N.; CHOI, C.H., 2002).

4.3 O MIFS-U

O algoritmo MIFS-U foi proposto por Kwak e Choi (2002) como uma melhoria no MIFS original de Battiti. Tal como o original, a eliminação de atributos deve ser motivada pela eliminação de atributos irrelevantes em relação à saída, bem como atributos redundantes dentre os presentes na massa de dados. É importante observar que o passo 4 do algoritmo de Battiti é uma heurística que visa aproximar a seguinte seleção ideal:

> 4) (Seleção gulosa) Repetir até que número desejado de variáveis seja alcançado:
>
> a) (Cálculo da informação mútua entre variáveis) para todo f_i pertencente a F, computar $I(C; f_i, S)$.
>
> b) (Escolha do próximo atributo) escolher o atributo f que maximiza $I(C; f_i, S)$; Fazer $F \leftarrow F \setminus \{ f_i \}$; $S \leftarrow S \cup \{ f_i \}$.

Como a estimação de probabilidades é uma tarefa difícil, é comum o uso de histogramas para a estimação da informação mútua a partir dos dados. Porém, vale observar que a estimação de informação mútua para vetores de dimensões elevadas é uma tarefa computacionalmente impraticável em razão do elevado consumo de memória. Desta forma, para a seleção de k atributos, com a classe de saída possuindo K_c valorações, o j-ésimo atributo é dividido em P_j partições para a geração do histograma, sendo assim necessárias $K_c \times \prod_j P_j$ (para j = 1,…, k) células de memória para computar $I(C; f, S)$ (KWAK, N.; CHOI, C. H., 2002).

Observando novamente o passo 4 do MIFS, verifica-se que a escolha do próximo atributo é direcionado pela maximização da expressão $I(C; f_i) - \beta\Sigma_{s \text{ em } S} I(f_i; f_s)$ onde f_i é o atributo candidato e f_s é um dado atributo pertencente ao conjunto dos atributos já selecionados *S.* Como já citado, caso β assuma o valor 0, apenas $I(C; f_i)$ será levado em consideração no processo de seleção. À medida que β cresce, a informação mútua entre atributos começa a influenciar o procedimento de seleção e o grau de redundância tende a cair. Para um valor de β elevado, o algoritmo tende a considerar apenas a relação entre atributos, minimizando assim a redundância, porém desprezando a relação entre os atributos e as classes de saída, o que não é algo aceitável.

A Figura 5 mostra a relação entre dois atributos e a classe de saída. A área hachurada representa $I(C; f_i, f_s)$ a informação mútua entre um par de atributos e a classe de saída. É fácil perceber que para $\beta = 1$, a expressão $I(C; f_i) - \beta\Sigma_{s \text{ em } S} I(f_i; f_s)$ corresponde à diferença entre a área 3 e a área 1. O MIFS-U, como descrito a seguir, busca de fato aproximar a soma das áreas 2, 3 e 4, ou seja, estimar $I(C; f_i, f_s)$.

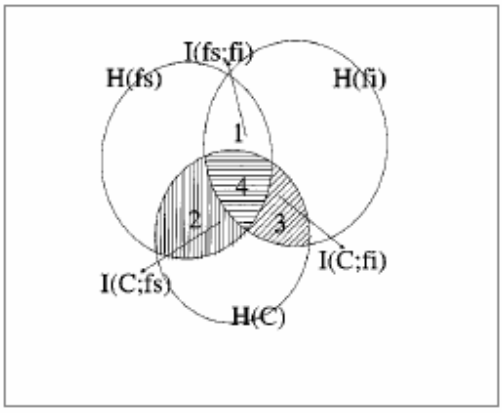

Figura 5 - Relação entre dois atributos e a classe de saída.

4.3.1 Descrição do Algoritmo

O valor $I(C; f_i, f_s)$, a partir da análise da Figura 5, pode ser reescrito:

$I(C; f_i, f_s) = I(C; f_s) + I(C; f_i \mid f_s)$, onde $I(C; f_i \mid f_s)$ representa a informação mútua remanescente entre a classe de saída C e o atributo f_i dado o atributo f_s, correspondendo à área 3 na Figura 5. O problema agora consiste em aproximar $I(C; f_i \mid f_s)$. Assumindo uma distribuição de probabilidade uniforme, os autores do algoritmo chegam a $I(C; f_i \mid f_s) = I(C; f_i) - (I(C; f_s) / H(f_s)) * I(f_s; f_i)$, que é então utilizado na expressão inicial de $I(C; f_i, f_s)$. E como o termo $I(C; f_s)$ é comum para qualquer atributo candidato, apenas o termo $I(C; f_i \mid f_s)$ é utilizado no passo 4 do algoritmo. Compartilhando os 3 primeiros passos com o MIFS, o passo 4 do MIFS-U será então dado por:

4) (Seleção gulosa) Repetir até que o número desejado seja alcançado:

a) (Cálculo da entropia) Para todo f_s pertencente a S, computar $H(f_s)$ caso ainda não computada.

b) (Cálculo da informação mútua entre variáveis) para todo par de variáveis (f_i, f_s) com f_i pertencente a F e f_s pertencente a S computar $I(f_i; f_s)$ caso ainda não computado.

c) (Escolha do próximo atributo) escolher o atributo f_i que maximiza $I(C; f_i) - \beta \Sigma_{s \text{ em } S} (I(C; f_s) / H(f_s)) * I(f_i; f_s)$. Fazer $F \leftarrow F \setminus \{ f_i \}$; $S \leftarrow S \cup \{ f_i \}$.

4.4 Estimação da Informação Mútua a partir dos dados

No presente trabalho, a informação mútua entre os pares de atributos foi estimada a partir da implementação do algoritmo de Fraser (FRASER, A. M; SWINNEY, H. L., 1986). A entropia de cada atributo e a informação mútua entre um dado atributo e a classe de saída foram estimadas com o uso de

histogramas planos, onde a quantidade de partições, para cada atributo, é obtida por um processo de otimização proposto por Shimazaki e Shinomoto (2007). Estes procedimentos são brevemente descritos a seguir.

4.4.1 O Algoritmo de Fraser

Utilizado exclusivamente para a estimação da informação mútua sobre dados contínuos, o algoritmo de Fraser é baseado na invariância da informação mútua sobre o efeito de transformações sobre os dados, realizando uma discretização adaptativa (*i.e.* geração de intervalos de largura variável com um suficiente número de eventos em cada um deles) (FRASER, A. M; SWINNEY, H.L., 1986). O algoritmo é recursivo, de maneira que cada chamada avança mais fundo em uma partição onde a distribuição conjunta de probabilidade possui estrutura mais detalhada. O passo básico da recursão é chamado quando não se encontra um volume expressivo de eventos para uma estimação mais acurada da probabilidade conjunta. A complexidade do algoritmo é de ordem *N log N*.

A Figura 6 ilustra um plano bi-dimensional particionado segundo o algoritmo de Fraser. Caso uma subestrutura seja encontrada ela será subdividida e gerará 2^d novas subestruturas, onde *d* é a dimensão do espaço. As subestrutura R2(k1, 0), R2(k1, 1), R2(k1, 2) e R2(k1, 3) foram geradas a partir da subestrutura que as contém, denominada R1(k1).

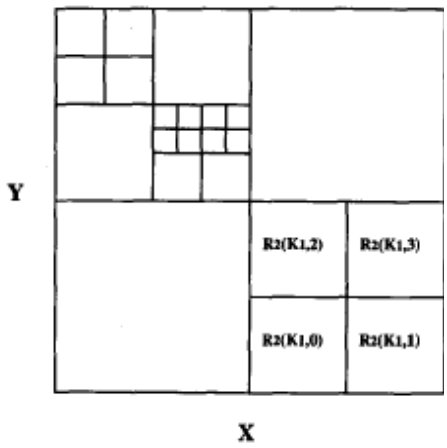

Figura 6 - Particionamento do plano pelo algoritmo de Fraser.

A seguir é descrito um resumo esquemático do algoritmo para o plano bi-dimensional (2 variáveis):

1) O algoritmo tem início com a transformação dos dados Reais em Inteiros, a partir da ordenação crescente dos dados em cada atributo. O conjunto transformado dos dados do atributo será composto pelos valores inteiros correspondentes ao índice no *array* dos atributos ordenados. Ou seja: cada atributo real dará lugar à sua respectiva posição no conjunto ordenado.

2) O plano é conceitualmente particionado com duas retas perpendiculares a cada eixo, onde cada reta divide o plano em duas partições com igual quantidade de elementos. São geradas desta forma 4 subestruturas iniciais. Uma forma de facilitar a contagem de elementos em cada subestrutura é utilizar uma quantidade inicial de registros igual a uma potência de 2 (dois), pois assim basta que se divida cada eixo de uma partição em duas metades.

3) A condição de verificação de subestrutura é feita com o teste do χ-quadrado.

4) A informação mútua ***I(X, Y)*** é computada:

$$I(X,Y) = \frac{F(R_0(K_0))}{N_0} - \log_2(N_0)$$

Onde ***F()*** é uma função recursiva, chamada cada vez que se encontra uma nova subestrutura. Caso uma subestrutura exista segundo o passo 3, o passo recursivo é dado por:

$$F(R_m(K_m)) = N_{Km}\log_2(4) + \sum_{j=0}^{3} F(R_{m+1}(K_m, j))$$

Caso não seja encontrada subestrutura, ***F()*** assume seu passo básico e finaliza a recursão na subestrutura:

$$F(R_m(K_m)) = N_{Km}\log_2(N_{Km})$$

O método que gerou a aproximação de ***I(X, Y)*** informada no passo 4 é detalhado em Fraser e Swinney (1986).

4.4.2 Seleção da largura da partição em um histograma

O método de seleção da largura da partição de um histograma está abaixo descrito. A teoria envolvida na aproximação alcançada pela função de custo está descrita em (SHIMAZAKI, H.; SHINOMOTO, S., 2007). O método proposto é baseado na estimação do MISE (*Mean Integrated Squared Error*), uma medida da aderência do histograma a uma distribuição desconhecida.

1) Dividir a faixa de dados em *N* partições com largura Δ. Contar o número de eventos em cada partição.

2) Calcular a média k e a variância v do número de eventos em relação a *N:*

$$k \equiv \frac{1}{N}\sum_{i=1}^{N} k_i \qquad v \equiv \frac{1}{N}\sum_{i=1}^{N}(k_i - k)^2.$$

3) Computar a função de custo *C*:

$$C(\Delta) = \frac{2k - v}{\Delta^2}.$$

4) Repetir os passos de 1 a 3 alterando o valor Δ, a partir da variação de *N.* Encontrar Δ que minimiza a função *C(Δ)*.

5 Resultados dos Experimentos

Para a avaliação do desempenho dos algoritmos de seleção de atributos estudados, foram utilizadas duas bases de dados: (i) a base da Mortalidade infantil até um ano de idade no Brasil no ano 2000, para a granularidade município e (ii) Base *Sonar Mines vs. Rocks* disponível no repositório do UCI. A seguir serão descritas bases de dados e os resultados obtidos nos experimentos.

5.1 A Base da Mortalidade Infantil 2000

A base foi montada a partir dos registros obtidos do *website* do IBGE, da base DATASUS e de dados extraídos do Atlas do Desenvolvimento Humano no Brasil (PNUD, 2003), software de consultas que congrega dados do IPEA, IBGE dentre outras instituições de pesquisa. Os dados se referem aos anos 2000 e 2002. A escolha dos atributos foi motivada por estudos que apontam para a existência de uma correlação entre as características sócio-econômicas e os níveis de MI nas sociedades (BOING, A. F.; BOING, A. C., 2008), sobretudo pelo estudo coordenado por Celso Simões do IBGE (IBGE, 1999), que aponta para a exclusão social de parcelas significativas da população do Brasil e a manutenção das desigualdades sociais e regionais como um forte obstáculo a reduções mais significativas dos níveis de mortalidade infantil.

5.1.1 Atributos e o Problema

Os atributos encontram-se subdivididos conforme os tipos de dados relativos: (i) à infra-estrutura (*e.g.* percentual de domicílios urbanos com serviços

de coleta de lixo, de pessoas que vivem em domicílios com água encanada, com rede de esgoto e saneamento básico); (ii) aos indicadores sócio-econômicos (*e.g.* tempo de escolaridade de pessoas com idade superior a 25 anos, percentual de analfabetos com idade superior a 25 anos, renda per capita do município, distribuição da população (zona rural x área urbana)); (iii) à existência e/ou acesso aos serviços de saúde (*e.g.* número de estabelecimentos de saúde geral e pública, números de leitos para internação, número de médicos residentes) e (iv) aos indicadores de natalidade (e.g. taxa de fecundidade total). Esses dados foram relacionados com a mortalidade até 1 ano de idade. Foram ainda criados 3 atributos a partir dos já existentes visando embutir conhecimento sobre a base. A base final possui um total de 33 atributos e 5.507 registros que se referem aos municípios existentes no Brasil de acordo com o Censo 2000 do IBGE.

A classe de saída representa o nível da taxa de mortalidade no município. Os municípios brasileiros estão então divididos em duas categorias em termos de taxas de mortalidade infantil tendo o limiar para a segregação em duas classes igual a 25 (vinte e cinco) mortes. Desta forma, os registros com taxas acima de 25 mortes antes de completar um ano de nascido por mil nascidos vivos, são agrupados na classe 1 (altas taxas de mortalidade) e, para registros com taxas menores ou iguais a esse limiar, é associada a classe 0 (baixas taxas de mortalidade). A Tabela 1 explicita a distribuição dos municípios por classes de mortalidade infantil.

Tabela 1 - Distribuição da classe de saída em relação ao total de registros.

Classe	**Quantidade de Municípios**	**Percentual em relação ao total de municípios**
Baixas Taxas – 0	2229	40,48
Altas Taxas – 1	3278	59,52

5.2 Metodologia e Resultados Obtidos

Para a análise de desempenho dos algoritmos MIFS e MIFS-U, a base de dados foi submetida ao programa implementado. A implementação gerou então um *ranking* de relevância das variáveis para os dois algoritmos bem como para a informação mútua em relação à classe de saída (ganho de informação univariado). Este *ranking* é explicitado na Tabela 2 para cada algoritmo.

Tabela 2 - *Ranking* de atributos gerados pelo MIFS segundo critério especifico de seleção.

Critério	***Ranking* de atributos**
MIFS, $\beta = 0$ (Ganho de Informação Univariado)	21, 18, 22, 31, 15, 16, 23, 14, 28, 24, 19, 3, 20, 17, 25, 13, 2, 26, 4, 6, 32, 29, 30, 33, 12, 8, 27, 5, 11, 34, 7, 9, 10
MIFS, $\beta = 0.25$	21, 18, 17, 25, 26, 2, 34, 7, 32, 10, 13, 27, 5, 8, 15, 30, 4, 3, 6, 24, 28, 20, 9, 31, 14, 22, 19, 29, 16, 23, 12, 11, 33
MIFS, $\beta = 0.5$	21, 20, 26, 25, 8, 10, 7, 32, 2, 5, 34, 27, 17, 4, 30, 13, 15, 6, 3, 24, 9, 28, 14, 22, 31, 19, 29, 16, 23, 18, 12, 11, 33
MIFS, $\beta = 0.75$	21, 25, 26, 2, 10, 34, 7, 27, 32, 5, 29, 4, 17, 8, 13, 20, 6, 15, 3, 9, 24, 28, 14, 22, 31, 19, 30, 16, 23, 18, 12, 11, 33
MIFS, $\beta = 1$	21, 2, 26, 10, 7, 34, 5, 27, 32, 30, 4, 25, 8, 17, 13, 20, 6, 15, 3, 9, 24, 28, 14, 23, 31, 19, 29, 16, 22, 12, 11, 18, 33

Os resultados gerados pelo MIFS-U encontram-se descritos na Tabela 3.

Tabela 3 - *Ranking* de atributos gerados pelo MIFS-U segundo critério especifico de seleção.

Critério	***Ranking* de atributos**
MIFS-U, β = 0.25	21, 18, 22, 31, 15, 16, 28, 23, 24, 17, 3, 14, 20, 19, 25, 13, 2, 26, 4, 32, 6, 34, 10, 5, 30, 27, 29, 8, 9, 7, 12, 11, 33
MIFS-U, β = 0.5	21, 18, 31, 22, 15, 28, 16, 17, 24, 20, 25, 3, 13, 14, 26, 23, 2, 32, 4, 10, 34, 7, 6, 5, 27, 8, 30, 19, 29, 9, 12, 11, 33
MIFS-U, β = 0.75	21, 18, 31, 22, 15, 28, 17, 25, 24, 20, 13, 26, 3, 2, 32, 10, 4, 34, 7, 5, 27, 6, 8, 29, 30, 19, 23, 16, 9, 14, 12, 11, 33
MIFS-U, β = 1	21, 18, 15, 22, 28, 17, 25, 31, 13, 26, 2, 32, 10, 34, 7, 4, 5, 27, 6, 8, 30, 20, 29, 3, 24, 19, 23, 9, 16, 14, 12, 11, 33

Depois de gerado o *ranking* das variáveis segundo os critérios definidos, para cada critério de seleção, os atributos foram subdivididos em 3 partições. A primeira partição contém o primeiro quarto do conjunto de atributos "ranqueados", a segunda a primeira metade e a terceira com os três primeiros quartos de atributos segundo o a ordenação gerada pelo algoritmo e seus respectivos critérios (termo β). Por exemplo, o primeiro quarto de atributos para o algoritmo MIFS-U, critério β = 0.5 é o subconjunto {21, 18, 31, 22, 15, 28, 16, 17}.

5.2.1 Treinamento da Rede

O desempenho para cada subconjunto gerado foi mensurado a partir da acurácia da rede MLP treinada com o algoritmo de aprendizagem *Backpropagation* disponível no *software* WEKA versão 3.5 (WEKA, 2008). Partindo da premissa empírica que a "melhor rede" é "a menor rede" que resolve

o problema de maneira mais satisfatória, a partir de alguns experimentos iniciais, chegou-se à configuração utilizada no MLP. Procederam-se diversos experimentos, com diversas combinações de taxa de aprendizagem, termo momentum e variações da quantidade de neurônios da camada intermediária. A configuração adotada da rede foi: 3 (três) neurônios na camada intermediária, taxa de aprendizagem de 0.1, termo momentum 0.2 e teve como critério de parada o ponto de menor erro quadrático médio (MSE) sobre o conjunto de validação e conjunto de testes de igual tamanho com 1.377 registros, ou seja, 25% da massa total. Os experimentos foram realizados com 100 épocas de treinamento.

5.2.2 Avaliação de Desempenho

Para análise das classificações realizadas pela rede MLP, foram utilizados os avaliadores: análise do gráfico do teste estatístico de Kolmogorov-Smirnov (KS); e geração e extração das métricas da curva ROC (*Receiving Operating Characteristic*). O conjunto de testes possui 1.377 registros, sendo 555 da classe 0 (municípios de baixas taxas de mortalidade infantil) não-alvo, e 822 da classe 1(municípios de altas taxas de mortalidade infantil) alvo.

O Teste de Kolmogorov-Smirnov (KS) e a Curva ROC

O teste estatístico de Kolmogorov-Smirnov visa verificar os quão aderentes a uma distribuição estão os dados (CONNOVER, W. J., 1999). Mede o nível de separação de duas classes a partir de suas funções de distribuição acumulada (FDAs).

A Figura 7 mostra um gráfico do teste de Kolmogorov-Smirnov para um experimento realizado com o primeiro quarto de atributos selecionados com o

algoritmo MIFS-U para β=0.25. O valor máximo de KS para este experimento foi de 0.8253 e a área sob a curva KS foi 0.469.

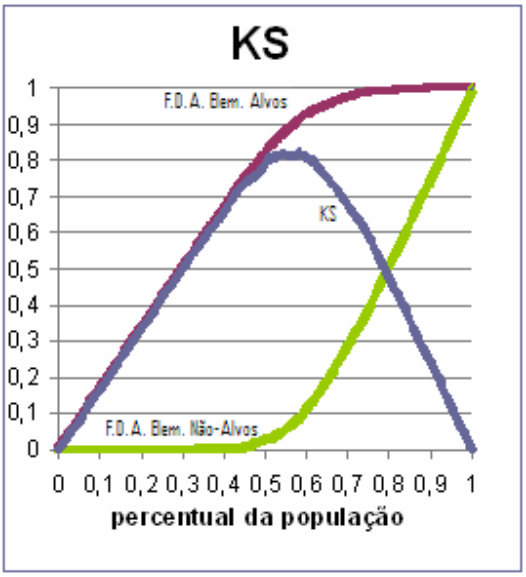

Figura 7 - Gráfico do teste de Kolmogorov-Smirnov.

A curva ROC originou-se da área das telecomunicações, com uso bastante difundido como estimador da avaliação de desempenho. A curva ROC relaciona, em um eixo bi-dimensional, a taxa de verdadeiros positivos com a taxa de falsos positivos gerados simulando a classificação produzida por todos os escores gerados pelo sistema, sendo o escore um valor que mensura a chance de um elemento pertencer a uma determinada classe. As taxas são ordenadas ao longo dos eixos, a partir da ordem decrescente de seus respectivos escores (FAWCETT, 2003).

A Figura 8 exibe a curva ROC para um experimento realizado com o primeiro quarto de atributos selecionados com o algoritmo MIFS-U para β=0.25. Para este experimento a área sob a curva é 0.969 e a distância mínima ao ponto (0; 1) do gráfico é 0,123.

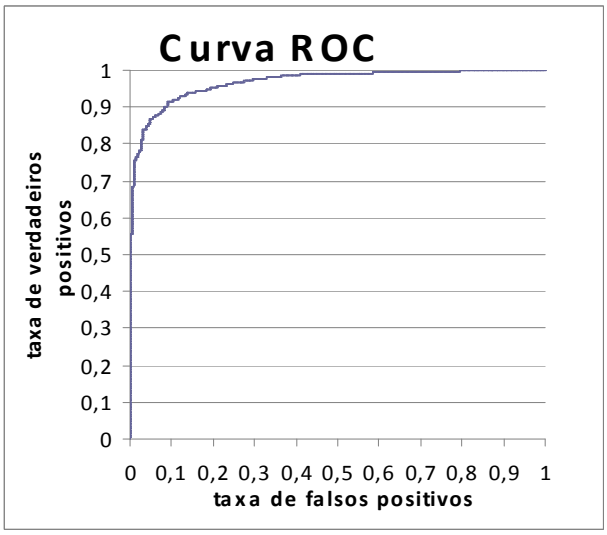

Figura 8 - Grafico da curva ROC.

Resultado da classificação da rede MLP

Cada rede foi treinada e testada 5 (cinco) vezes para cada subconjunto do vetor de entrada inicial, com os valores dos pesos fixados aleatoriamente a cada treinamento. Os resultados obtidos estão descritos nas Tabelas 4 e 5. Os pontos de corte para cada resultado foram o escore no ponto de menor distância em relação ao ponto ótimo (0; 1) da curva ROC e o escore no ponto de KS máximo.

Tabela 4 - Resultado de classificações para o MIFS.

Algoritmo / Critério	**Subconjunto**	**Acurácia (corte ROC)**	**Desvio-padrão**	**Acurácia (corte KS)**	**Desvio-padrão**
MIFS, β=0 (Ganho de Informação Univariado)	Primeiro quarto	90,49%	0,16	90,48%	0,16
	Primeira metade	90,87%	0,21	90,844%	0,20
	3 primeiros	90,62%	0,20	90,35%	0,32

	quartos				
MIFS, β=0.25	Primeiro quarto	89,87%	1,21	89,79%	1,17
	Primeira metade	87,75%	0,85	87,63%	1,04
	3 primeiros quartos	88,62%	1,08	88,59%	1,12
MIFS, β=0.5	Primeiro quarto	90,96%	0,1	90,90%	1,16
	Primeira metade	86,93%	3,39	87,01%	3,21
	3 primeiros quartos	88,21%	0,91	88,04%	1,13
MIFS, β=1	Primeiro quarto	89,65%	1,12	89,65%	1,12
	Primeira metade	87,24%	3,59	87,37%	3,32
	3 primeiros quartos	89,44%	0,29	89,31%	0,3

Tabela 5 - Resultado de classificações para o MIFS-U.

Algoritmo / Critério	**Subconjunto**	**Acurácia (corte ROC)**	**Desvio-padrão**	**Acurácia (corte KS)**	**Desvio-padrão**
MIFS-U, β=0.25	Primeiro quarto	91,04%	0,15	90,75%	0,37
	Primeira metade	90,80%	0,19	90,68%	0,31
	3 primeiros	89,24%	0,61	89,14%	0,58

	quartos				
MIFS-U, β=0.5	Primeiro quarto	90,81%	0,16	90,72%	0,27
	Primeira metade	90,89%	0,20	90,76%	0,25
	3 primeiros quartos	89,19%	0,61	88,72	0,95
MIFS-U, β=1	Primeiro quarto	90,55%	0,10	90,46%	0,12
	Primeira metade	89,32%	0,64	89,16	0,59
	3 primeiros quartos	89,24%	0,64	89,03%	0,56
CFS	12 atributos	91,10%	0,13	91,05%	0,19
Base Inteira (Todos os atributos)	-	90,8%	0,2	90,77%	0,24

5.3 A Base *Mines vs. Rocks*

A base de dados pública *Mines vs. Rocks* contém 111 atributos referentes à classe “Mina” e 97 atributos à classe “Rocha”. Cada classe foi obtida, respectivamente, a partir da aplicação de sinais de sonar em um metal cilíndrico e sobre rochas em vários ângulos e sobre várias condições (GORMAN, R. P.; SEJNOWSKI, T. J., 1988). A base de dados está atualmente disponível no repositório UCI, possuindo 60 atributos numéricos.

Seguindo procedimentos semelhantes aos realizados para a base da Mortalidade Infantil, foram obtidos os resultados de classificação descritos nas Tabelas 6 e 7.

Tabela 6 - Resultado de classificações para o MIFS. Base *Mines vs. Rocks*.

Algoritmo / Critério	**Subconjunto**	**Acurácia (corte ROC)**	**Desvio-padrão**	**Acurácia (corte KS)**	**Desvio-padrão**
MIFS, β=0 (Ganho de Informação Univariado)	Primeiro quarto	73,07%	0	73,07%	0
	Primeira metade	79,48%	1,81	79,48%	1,81
	3 primeiros quartos	76,92%	1,56	76,92%	1,56
MIFS, β=0.5	Primeiro quarto	82,68%	1,57	82,68%	1,57
	Primeira metade	79,48%	0,90	79,48%	0,90
	3 primeiros quartos	78,84%	0	78,84%	0
CFS	20 atributos	81,40%	0,90	81,40%	0,90
Base Inteira (Todos os atributos)	-	83,16%	2,09	84,12%	2,09

Tabela 7 - Resultado de classificações para o MIFS-U. Base *Mines vs. Rocks*.

Algoritmo / Critério	**Subconjunto**	**Acurácia (corte ROC)**	**Desvio-padrão**	**Acurácia (corte KS)**	**Desvio-padrão**
MIFS-U,	Primeiro	85,25%	0,9	85,25%	0,9

β=0.5	quarto				
	Primeira metade	79,48%	0,9	79,48%	0,9
	3 primeiros quartos	77,56%	0,9	77,56%	0,9

5.4 Considerações sobre os Resultados

A partir dos resultados obtidos nas classificações, pode-se verificar a eficiência das implementações realizadas. Para a base da Mortalidade Infantil pode-se observar que a redução para 8 atributos (primeiro quarto) foi capaz de manter atributos significativos ao passo que eliminou redundantes. A acurácia de de 90,80% para a base com todos os atributos chegou a ser superada pelo subconjunto do primeiro quarto para algumas situações. Como ponto negativo, verifica-se que valores elevados de β tendem a lançar atributos significativos para as últimas posições do *ranking* de atributos. Isso se deve ao fato de que a redundância foi priorizada em detrimento da relevância em relação à saída à medida que aumenta a cardinalidade do conjunto dos atributos já selecionados. Este ponto negativo é abordado em outras variações do MIFS, como o NMIFS (ESTÉVEZ, P. A. *et al.*, 2009) que busca normalizar cada informação mútua entre pares de atributos no segundo termo da subtração do passo 4 no algoritmo MIFS ($I(C; f_i) - \beta\Sigma_{s \text{ em } S} I(f_i; f_s)$). Observa-se que o segundo termo ($\beta\Sigma_{s \text{ em } S} I(f_i; f_s)$) cresce à medida que o a quantidade de atributos aumenta. Assim, após normalizar a informação mútua em cada atributo ($I(f_i; f_s)/min\{H(f_i),H(f_s)\}$) o NMIFS extrai a informação mútua média entre os pares de atributos de forma a tornar o segundo termo da expressão citada comparável ao primeiro. O NMIFS também foi testado sobre as bases, porém seus resultados foram bastante próximos aos alcançados pelo ganho de informação univariado, assim seus resultados foram desconsiderados neste trabalho.

O tempo de execução do algoritmo é dado pelo tempo da execução do algoritmo de estimação da informação mútua entre atributos, que no caso do algoritmo de Fraser é função de sua complexidade, *N log N*, onde *N* é o total de registros. Assim, para a base da mortalidade infantil, com 5507 registros, cada estimação da informação mútua entre um par de atributos consumiu um tempo de 0,92s, em um computador AMD Semprom, 1,60 G*Hz* freqüência de clock e núcleo único de processamento. Vale ressaltar que este tempo é o dobro do tempo necessário para computar 4096 registros (a menor potência de 2 mais próxima de 5507), pois para cada base, cada par de atributos teve sua estimação da informação mútua realizada duas vezes: para os 4096 primeiros registros e para os 4096 últimos. A média da informação mútua entre as duas execuções foi utilizada como a informação mútua final entre os dois atributos.

6 Conclusão

O presente trabalho apresentou um estudo sobre o uso da informação mútua na seleção de atributos para o treinamento de redes neurais. Foi estudada a teoria por trás dos métodos baseados no MIFS de Battiti e experimentalmente observadas algumas de suas deficiências. Devido ao fato dos algoritmos baseados no MIFS, a exemplo do MIFS-U e NMIFS, utilizarem uma heurística semelhante à de Battiti, as relações não-lineares existentes entre grupos de atributos influenciando conjuntamente certa classe de saída não podem ser captadas por estes métodos. Eles, todavia, são bons na avaliação da redundância e ponderação em relação à classe de saída. Os métodos utilizados na estimação da informação mútua provaram ser eficientes na estimação destes valores.

Em que pese o fato dos algoritmos analisados gerarem um *ranking* por ordem de relevância e minimização de redundância, ao invés de um subconjunto único, verifica-se ser fácil utilizar uma função de custo que eliminaria alguns atributos mantendo a ordem do *ranking* e gerando assim um subconjunto otimizado de acordo com uma dada heurística. Esta função estaria lançada dentro de um algoritmo baseado no MIFS representando um novo passo dentro do procedimento. Esta abordagem poderá ser utilizada em trabalhos futuros.

Com relação à estimação da informação mútua, outras abordagens de naturezas diversas de histogramas ou histogramas adaptativos, como Parzen-Window, poderão ser empregadas para o computo direto da informação mútua entre grupos de atributos em relação às classes de saída. Ao invés do uso de uma heurística de estimação, como ocorre nos métodos estudados, a estimação direta poderá ser utilizada com métodos que requeiram menor consumo de memória, apesar de maior custo computacional.

Referências

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A., **Database mining: A performance perspective**, IEEE Trans. Knowledge Data Eng., vol. 5, Dez. 1993.

BATTITI, R. **Using Mutual Information for Selecting Features in Supervised Neural Net Learning**, IEEE Trans. Neural Networks, vol. 5, nº 4, pp. 537-550, Jul. 1994.

BEALE, R.; JACKSON, T., **Neural Computing: An Introduction.** Bristol, J W Arrowsmith Ltd., 1990.

BELLMAN, R. **Adaptative Control Process: A Guided Tour.** Princeton, NJ: Princeton University Press, 1961.

BOING, A. F.; BOING, A. C., CADERNOS DA SAÚDE PÚBLICA. Rio de Janeiro: Fundação Oswaldo Cruz, 24 fev. 2008. Disponível em: <http://www.ensp.fiocruz.br/csp/bib.html>. Acesso em: 12 jan. 2009.

BRAGA, Antônio de P.; CARVALHO, André Ponce de Leon F. de; LUDERMIR, Teresa Bernarda. **Redes Neurais Artificiais: Teoria e Aplicações**. 2. ed. Rio de Janeiro: LTC, 2007. 283 p.

CONOVER, W. J. **Practical Nonparametric Statistics**. John Wiley & Sons, 1999.

ESTÉVEZ, Pablo A. et al. **Normalized Mutual Information Feature Selection.** IEEE Transactions on Neural Networks 20, pp. 189-201, (2009).

FAWCETT, T. **ROC Graphs: Notes and Practical Considerations for Researchers**., Palo Alto,p.1-38, 16 mar. 2004. Disponível em: <http://www.binf.gmu.edu/mmasso/ROC101.pdf.>. Acesso em: 15 nov. 2008.

FRASER, A. M.; SWINNEY, H. L., **Independent Coordinates for Strange Attractors from Mutual Information**, Physical Review A, vol. 33, no. 2, pp. 1134-1140, 1986.

GORMAN, R. P.; SEJNOWSKI, T. J., **Analysis of Hidden Unitsin a Layered Network Trained to Classify Sonar Targets** in Neural Networks, Vol. 1, pp. 75-89, 1988.

HALL, M.A., **Correlation-based feature subset selection for machine learning**, Ph.D. Dissertation, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 1999.

HAYKIN, Simon. **Redes Neurais: Príncipios e Prática**. 2. ed. Porto Alegre: Bookman, 2002.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Evolução e Perspectivas da Mortalidade Infantil no Brasil,** 1999. Disponível em: <http://www.ibge.gov.br/home/estatistica/populacao/evolucao_perspectivas_mortalidade>. Acesso em: 01 nov. 2008.

JAIN, A. K.; DUIN, R. P.; MAO, J., **Statistical pattern recognition: A review**, IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 1, pp. 4–37, Jan. 2000.

KOHAVI, J.; PFLEGER, K., **Irrelevant features and the subset selection Problem,** Proc. 11th Int. Conf. Mach. Learn., pp. 121–129, 1994.

KOIRAN, P.; SONTAG, E.D. **Neural Networks With Quadratic V-C Dimension.** Journal of Computer and System Sciences 54, 190-198, 1997.

KWAK, N.; CHOI, C.-H., **Input feature selection for classification problems**, IEEE Trans. Neural Net., vol. 3, no. 1, pp. 143–159, Jan. 2002.

LIU, H**.** *et al.* **Feature Selection with Dynamic Mutual Information**. Pattern Recognition, Elsevier Ltd. 2008.

MCCULLOCH, W. S.; PITTS, W., **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biophysic, 5:115–133, 1943.

MITCHELL, T. M. **Machine Learning.** Burr Ridge, Illinois: McGraw-Hill, 1997.

PNUD - PROGRAMA DAS NAÇÕES UNIDAS PARA O DESENVOLVIMENTO. **Atlas do Desenvolvimento Humano no Brasil**. Disponível em: <http://www.pnud.org.br/atlas/>. Acesso em: 01 set. 2008.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J., **Learning Representations by Back-Propagating Errors.** Nature, 1986.

SETIONO, R.; LIU, H., **Neural network feature selector,** IEEE Trans. Neural Networks, vol. 8, maio 1997.

SHANNON, C. E, **A Mathematical Theory of Communication,** The Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, julho, outubro, 1948.

SHIMAZAKI, H.; SHINOMOTO, S., **A Method for Selecting the Bin Size of a Time Histogram**, Neural Computation 19, 1503-1527, Massachusetts Institute of Technology, 2007.

WEKA - WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS (Org.). **Weka 3.5 - Data Mining with Open Source Machine Learning Software in Java.** Vers. 3.5. Computer software. [S.l.]: WEKA, 2008. Disponível em: <http://www.cs.waikato.ac.nz/ml/weka>. Acesso em: 01 set. 2008.

YU, L.; LIU, H., **Efficient Feature Selection via Analysis of Relevance and Redundancy**, Journal of Machine Learning Research 5, pp. 1205–1224, 2004.

Assinaturas

Paulo Jorge Leitão Adeodato
Orientador Trabalho de Graduação

Paulo Fagner Tenório Barros de Morais
Aluno