ANDRÉ SPINELLI SCHIAVONI

UM ESTUDO COMPARATIVO DE MÉTODOS PARA BALANCEAMENTO DO CONJUNTO DE TREINAMENTO EM APRENDIZADO DE REDES NEURAIS ARTIFICIAIS

LAVRAS - MG
2010

ANDRÉ SPINELLI SCHIAVONI

UM ESTUDO COMPARATIVO DE MÉTODOS PARA BALANCEAMENTO DO CONJUNTO DE TREINAMENTO EM APRENDIZADO DE REDES NEURAIS ARTIFICIAIS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Orientador:

Me. Cristiano de Leite Castro

LAVRAS - MG
2010

ANDRÉ SPINELLI SCHIAVONI

UM ESTUDO COMPARATIVO DE MÉTODOS PARA BALANCEAMENTO DO CONJUNTO DE TREINAMENTO EM APRENDIZADO DE REDES NEURAIS ARTIFICIAIS

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

APROVADA em ____ de _____ de _____

_____

Dr. Ricardo Martins de Abreu Silva – UFLA

_____

Dr. Wilian Soares Lacerda – UFLA

_____

Me. Cristiano Leite de Castro

Orientador

LAVRAS - MG
2010

Ao meu pai Rogério.

Aos meus avôs Vicente (in memoriam) e José Geraldo (in memoriam).

A minha mãe Silvani.

Às minhas avós Elza (in memoriam) e Carmelita.

A minha irmã Natália e meu irmão Fábio.

Aos meus amigos da república Vakatolada.

À minha namorada e parceira Karen.

DEDICO.

AGRADECIMENTOS

Agradeço primeiramente a Deus que me proporcionou uma longa e abençoada caminhada durante todo o curso.

Agradeço aos meus pais Rogério e Silvani por acreditarem em mim e me sustentarem em toda a minha estadia em Lavras e ao meu irmão Fábio e minha irmã Natália que sempre me apoiaram nesta fase de minha vida.

Agradeço a todos os meus colegas de curso, especialmente à turma 2007/01, que sempre estiveram juntos com amizade e companheirismo para o sucesso de todos.

Agradeço a todos os meus companheiros da República Vakatolada, André Luiz, Diego, Marcos, Danilo, Leonardo, Anderson e Felipe que me suportaram durante toda essa jornada.

Agradeço à minha namorada Karen que sempre me apoiou nos momentos alegres e tristes com muito carinho e amor.

Agradeço a todos os professores do departamento de Ciência da Computação pelo conhecimento transmitido e dedicação para a minha formação, em especial, ao professor Cristiano Leite de Castro por acreditar em mim e me orientar na construção deste trabalho.

Finalmente agradeço à UFLA por proporcionar que tudo isto fosse possível!

RESUMO

Estudos na área de aprendizado supervisionado têm mostrado que classificadores induzidos a partir de bases de dados desbalanceadas não têm apresentado um bom desempenho. Uma possível solução para este problema é balancear o conjunto de treinamento. Este trabalho propõe um estudo, implementação e comparação de métodos para o balanceamento artificial do conjunto de treinamento para aprendizado supervisionado de Redes Neurais Artificiais. Para a realização deste estudo foram selecionados cinco métodos: Smote, Smote + ENN, Smote + Tomek Link, BED e ADASYN. Ao final foi feita uma análise dos resultados através de métricas extraídas da Análise ROC.

Palavras-chave**:** Classes Desbalanceadas, Redes Neurais Artificiais, Análise ROC, Smote, BED, ADASYN, Tomek Link.

ABSTRACT

Studies in supervised learning have shown that classifiers induced from imbalanced data sets have presented a reduced performance. A intuitive solution for this problem is to balance the training set. This work presents a study of resampling methods for balancing training sets of neural networks. Particularly, five methods were selected and tested: Smote, Smote + ENN, Smote + Tomek Link, BED and ADASYN. The results obtained were analyzed using metrics from ROC analysis.

Keywords***:*** Unbalanced Class, Neural Networks, ROC Analysis, Smote, BED, ADASYN, Tomek Link, ENN.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

MCP	*McCulloch e Pitts*
RNA	Rede Neural Artificial
ROC	*Receiver Operating Characteristic*
AUC	*Area Under Curve*
RMP	*Redes Multilayer Perceptron*
Smote	*Synthetic Minority Over-Sampling Technique*
ENN	*Edited Nearest Neighbors*
BED	*Boundary Elimination and Domination*
ADASYN	*Adaptive Synthetic Sampling*

SUMÁRIO

1 INTRODUÇÃO

Algoritmos de aprendizado supervisionado são freqüentemente utilizados em aplicações computacionais, tais como reconhecimento de padrões, mineração de dados, dentre outras. Muitos aspectos são relevantes na avaliação do desempenho desses algoritmos de classificação. Um deles é a quantidade de amostras e a distribuição das mesmas entre as classes no conjunto de treinamento. Quando o número de exemplos é representativo, mas estes exemplos não estão balanceados, ou seja, há uma grande desproporção do número de dados de uma das classes de treinamento, surge o problema conhecido na literatura como problema de classes desbalanceadas. Nestas condições, modelos de classificação que são otimizados em relação à acurácia global têm tendência de criar modelos triviais, que quase sempre predizem a classe majoritária (BATISTA et al., 2003).

Muitos problemas reais apresentam classes desbalanceadas. Por exemplo, na detecção de fraudes de chamadas telefônicas (PROVOST, 1997) e transações com cartões de crédito (STOLFO, 1997), o número de transações legítimas é muito superior ao número de transações ilegais (BATISTA et al., 2003).

Sendo assim, muitos sistemas de aprendizado não estão preparados para induzir bons modelos de classificação na presença de conjuntos desbalanceados. Ao treinar um modelo com esses dados ele frequentemente apresentará uma boa acurácia para dados pertencentes à classe majoritária, mas uma acurácia inaceitável para os dados da classe minoritária.

Dentre as soluções propostas para o problema de classes desbalanceadas, uma das técnicas mais populares é a utilização de métodos de balanceamento artificial do conjunto de treinamento. A maioria desses métodos baseia-se na remoção ou criação/replicação de exemplos da classe majoritária e minoritária respectivamente.

Além disso, modelos de classificação têm sido frequentemente avaliados através de medidas sensíveis à distribuição de classes como, por exemplo, a acurácia global (ou taxa de erro). A utilização dessas medidas impede que sejam feitas inferências corretas a respeito da qualidade desses métodos na presença de dados naturalmente desbalanceados (BATISTA et al., 2003). Para contornar esse problema, é necessário utilizar métricas que sejam indicadas para avaliação desses classificadores, como é o caso das métricas da análise ROC[1].

Um gráfico ROC pode ser utilizado para analisar a relação entre a taxa de erro e a taxa de acerto, em relação à classe de interesse, assim como fornece métricas para avaliação sensíveis ao problema de desbalanceamento.

1.1 Objetivos do Trabalho

1.1.1 Objetivo Geral

Este trabalho tem como objetivo aplicar diferentes tipos de métodos de balanceamento artificial propostos na literatura, que buscam melhorar o problema das classes desbalanceadas em modelos de classificação baseados em redes neurais artificiais (RNAs). Os resultados são então analisados usando métricas provenientes da análise ROC. Ao final, uma discussão da análise é fornecida com o objetivo de apontar as melhores estratégias para resolver o problema.

1.1.2 Objetivos Específicos

Como objetivos específicos, este trabalho propõe o estudo, implementação e comparação de cinco métodos de balanceamento artificial:

[1] ROC é um acrônimo para Receiver Operating Characteristic

- Smote (CHAWLA, 2002)
- Smote + ENN (BATISTA et al., 2004)
- Smote + *Tomek Links* (BATISTA et al., 2004)
- *BED* (CASTRO, 2009)
- ADASYN (HE, 2008)

Treinamento e avaliação de classificadores baseados em redes *Multi-Layer Perceptron* e análise dos resultados através de métricas ROC.

2 CLASSIFICAÇÃO COM REDES NEURAIS ARTIFICIAIS

Neste capítulo será apresentado um referencial bibliográfico abordando os principais tópicos de um dos métodos de aprendizado supervisionado mais utilizado na literatura, as redes neurais artificiais.

2.1 Redes Neurais Artificiais – Definição

RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (neurônios artificiais) que calculam determinadas funções matemáticas (normalmente não-lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos, essas conexões estão associadas a pesos, os quais armazenam o conhecimento adquirido pelo modelo e servem para ponderar a entrada recebida por cada neurônio da rede. Essa forma de computação não-algorítmica é caracterizada por sistemas que, em algum nível, relembram a estrutura do cérebro humano (BRAGA, 2007).

A solução de problemas por meio de RNAs é bastante atrativa, já que a forma como estes são representados internamente pela rede e o paralelismo natural inerente à arquitetura das RNAs criam a possibilidade de um desempenho superior ao dos modelos convencionais. Em RNAs, o procedimento usual na solução de problemas passa inicialmente por uma fase de aprendizagem, em que um conjunto de exemplos é apresentado para a rede, a qual extrai características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas para o problema (BRAGA, 2007).

2.2 Histórico

O primeiro modelo de um neurônio biológico foi fruto do trabalho pioneiro de Warren McCulloch e Walter Pitts, em 1943 (PITTS, 1943). McCulloch, psicólogo e neurofisiologista, dedicaram sua carreira à tentativa de representar e modelar eventos no sistema nervoso. Pitts, um matemático recém-graduado, juntou-se a ele em 1942. No trabalho publicado em 1943, "*A logical Calculus of the Ideas Immament in Nervous Activity*" (PITTS, 1943), são apresentadas uma discussão sofisticada de redes lógicas de neurônios artificiais, além de novas ideias sobre máquinas de estados finitos, elementos de decisão de limiar lineares e representações lógicas de várias formas de comportamento e memória. O trabalho de McCulloch e Pitts concentrou-se muito mais em descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais do que em apresentar técnicas de aprendizado.

O primeiro trabalho demonstrando ligação direta com o aprendizado de redes artificiais de que se tem notícia foi apresentado em 1949 por Donald Hebb (HEBB, 1949), em 1949. Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através de variação dos pesos de entrada dos neurônios. Mais tarde Widrow e Hoff sugeriram uma regra de aprendizado conhecido como regra de Widrow-Hoff (WINDROW, 1960), ou regra delta, que é ainda hoje bastante utilizada.

Em 1958, Frank Rosenblatt (ROSENBLATT, 1958) demonstrou, com o seu novo modelo, o perceptron, que se fossem acrescidas sinapses ajustáveis, as RNAs com neurônios MCP poderiam ser treinadas para classificar certos tipos de padrões. Rosenblatt descreveu uma topologia de RNA, estruturas de ligação entre os neurônios e, o mais importante, propôs um algoritmo para treinar a rede para executar determinados tipos de funções.

Em 1969, Minsky e Papert (PAPERT, 1969) chamaram a atenção para algumas tarefas que o perceptron de uma única camada descrito por Rosenblatt

não era capaz de executar. O perceptron simples, como ficou conhecido o modelo de uma única camada de pesos ajustáveis, está limitado à resolução de problemas linearmente separáveis, ou seja, problema cuja solução somente pode ser obtida dividindo-se o espaço de entrada em duas regiões por meio de uma superfície linear. O perceptron, por exemplo, não consegue detectar paridade conectividade e simetria, que são problemas não - linearmente separáveis (BRAGA, 2007).

Nos anos de 1970, a abordagem conexionista ficou adormecida, em grande parte devido à repercussão do trabalho de Minsky e Papert que eram bastante pessimistas em relação à capacidade de RNAs resolverem problemas mais complexos, apesar de alguns poucos pesquisadores continuarem a trabalhar na área.

Em 1982, John Hopfield (HOPFIELD, 1982) publicou um artigo que chamou a atenção para as propriedades associativas de RNAs. Esse artigo foi responsável por parte da retomada das pesquisas na área. O grande feito de Hopfield foi, sem dúvida, mostrar a relação entre redes recorrentes auto-associativas e sistemas físicos, o que também abriu espaço para a utilização de teorias correntes da física para estudar tais modelos. Não obstante, a descrição do algoritmo de *back-propagation* alguns anos mais tarde (RUMELHART, 1986) mostrou que a visão de Minsky e Papert sobre o perceptron era bastante pessimista. As RNAs de múltiplas camadas são capazes de resolver "problemas difíceis de aprender" (BRAGA, 2007).

Desde o ressurgimento do interesse pelas RNAs na década de 1980 até os dias atuais, a área passou por grandes transformações, tanto devida ao avanço da tecnologia, sobretudo na microeletrônica e no poder computacional, como nas pesquisas na área de inteligência artificial. Após a descrição do algoritmo de back-propagation, boa parte das pesquisas na área se concentrou em propor variações do algoritmo que tivessem maior velocidade de convergência. Paralelamente,

surgiram varias propostas de aceleração do treinamento por meio de implementações em hardware.

2.3 Neurônio Artificial: modelo MCP

O modelo de neurônio artificial proposto por McCulloch e Pitts (PITTS, 1943) é uma simplificação do que se sabia na época a respeito do neurônio biológico. Sua descrição matemática resultou em um modelo com *n* terminais de entrada que recebem os valores, $x = (x_1, x_2, \dots, x_n)$ e apenas um terminal de saída y. Para representar o comportamento das sinapses, os terminais de entrada do neurônio têm pesos acoplados $w = (w_1, w_2, \dots, w_n)$ cujos valores podem ser positivos ou negativos, dependendo das sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular i no neurônio pós-sináptico é dado por $(w_i x_i)$. Os pesos determinam "em que grau" o neurônio deve considerar sinais de disparo que ocorrem naquela conexão (BRAGA, 2007).

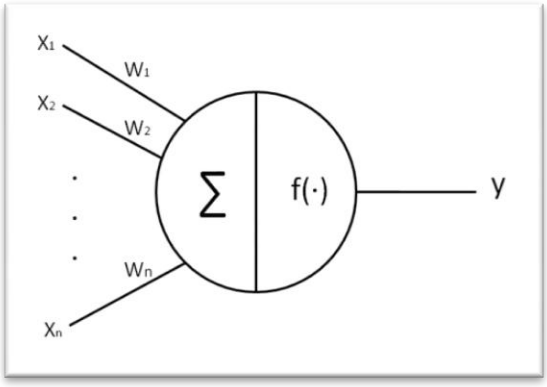

Figura 1 Neurônio de McCulloch, extraída de (BRAGA, 2007).

Um neurônio biológico dispara quando a soma dos impulsos que ele recebe ultrapassa o seu limiar de excitação (*threshold*). Esse comportamento do neurônio

biológico, por sua vez, é representado no modelo artificial por um mecanismo simples, que faz a soma dos valores $(w_i x_i)$ recebidos pelo neurônio (soma ponderada) e decide se o neurônio deve ou não disparar (saída igual a 1 ou a 0), comparando a soma obtida ao limiar ou threshold do neurônio. No modelo MCP, a ativação do neurônio é obtida através da aplicação de uma função de ativação, que ativa ou não a saída, dependendo do valor da soma ponderada das suas entradas (BRAGA, 2007).

2.4 Funções de Ativação

A função de ativação é responsável por gerar a saída y do neurônio a partir dos valores dos vetores de pesos $w = (w_1, w_2, \dots, w_n)$ e de entrada $x = (x_1, x_2, \dots, x_n)^T$. A função de ativação de um neurônio MCP, apresentada na Equação 1, é do tipo degrau deslocada ao limiar de ativação θ em relação à origem, ou seja, a saída y terá 1 para $\sum_{i=1}^{n} x_i w_i \geq \theta$ e 0 para $\sum_{i=1}^{n} x_i w_i < \theta$.

$$f(u) = \begin{cases} 1 \; se \; u \geq \theta \\ 0 \; se \; u < \theta \end{cases} \tag{1}$$

onde $u = \sum_{i=1}^{n} x_i w_i$ e θ é o limiar da função de ativação

Uma aproximação contínua da função degrau, cuja derivada é conhecida em sua forma analítica, é a função mostrada na Equação 2. Essa função além de ser diferençável, possui uma região semilinear, que pode ser importante na aproximação de funções contínuas.

$$f(u) = \frac{1}{1+e^{-\beta u}} \tag{2}$$

onde β é a inclinação da função

Dependendo do tipo do problema a ser abordado, neurônios com funções do tipo lineares também pode ser utilizados como mostrado na Equação 3.

$$f(u) = u \tag{3}$$

Há vários outros tipos de funções de ativação, e para cada problema uma delas deve ser escolhida para resolvê-lo. Alguns gráficos de funções de ativação estão exemplificados abaixo.

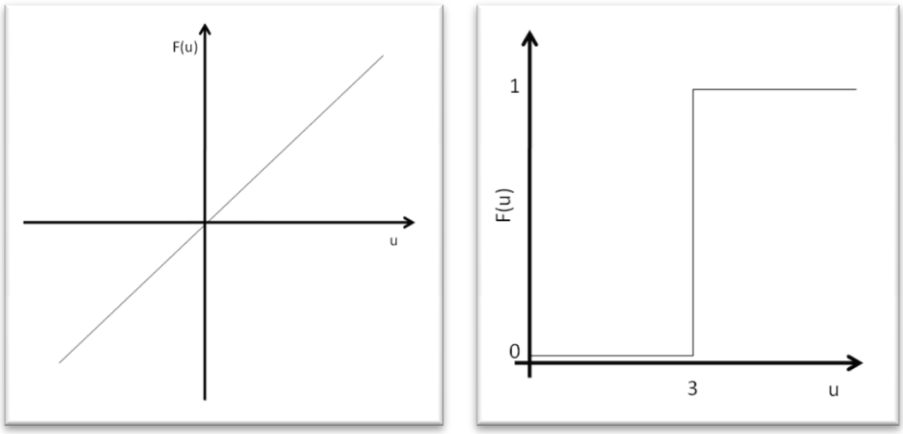

Figura 2 Função de ativação linear e degrau respectivamente, extraída de (BRAGA, 2007).

2.5 Perceptron

O trabalho original de McCulloch e Pitts (PITTS, 1943) enfocou a modelagem de um neurônio biológico e sua capacidade computacional por meio da apresentação de vários exemplos de topologias de rede com capacidade de execução de funções booleanas. Foi somente em 1958 que o conceito de aprendizado em RNAs foi introduzido. O modelo proposto por Rosenblatt (ROSENBLATT, 1958), conhecido como perceptron era composto por uma estrutura de rede, tendo como unidades básicas neurônios MCP, e por uma regra de aprendizado. Alguns anos mais tarde Rosenblatt demonstrou o teorema de convergência do perceptron, que mostra que um neurônio MCP treinado com o algoritmo de aprendizado do perceptron sempre converge caso o problema em questão seja linearmente separável (ROSENBLATT, 1958).

A topologia descrita por Rosenblatt era composta por unidades de entrada (retina), por um nível intermediário formado pelas unidades de associação e por um nível de saída formado pelas unidades de resposta. Embora essa topologia possua três níveis, ela é conhecida como perceptron de única camada, já que somente o nível de saída (unidades de resposta) apresenta propriedades adaptativas. A retina consiste basicamente em unidades sensoras, e as unidades intermediárias de associação, embora formados por neurônios MCP, possuem pesos fixos, definidos antes do período de treinamento.

Embora tenha causado muita euforia na comunidade científica da época, esta não teve grande duração devido a sua limitação computacional já descrita por Minsky e Papert (PAPERT, 1969) em 1969. Essa visão mudou com as descrições do algoritmo de *backpropagation* em 1986. Foi em conseqüência de trabalhos como esse que a comunidade científica ganhou um novo impulso e as áreas de aprendizado de máquina e RNAs se desenvolveram.

2.6 Aprendizado

Uma das características mais importantes das RNAs é a sua capacidade de aprender por meio de exemplos. A etapa de aprendizado de uma RNA consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões, que guardam, ao final do processo, o conhecimento que a rede adquiriu do ambiente externo. Uma definição geral do que vem a ser aprendizado pode ser expressa da seguinte forma (MENDEL, 1970).

Aprendizado é o processo pelo qual os parâmetros livres de uma rede neural são ajustados por meio de uma forma continuada de estímulo pelo

ambiente externo, sendo o tipo especifico de aprendizado definido pela maneira particular como ocorrem os ajustes dos parâmetros livres.

É importante ressaltar que o conceito de aprendizado está relacionado com a melhoria do desempenho da rede segundo algum critério preestabelecido. O erro quadrático médio da resposta da rede em relação ao conjunto de dados fornecido pelo ambiente, por exemplo, é utilizado como critério de desempenho pelos algoritmos de correção de erros. Assim quando estes algoritmos são utilizados no treinamento de RNAs, espera-se que o erro diminua à medida que o aprendizado prossiga.

De uma forma genérica, o valor do vetor de pesos *w(t+1)* no instante *t+1* pode ser escrito como mostrado na Equação 4,

$$w(t+1) = w(t) + \Delta w(t) \tag{4}$$

onde w(t) e w(t+1) representam os valores dos pesos nos instantes *t* e *t+1*, respectivamente, e $\Delta w(t)$ é o ajuste aplicado aos pesos (BRAGA, 2007).

Os algoritmos de aprendizado diferem basicamente, na forma como o $\Delta w(t)$ é calculado. Há vários algoritmos diferentes para treinamento de redes neurais, podendo os mesmos serem agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado.

2.6.1 Aprendizado Supervisionado

Aprendizado supervisionado implica, necessariamente, a existência de um supervisor, ou professor externo, o qual é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída calculada pela mesma,

comparando-a com a saída desejada. Como a resposta da rede é função dos valores atuais do seu conjunto de pesos, estes são ajustados de forma a aproximar a saída da rede da saída desejada (BRAGA, 2007).

Em outras palavras aprendizado supervisionado pode ser definido como uma técnica para deduzir uma função a partir de um conjunto de dados de treinamento, que quando criada possa predizer automaticamente a classe de um exemplo não classificado.

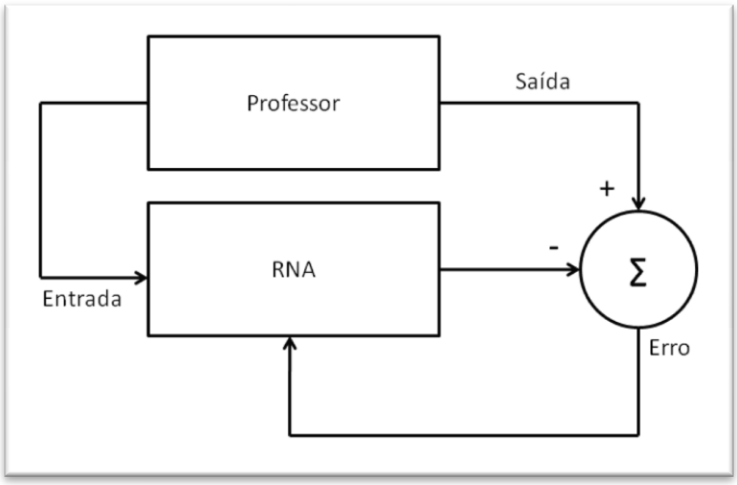

Figura 3 Aprendizado Supervisionado, extraída de (BRAGA, 2007).

O aprendizado supervisionado se aplica a problemas em que se deseja obter um mapeamento entre padrões de entrada e saída. Os exemplos mais conhecidos de algoritmos para aprendizado supervisionado são a regra delta (WINDROW, 1960) e a sua generalização para as redes de múltiplas camadas, o algoritmo de back-propagation (RUMELHART, 1986).

O exemplo mais típico de aprendizado supervisionado é o aprendizado por correção de erros, em que se procura minimizar o erro da resposta atual da rede em relação à saída desejada. A expressão genérica para o erro $e(t)$ no instante de

tempo t pode ser escrita como: $e = \gamma_d(t) - \gamma(t)$ em que $\gamma_d(t)$ é a saída desejada e $\gamma(t)$ é a resposta atual calculada pela rede. A forma genérica para atualização dos pesos por correção de erros, utilizada pelos algoritmos de aprendizado é apresentada na Equação 5,

$$w_i(t+1) = w_i(t) + \mu e(t) x_i(t) \tag{5}$$

onde $w_i(t)$ corresponde ao peso de entrada *i*, μ a taxa de aprendizado, e(t) é uma medida de erro e $x_i(t)$ a entrada *i* do neurônio (BRAGA, 2007).

Segundo a Equação 5, o ajuste dos pesos deve ser proporcional ao produto do erro pelo valor de entrada da sinapse naquele instante de tempo. Essa expressão aparece tanto do algoritmo de treinamento do perceptron (ROSENBLATT, 1958) quanto no algoritmo para treinamento de redes ADALINE (WINDROW, 1960) e na posterior generalização para o algoritmo de back-propagation (RUMELHART, 1986).

2.6.2 Aprendizado Não-Supervisionado

No aprendizado não-supervisionado, como o próprio nome sugere, não há um professor ou supervisor externo para acompanhar o processo de aprendizado. Neste esquema de treinamento somente os padrões de entrada estão disponíveis para a rede, ao contrário do aprendizado supervisionado, cujo conjunto de treinamento possui pares de entrada e saída. Durante o processo de aprendizado os padrões de entrada são apresentados continuamente à rede, e a existência de regularidades nesses dados faz com que o aprendizado seja possível. Regularidade e redundância nas entradas são características essenciais para haver aprendizado não-supervisionado. Um esquema genérico do aprendizado não-supervisionado é apresentado na Figura 4.

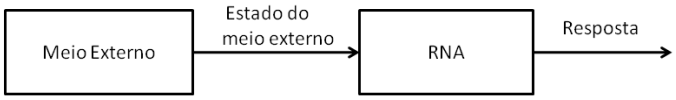

Figura 4 Aprendizado Não-Supervisionado Extraída de (BRAGA, 2007).

2.7 Redes Perceptron de Múltiplas Camadas

Frank Rosenblatt desenvolveu o modelo *Perceptron* para RNAs em 1957. Este modelo ficou conhecido como *perceptron* de camada única, pois, apesar de possuir três níveis, um de entrada, um intermediário e um de saída, apenas o nível de saída possuía propriedades adaptativas. Em 1962, foi provado que este modelo conseguia atingir a convergência sempre que o problema em questão fosse linearmente separável.

O maior problema do *perceptron* é o fato de não conseguir tratar bem os problemas que não sejam linearmente separáveis.

As Redes *Multilayer Perceptron* (MLP) possuem múltiplas camadas intermediárias (escondidas) e conseguem tratar de problemas que não sejam linearmente separáveis. Em uma rede multicamadas o processamento feito por cada neurônio depende dos resultados obtidos pelos neurônios da camada anterior. Os números de neurônios nas camadas escondidas também influenciam nas respostas da rede, já que quanto maior o número de neurônios maior o número de ligações (HAYKIN, 2007).

A Figura 5 mostra um exemplo de RNA com três neurônios na camada de entrada, quatro na camada intermediária (escondida) e dois na camada de saída.

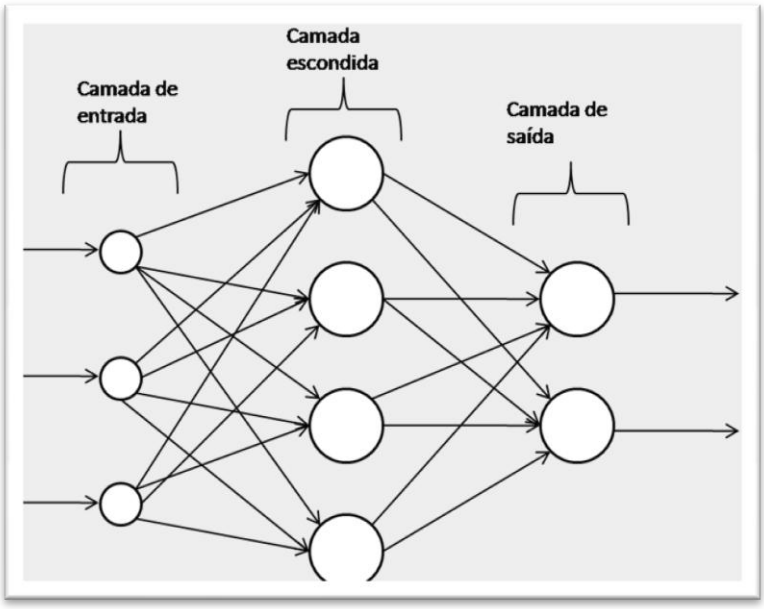

Figura 5 RNA com três camadas, extraída de (HAYKIN, 2007).

2.7.1 Algoritmo *Backpropagation*

O termo *backpropagation* surgiu após 1985. No entanto, a idéia básica foi primeiramente descrita por Werbos em sua tese de doutorado em 1974. Em 1986, foi redescoberto por Rumelhart, Hinton e Williams e popularizado através da publicação do livro *Parallel Distributed Processing* de Rumelhart e McClelland em 1986 (HAYKIN, 2007).

O desenvolvimento do *backpropagation* representa um marco fundamental em redes neurais, pois é um método computacionalmente eficiente para o treinamento de redes MLPs e por ter resolvido o problema de realizar a propagação reversa do erro em RNAs com múltiplas camadas, problema este que atrasou por muitos anos o desenvolvimento da área de redes neurais artificiais.

O algoritmo *backpropagation* baseia-se na heurística do aprendizado por correção de erro (em que o erro é retro-propagado da camada de saída para as camadas intermediárias da RNA). Este algoritmo pode ser visto como uma generalização do Algoritmo *Least Mean Square* (LMS) desenvolvido por Bernard Widrow (CASTRO, 1995).

Basicamente, o algoritmo *backpropagation* consiste de dois passos através das diferentes camadas do MLP: um passo direto e um passo reverso.

No passo direto um padrão de atividade do processo a ser aprendido (ou vetor de entrada) é aplicado aos nós de entrada do MLP e o seu efeito se propaga através da rede, camada por camada, produzindo na camada de saída a resposta do MLP à excitação aplicada (vetor de saída). Durante o passo direto os pesos sinápticos são todos fixos.

Durante o passo reverso os pesos sinápticos são todos ajustados de acordo com a regra de aprendizado por correção de erro. Especificamente, a resposta do MLP à excitação é subtraída de um padrão de resposta desejado para aquela excitação aplicada, de forma a produzir um sinal de erro, de forma semelhante ao algoritmo LMS. Este sinal de erro é, então, propagado de volta através dos mesmos neurônios utilizados no passo direto, mas no caminho contrário do fluxo de sinal nas conexões sinápticas - daí o nome *backpropagation* (CASTRO, 1995).

Os pesos sinápticos são, então, ajustados de forma que a resposta obtida do MLP aproxime-se mais do padrão de resposta desejado.

2.8 Conclusões do capítulo

Neste capítulo foram abordados os principais tópicos necessários para a leitura desse documento visando listar um referencial teórico para melhor entendimento dos assuntos descritos no trabalho.

3 CLASSES DESBALANCEADAS E ANÁLISE ROC

O problema de classes desbalanceadas é caracterizado a partir de um conjunto de dados treinamento em que o número de exemplos em cada classe não se encontra em equilíbrio.

Neste capítulo será abordado o problema das classes desbalanceadas, uma breve descrição das soluções propostas na literatura para aliviar o problema e métodos de análise de avaliação, incluindo fundamentos da análise ROC (*Receiver Operating Characteristic*).

3.1 Caracterização do Problema de Classes Desbalanceadas

Tecnicamente qualquer conjunto de dados que apresenta uma desigualdade da distribuição entre as classes pode ser considerada desbalanceada. Não é raro o desbalanceamento entre as classes serem da ordem de 100:1, 1000:1 e 10000:1 onde, em cada caso, uma classe está extremamente inserida entre os exemplos da outra classe (HE, 2008).

A fim de destacar as implicações do problema do aprendizado desbalanceado, será apresentado um exemplo de uma aplicação biomédica. Considere o conjunto de dados "*Mammography Data Set*", uma coleção de imagens obtidas a partir de uma série de exames de mamografia realizados sobre um conjunto de pacientes distintos, que tem sido amplamente utilizada na análise de algoritmos de classificação (CHAWLA, 2002).

Analisando as imagens em um sentido binário, as duas classes possíveis seriam positiva e negativa, respectivamente, das pessoas que possuem câncer e das pessoas saudáveis. A partir de uma experiência é esperado que o número de pacientes saudáveis seja muito maior que o número das pessoas com câncer; na verdade, este conjunto de dados contém 10.923 amostras "negativas" (classe

majoritária) e 260 amostras “positivas” (classe minoritária). Preferivelmente, espera-se obter um classificador que forneça um grau equilibrado de acurácia preditiva (idealmente 100 por cento) para ambas as classes (majoritária e minoritária) do conjunto de dados. Na realidade, os classificadores tendem a proporcionar um grave desequilíbrio no grau de precisão, com a classe majoritária com acurácia perto de 100 por cento e a classe minoritária tendo acurácia de 0-10 por cento, por exemplo (CHAWLA, 2002). Suponha que um classificador atinja dez por cento de acurácia na classe minoritária do conjunto de dados. Analiticamente, isso sugere que 234 amostras da classe minoritária são classificadas erroneamente como da classe majoritária. A conseqüência disso é equivalente a 234 pacientes cancerosos classificados (diagnosticados) como saudáveis. Na indústria médica, as ramificações de tal conseqüência podem ser extremamente caras, mais do que classificar um paciente saudável como cancerígeno (RAO, 2006). Por isso, é evidente que para este domínio, precisamos de um classificador que dará (CHAWLA, 2002) alta acurácia para a classe minoritária sem comprometer gravemente a acurácia da classe majoritária (HE, 2009).

3.2 Soluções Propostas na Literatura

Uma das técnicas mais populares para aliviar o problema das classes desbalanceadas baseia-se na modificação do conjunto de dados originais. Tais técnicas são conhecidas como reamostragem de dados Estudos têm mostrado que para vários algoritmos de classificação, um conjunto balanceado proporciona uma melhoria significativa no desempenho das classificações comparadas com um conjunto de dados desbalanceados (HE, 2008). Estes resultados justificam o uso de métodos de reamostragem para aprendizado desbalanceado (PROVOST, 2001). Entretanto, não implica que o classificador não pode aprender a partir de

um conjunto de dados desbalanceado; ao contrário, estudos têm mostrado que classificadores induzidos com certos conjuntos de dados desbalanceados podem ser comparados com classificadores induzidos do mesmo conjunto de dados por algumas técnicas de reamostragem (BATISTA et al., 2004).

Nas próximas seções serão mostradas algumas das principais técnicas de reamostragem de dados utilizadas como base para vários métodos.

3.2.1 *Oversampling* e *Undersampling* Randômico

O método de *oversampling* randômico segue naturalmente sua descrição. Selecionam-se aleatoriamente amostras do conjunto minoritário e adicionam em um conjunto E. Assim, é possível incrementar o número de exemplos do conjunto minoritário original adicionando os valores presentes no conjunto E com uma simples replicação de dados. Enquanto o método de oversampling adiciona dados ao conjunto minoritário, *undersampling* randômico remove dados da classe majoritária selecionados aleatoriamente.

3.2.2 *Undersampling* Informativo

O objetivo desse método é superar as deficiências da perda de informação gerada pelo *undersampling* tradicional. Umas das técnicas de *undersampling* informativo utiliza o algoritmo de KNN (*K-Nearest Neighbors)* para selecionar os exemplos a serem removidos para realizar a sub-amostragem. Com base nas características do algoritmo de KNN algumas técnicas são utilizadas para realizar o *undersampling* de maneira mais eficiente.

3.2.3 *Oversampling* por Geração de Dados Sintéticos

A técnica de *oversampling* sintético tem se mostrado um poderoso método em várias aplicações (CHAWLA, 2002). O algoritmo Smote, por exemplo, cria dados artificiais com base nas características espaciais entre exemplos da classe minoritária. Especificamente, para um subconjunto S_{min} (conjunto da classe minoritária), considere os *k* vizinhos mais próximos para cada exemplo x_i que pertençam a S_{min} para algum valor inteiro *k*. Dependendo da quantidade de *oversampling* escolhida os k vizinhos mais próximos são escolhidos aleatoriamente. Amostras sintéticas são geradas da seguinte maneira: calcule a distância euclidiana entre o vetor de pontos (amostras) em questão e seu vizinho mais próximo. Multiplique essa distância por um número aleatório entre zero e um e o adicione ao vetor de pontos em consideração. Isso causa a seleção de um ponto ao longo de uma linha entre os dois pontos escolhidos. Esta aproximação eficaz faz com que a região da classe minoritária fique com mais força para tornar-se mais geral (CHAWLA, 2002).

A Figura 6 mostra um exemplo do algoritmo Smote.

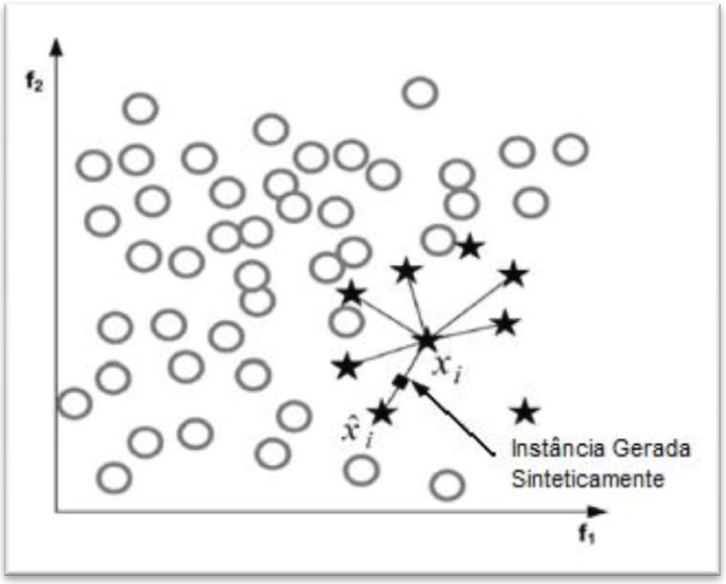

Figura 6 Criação de dados Sintéticos, extraída de (HE, 2009).

3.3 Análise ROC

Análise ROC é uma técnica para visualizar, avaliar, organizar e selecionar classificadores baseado em seus desempenhos. Para realizar estas análises, gráficos ROC podem mostrar a relação entre taxas de acertos e alarmes falsos (taxas de erros) dos classificadores. Um dos primeiros usos dos gráficos ROC em aprendizado por máquina apresentado em (SPACKMAN, 1989), que demonstrou a utilidade das curvas ROC na avaliação e comparação entre algoritmos.

A análise ROC também tem sido de grande utilidade para visualizar e analisar o comportamento de sistemas de diagnóstico (SWETS, 1988), principalmente na medicina. Em (VILARINO et al., 2006) os gráficos ROC são utilizados para otimizar o limiar que classifica classes de contração ou não contração intestinal em imagens geradas através do exame de endoscopia. Atualmente há um grande crescimento no uso de gráficos ROC na comunidade de aprendizado por máquina, em parte pelo uso desta técnica na análise de classificadores (SILVA, 2006).

3.3.1 Matriz de Confusão

Para se entender os erros gerados por um classificador é possível resumi-los a uma matriz de erros denominada *matriz de confusão*. A Tabela 1 ilustra os possíveis resultados que possam ser gerados por este classificador.

Tabela 1 Matriz de Confusão, extraída de (HE, 2009).

	Predição Positiva	Predição Negativa
Classe Positiva	Verdadeiro Positivo (tp)	Falso Negativo (fn)
Classe Negativa	Falso Positivo (fp)	Verdadeiro Negativo (tn)

A partir desta matriz é possível extrair métricas de qualidade para avaliar o desempenho de um sistema de aprendizagem, dentre elas a taxa de erro e a acurácia.

Taxa de Erro: $E = \frac{fp+fn}{tp+fp+tn+fn}$ e Acurácia: $Acc = \frac{tp+tn}{tp+fp+tn+fn}$

Estas duas métricas, taxa de erro (E) e acurácia (Acc) são muito utilizadas na análise de sistemas de aprendizado. Entretanto, quando as classes são naturalmente desbalanceadas, essas medidas podem inferir resultados que facilmente podem ser mal interpretados ou até mesmo sendo considerados como resultados enganosos. Como exemplo, considere um classificador que tenha 99% de acurácia (taxa de erro de 1%). Se o conjunto de dados apresenta 99% de exemplos pertencentes à classe majoritária, ele "acertaria" a classificação apenas rotulando os dados como sendo da classe majoritária, e erraria todas as classificações da classe minoritária. Este mesmo classificador poderia ser considerado ótimo, mas na verdade não conseguem ter um bom desempenho para ambas as classes.

Outro fato relevante a ser considerado contra o uso da acurácia para a avaliação de um sistema de aprendizado é que essa métrica considera igualmente importante diferentes erros de classificação. Na prática isso não é muito aconselhável, visto que diferentes tipos de classificação podem ter conseqüências relevantes para diferentes casos. Por exemplo, um sistema para classificar pessoas como "saudável" e "doente" constitui um exemplo de erros com pesos diferentes, pois um paciente saudável diagnosticado como doente é considerado um erro menos grave, uma vez que este erro pode ser corrigido em exames futuros. Um erro que pode ser fatal é no caso em que um paciente doente é diagnosticado como saudável.

Para problemas desbalanceados e/ou com diferentes custos de classificação é mais interessante o uso de métricas que dissocia os erros, ou acertos, ocorridos em cada uma das classes. Da Matriz de Confusão é possível extrair quatro métricas de desempenho que medem diretamente o desempenho de classificação para as classes positivas e negativas independentemente, são elas:

- Taxa de falsos positivos: $FP = \frac{fp}{fp+tn}$ é a porcentagem de casos negativos classificados erroneamente como pertencentes à classe positiva.
- Taxa de verdadeiros positivos: $TP = \frac{tp}{tp+fn} = 1 - FN$ é a porcentagem de casos positivos classificados corretamente como pertencentes à classe positiva;
- Taxa de falsos negativos: $FN = \frac{fn}{tp+fn}$ é a porcentagem de casos positivos classificados erroneamente como pertencentes à classe negativa, também conhecida como sensibilidade;
- Taxa de verdadeiros negativos: $TN = \frac{tn}{fp+tn} = 1 - FP$ é a porcentagem de casos negativos classificados corretamente como pertencentes à classe negativa, também conhecida como especificidade;

Estas quatro medidas de desempenho têm a vantagem de serem independentes dos custos das classes e da probabilidade a priori. É óbvio que o objetivo principal de um classificador é minimizar as taxas de falsos positivos e falsos negativos, similarmente, maximizar as taxas de verdadeiros positivos e verdadeiros negativos (MONARD, 2003). Os gráficos ROC podem ser utilizados para analisar a relação entre as taxas de verdadeiros positivos e verdadeiros negativos, ou falsos negativos e falsos positivos, para um ou vários classificadores (PROVOST, 1997).

Outras importantes medidas que podem ser extraídas da matriz de confusão são:

- Sensibilidade = $TP = \frac{tp}{tp+fn}$ (6)

onde esta representa a proporção de verdadeiros positivos que foram corretamente identificados.

- Especificidade = $TN = \frac{tn}{tn+fp}$ (7)

onde esta representa a proporção de verdadeiros positivos que foram corretamente identificados.

- $Acurácia = \frac{tp+tn}{tp+fp+fn+tn}$ (8)

onde esta representa a proporção de resultados verdadeiros (verdadeiros positivos e verdadeiros negativos) no resultado. Uma acurácia de 100% significa que os valores medidos são exatamente os mesmos que os valores esperados.

- G-Mean = $\sqrt{TP \times TN}$ (9)

onde esta representa a média geométrica entre as taxas corretas de classificação para a classe positiva (sensibilidade) e negativa (especificidade), respectivamente; Valores elevados de *G-Mean* refletem taxas elevadas e equilibradas de sensibilidade e especificidade.

3.3.2 Espaço ROC

Gráficos ROC foram originalmente utilizados para a detecção de sinais em um canal com ruído e também são utilizados em psicologia para avaliar a capacidade de indivíduos distinguirem entre estímulos e não estímulos; em medicina para analisar a qualidade de um determinado teste clínico; em economia onde é conhecido como gráfico de Lorenz para análise de desigualdade de renda; e em previsão do tempo para se avaliar a qualidade das predições de eventos raros (PRATI, 2008).

Recentemente a análise ROC passou a ser utilizada em sistemas de aprendizado de máquina e mineração de dados como uma ferramenta muito útil e poderosa para a avaliação de modelos de classificação (BRADLEY, 1997). Particularmente é muito útil em ambientes onde existe uma grande desproporção entre as classes ou quando se devem levar em conta diferentes custos para diferentes resultados de classificação.

Um modelo de classificação é representado por um ponto no espaço ROC. Para se obter um ponto no espaço ROC correspondente a um modelo de classificação, calcula-se a taxa de falsos positivos (FP) e a taxa de verdadeiros positivos (TP) deste modelo a partir da matriz de Confusão (Tabela 1) (PRATI, 2008).

A análise de alguns pontos do espaço ROC merece destaque. A figura 7 ilustra um espaço ROC com alguns pontos importantes. O ponto D representa um modelo que não apresenta nenhum falso positivo, mas também não consegue classificar nenhum verdadeiro positivo. O ponto C já mostra uma abordagem inversa onde o modelo sempre classifica a entrada e com isso ele terá uma taxa de FP e TP de 100%. Já o ponto A é o classificador ideal, onde a taxa de FP é nula e a taxa de TP é máxima, isto é, o modelo sempre classifica corretamente todas as entradas classificadas. O ponto B onde a taxa de TP é nula e a taxa de FP é máxima, é o modelo que sempre faz classificações errôneas.

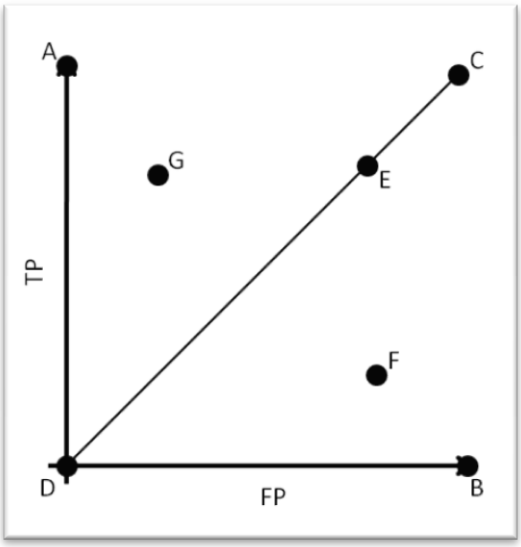

Figura 7 Representação do espaço ROC

Modelos próximos ao canto inferior esquerdo são chamados de "conservativos": eles fazem uma classificação positiva somente se têm grande segurança na classificação. Como conseqüência, eles cometem poucos erros falsos positivos, mas freqüentemente têm baixa taxa de verdadeiros positivos. Modelos próximos ao canto superior direito podem ser considerados "liberais": eles predizem uma das classes com grande freqüência, de tal maneira que classificam a maioria dos exemplos corretamente, mas, geralmente com altas taxas de falsos positivos (PRATI, 2008).

É fácil perceber que um ponto no espaço ROC é melhor que outro se ele está mais acima e a esquerda do que o outro ponto. Na Figura 7 o ponto G poderia ser considerado melhor que o ponto F, uma vez que a taxa de FP do ponto G é menor que a do ponto F e a taxa de TP é maior que a do ponto G.

3.3.3 Curva ROC

Muitos classificadores, tais como árvores de decisão, são desenvolvidos para produzir uma única saída discreta que indica a classe, por exemplo, p ou n, para cada instância. Portanto, um classificador discreto gera apenas um ponto no espaço ROC.

Alguns classificadores, tais como *Naive Bayes* ou redes neurais, naturalmente produzem uma probabilidade ou valor para cada instância que representa o grau de pertinência da instância à determinada classe. A partir destes classificadores, podem-se gerar classificadores discretos com uma simples aplicação de um limiar (*threshold*). Cada limiar produzirá um conjunto distinto de pontos no espaço ROC (SILVA, 2006).

Por exemplo, variando-se o limiar de 1.0 até 0.0 em intervalos igualmente espaçados é possível produzir uma série de pontos no espaço ROC que, interligados por segmentos de reta, produzem uma Curva ROC (Veja Figura 8).

A Figura 8 mostra um exemplo de uma curva ROC produzida a partir de 20 instâncias (pontos). A Tabela 2 representa uma instância, para um dado valor de *threshold*, que a partir dos valores de TPrate e FPrate, se plota um ponto na curva ROC.

Tabela 2 Instâncias geradas por um classificador probabilístico para um dado valor de *threshold*.

Instância	Classe	Valor	Instância	Classe	Valor
1	P	0.9	11	P	0.4
2	P	0.8	12	N	0.39
3	N	0.7	13	P	0.38
4	P	0.6	14	N	0.37
5	P	0.55	15	N	0.36
6	P	0.54	16	N	0.35
7	N	0.53	17	P	0.34
8	N	0.52	18	N	0.33
9	P	0.51	19	P	0.3
10	N	0.505	20	N	0.1

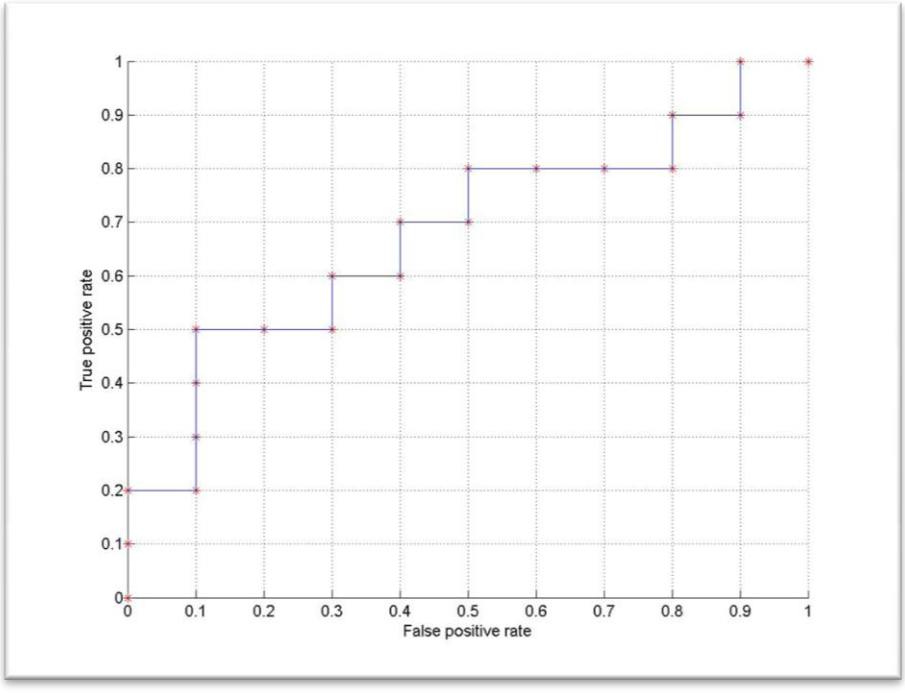

Figura 8 Uma curva ROC criada pela limiarização do conjunto de teste (Tabela 2).

3.4 *Area Under the ROC Curve* (AUC)

Uma curva ROC é uma demonstração bidimensional da performance de um classificador. Para comparar classificadores é preciso reduzir a curva ROC a um valor escalar. Um método comum para realizar esta redução é calcular a área abaixo da curva ROC (AUC). Como a AUC é uma porção da área do quadrado unitário (espaço ROC) seus valores vão de 0.0 à 1.0. A Figura 9 mostra a área abaixo de duas curvas ROC, A e B. O classificador B possui uma área maior e, portanto, tem um melhor desempenho médio.

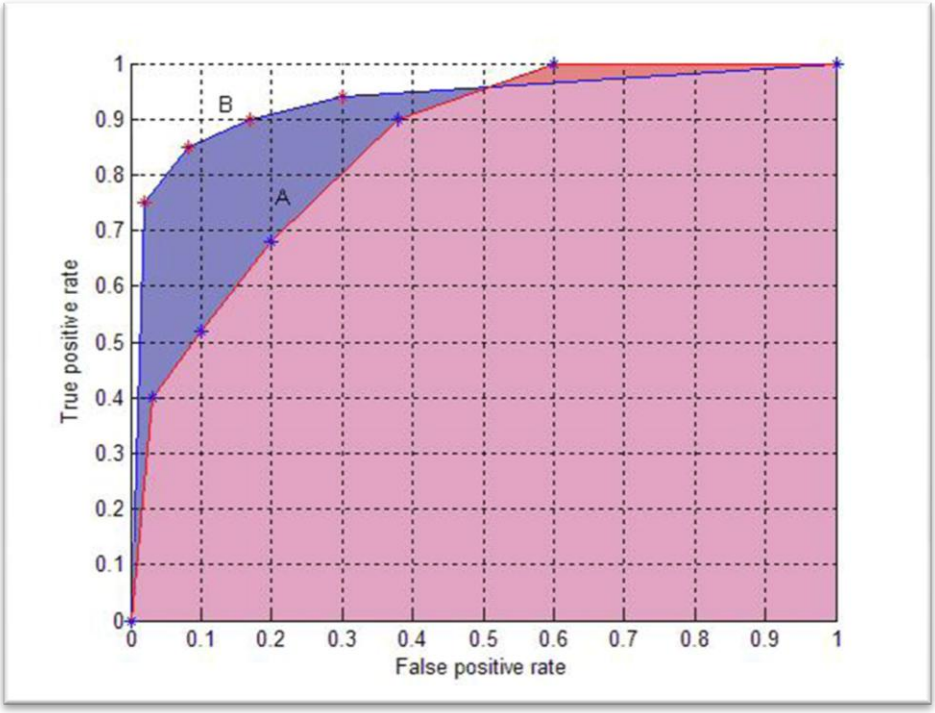

Figura 9 Área abaixo das curvas ROC A e B.

É possível que em algumas regiões do espaço ROC um classificador seja melhor que outro. Na Figura 9 o classificador B é geralmente melhor que A exceto em FP > 0.6, onde A leva uma pequena vantagem.

3.5 Conclusões do Capítulo

Neste capítulo foi apresentado o problema das classes desbalanceadas e como que estão sendo feito estudos para tentar amenizar e contornar esse problema. Foram apresentados alguns métodos mais simples nos quais se baseiam a maioria das técnicas de reamostragem. Também foram apresentadas maneiras para análise dos classificadores e métricas para comparação destes classificadores.

4 METODOLOGIA

Neste capítulo é apresentado o tipo de pesquisa, os procedimentos metodológicos aplicados no trabalho, o detalhamento das técnicas de balanceamento artificial utilizadas para comparação, as bases de dados reais e suas características e as decisões de projeto.

4.1 Tipo de pesquisa

O trabalho pode ser classificado como uma pesquisa tecnológica ou aplicada, pois, baseia-se em um estudo comparativo de métodos de balanceamento artificial de conjunto de dados para aprendizado supervisionado.

Quanto aos objetivos, a pesquisa é caracterizada como uma pesquisa exploratória, pois tem a finalidade de avaliar quais teorias ou conceitos existentes podem ser aplicados a um determinado problema ou se novas teorias e conceitos devem ser desenvolvidos.

Finalmente quanto aos procedimentos a pesquisa é definida como experimental realizada em laboratório visando entender como será o comportamento de cada método de balanceamento artificial para diferentes tipos de base de dados reais.

4.2 Procedimentos Metodológicos

Para fins de ilustração a Figura 9 apresenta um diagrama que descreve a sequência de passos que compõem a metodologia adotada na comparação dos métodos de reamostragem de dados aplicadas a classificadores baseados em RNAs.

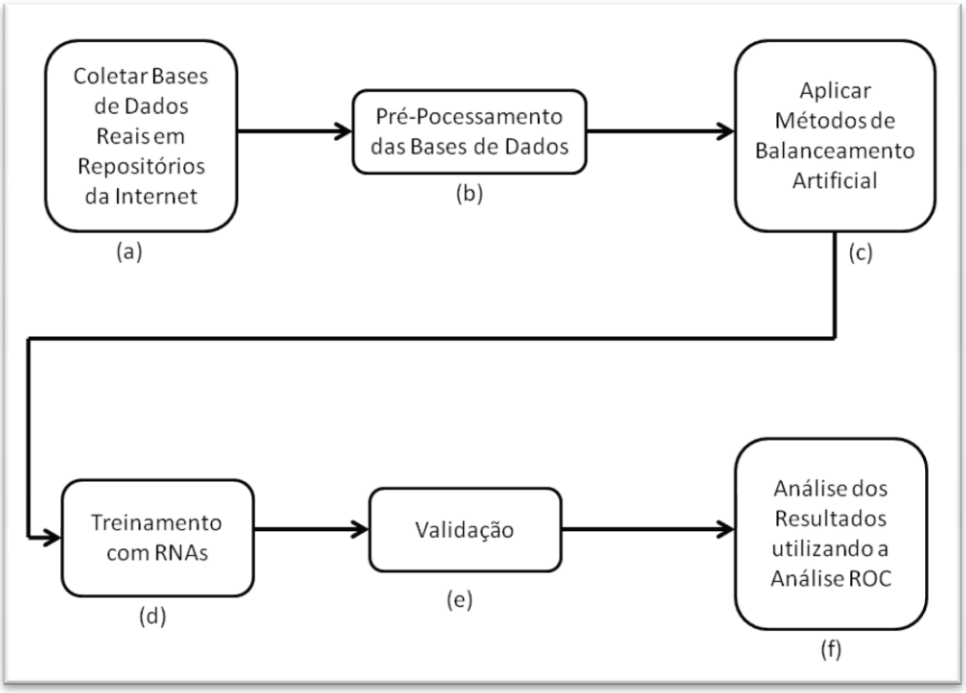

Figura 10 Diagrama da Metodologia

A Figura 10 (a) representa a fase de coleta de bases de dados reais em repositórios na internet que são frequentemente utilizados para avaliar algoritmos de aprendizado. Nesta fase foram escolhidos vários tipos diferentes de bases de dados, desbalanceadas, para servir de entrada para os métodos de balanceamento artificial.

A Figura 10 (b) representa a fase de pré-processamento dos dados selecionados. Nesta fase foram feitas algumas adequações nos dados como, por exemplo, a rotulação adequada das classes, divisões em conjuntos de treinamento e de teste e normalização dos exemplos de entrada.

A Figura 10 (c) representa a fase onde são aplicados os métodos de balanceamento artificial no conjunto de entrada.

A Figura 10 (d) representa a fase de treinamento utilizando um sistema de aprendizado supervisionado. Nesta fase foram utilizadas as redes neurais artificiais para a criação de um classificador a partir da base de dados balanceadas artificialmente pelos métodos da fase anterior.

A Figura 10 (e) representa a fase da validação onde as redes treinadas são avaliadas sobre os conjuntos de testes.

A Figura 10 (f) representa a fase de análise dos resultados obtidos através de métricas extraídas da Análise ROC.

4.3 Estratégias de Balanceamento Artificial Selecionadas

Nesta seção é detalhado cada método de balanceamento artificial usado no trabalho, explicando o seu funcionamento e mostrando como foi implementado.

4.3.1 Smote

Em relação à geração de dados sintéticos a técnica de oversampling (Smote) tem mostrado um grande sucesso em várias aplicações (CHAWLA, 2002). O algoritmo Smote cria dados artificiais com base no recurso da semelhança entre os exemplos da classe minoritária existentes. Especificamente, para um subconjunto $S_{min} \in S$, considere o k vizinho mais próximo para cada exemplo $x_i \in S_{min}$, para algum número inteiro especificado *k;* o k vizinho mais próximo é definido como elemento de S_{min} se a distância euclidiana entre ele e x_i em consideração exibir uma menor magnitude ao longo do espaço n-dimensional X. Para criar um exemplo sintético, seleciona-se aleatoriamente um dos seus k vizinhos mais próximos, então multiplica a diferença entre o correspondente valor por um número aleatório entre [0,1], e finalmente, adiciona este novo valor ao subconjunto S_{min}

$$x_{new} = x_i + (y_i - x_i) * \delta \quad (10)$$

onde $x_i \in S_{min}$ é um exemplo da classe minoritária em consideração, y_i é um dos seus k vizinhos mais próximos de x_i: $y_i \in S_{min}$, e $\delta \in [0,1]$ é um número aleatório. Portanto, o resultado do exemplo sintético gerado de acordo com a equação seis é um ponto ao longo da reta que une o ponto x_i em consideração com um dos k vizinhos mais próximos escolhidos aleatoriamente (HE, 2009).

4.3.2 Smote + Tomek Links

Exemplos próximos à borda e ruído podem ser identificados por meio das ligações Tomek, e removidos do conjunto de dados. Uma ligação Tomek pode ser definida da seguinte maneira:

Sejam e_i e e_j dois exemplos de classes diferentes. Seja d uma função de distância entre exemplos. Um par de exemplos (e_i, e_j) constitui uma ligação Tomek se não existe um exemplo e_k, tal que a distância $d(e_k, e_i) < d(e_i, e_j)$ ou $d(e_k, e_j) < d(e_i, e_j)$.

Se dois exemplos (e_i, e_j) formam uma ligação *Tomek*, então, $e_i$ou $e_j$são exemplos próximos à borda de decisão, ou um desses exemplos é possivelmente ruído. No uso de ligações *Tomek* para o balanceamento de conjuntos de dados, apenas os exemplos da classe majoritária que possuem ligações *Tomek* são removidos (BATISTA et al., 2004).

O uso de técnica de limpeza como Tomek Link após o uso do Smote está no fato que o Smote não considera a vizinhança em relação à outra classe durante o processo de interpolação. Assim ele tende a aumentar o grau de sobreposição para elementos ruidosos e próximos à borda. Com o objetivo de melhorar a definição dos agrupamentos de dados no conjunto de treinamento, em (BATISTA

et al., 2004) foi proposta a aplicação do método para a identificação de ligações *Tomek* após a aplicação do método *Smote*. Diferentemente da aplicação de ligações *Tomek* como método de *undersampling*, nesse caso (como o conjunto de dados foi previamente balanceado com exemplos "sintéticos") são removidos exemplos de ambas as classes (BATISTA et al., 2004).

4.3.3 Smote + ENN

O método ENN (*Edited Nearest Neighbors*) foi criado por Wilson em 1972 (GARCÍA, 2008) e consiste na eliminação de todos os objetos que são mal classificados por seus k-vizinhos mais próximos (k-NN). Este processo baseia-se basicamente em:

1. Para cada ponto $x \in Base\ de\ Dados$
    a. Descobrem-se quem são os k-vizinhos mais próximos de x.
    b. Verifica-se qual a classificação de todos dos k-vizinhos de x.
    c. Se a classificação de x for diferente da classificação de todos os seus k-vizinhos, este ponto é eliminado.

A aplicação do método ENN é feita após a aplicação do método Smote. A motivação é similar a do método Smote + *Tomek Links*. Entretanto, ENN tende a remover mais exemplos do que as ligações Tomek, promovendo uma maior limpeza no conjunto de dados. Similarmente ao método Smote + *Tomek Links*, ENN é utilizado para remover exemplos de ambas as classes. Então qualquer exemplo cuja classificação seja diferente de seus três vizinhos mais próximos é eliminado.

4.3.4 BED

O *Boundary Elimination and Domination* (BED) tem como principal idéia incrementar a representatividade da classe minoritária do conjunto de treinamento através do pré-processamento dos dados.

O pré-processamento dos dados é feito removendo (*undersampling*) dados ruidosos da classe majoritária e gerando (*oversampling*) novos exemplos da classe minoritária. Todo o processo de re-amostragem é guiado pela informação da densidade em torno dos exemplos de treinamento. Essa informação torna possível para o algoritmo BED identificar exemplos isolados ou que pertencem à região de sobreposição. O processo de eliminação e síntese de exemplos dessas regiões melhora a representatividade dos dados para que um classificador possa estimar uma melhor superfície de separação.

Para obter a informação sobre a densidade, BED define uma taxa de credibilidade para cada exemplo do conjunto de treinamento. Esta informação é calculada a partir dos *k* vizinhos mais próximos (com base na distância euclidiana) do exemplo avaliadas de acordo com a seguinte equação,

$$cs(x_i^c) = \frac{nC^c}{nC^+ + nC^-} \tag{11}$$

onde $cs(x_i^c)$ corresponde ao i-ésimo exemplo de treinamento da classe C^c. O símbolo $c = \{-, +\}$ indica se o exemplo pertence à classe negativa (majoritária) ou à classe positiva (minoritária), respectivamente. O valor de nC^+ corresponde ao número de vizinhos positivos de x_i^c.

Para a classe majoritária (x_i^-), o índice de credibilidade $cs(x_i^-)$ é usada para encontrar exemplos ruidosos que ocupam a área de sobreposição e exemplos isolados ao longo da região da classe minoritária. BED estabelece regras para detectar e eliminar esses exemplos. Essa regra depende do parâmetro *maxtol* e é dada por:

Se $cs(x_i^-) < maxtol \rightarrow x_i^-$ é eliminado;

Se $cs(x_i^-) \geq maxtol \rightarrow x_i^-$ não é eliminado.

O valor do parâmetro *maxtol* ($0 \leq maxtol \leq 1$) pode ser definido pelo usuário e irá determinar a intensidade do *undersampling*. Quanto maior o valor do *maxtol* mais exemplos da classe majoritária são eliminados. Nos casos de alto índice de desbalanceamento das classes, valores próximos de 1 são sugeridos, uma vez que é provável que um grande número de exemplos negativos (majoritária) ao longo das áreas pertencentes à classe positiva (minoritária). Quando o desequilíbrio é mais suave, valores menores são sugeridos.

Para os exemplos da classe minoritária (x_i^+), a taxa de credibilidade $cs(x_i^+)$ é usada para sintetizar novos exemplos. Se $cs(x_i^+)$ é considerado válido pelo BED, um novo exemplo α^+ é criado como segue:

1. Para cada atributo contínuo *m*, o seu valor é calculado com base na média dos seus *m*-valores dos seus vizinhos mais próximos positivos nC^+ e o exemplo x_i^+,

$$\alpha^+(m) = \frac{\sum_{j=1}^{nC^+} x_j^+(m) + x_i^+(m)}{nC^+ + 1} \quad (12)$$

2. Para cada atributo nominal *p* assume-se o valor mais freqüentemente observado entre os *p*-valores dos seus vizinhos mais próximos positivos nC^+ e o exemplo x_i^+.

Entretanto, se $cs(x_i^+)$ não é considerado válido, x_i^+ é considerado um ruído ou um exemplo isolado e, um novo exemplo não é criado ao redor dele. A regra que avalia o exemplo positivo é baseada no parâmetro *mintol* e é dado por:

Se $cs(x_i^+) < mintol \rightarrow \alpha^+$ não é criado;

Se $cs(x_i^+) \geq mintol \rightarrow \alpha^+$é criado.

O parâmetro *mintol* define o grau de validade dos exemplos positivos no espaço de entrada e similar ao parâmetro m*axtol* pode ter valores entre 0 e 1. Quanto menor o valor *mintol* maior a probabilidade de um exemplo positivo na área de sobreposição ser considerado válido e utilizado para gerar um novo exemplo sintético α^+. No entanto valores do *mintol* não podem ser muito

próximos de 0 para garantir que exemplos isolados pertencentes à classe positiva nC^+ não gerem novos exemplos. O ajuste do *mintol* também deve ser feito pelo usuário de acordo com o nível de desequilíbrio e sobreposição das classes.

A cada iteração o algoritmo BED, um novo conjunto de treinamento é analisado. O algoritmo termina quando as classes se encontram balanceadas (CASTRO, 2009).

4.4 ADASYN

Motivados pelo sucesso recente dos algoritmos de abordagens sintéticas incluindo o Smote, SmoteBoost e DataBoost-IM, o método ADASYN (*Adaptive Synthetic Sampling*) faz uma adaptação para tentar facilitar a aprendizagem a partir do conjunto de dados desequilibrado (HE, 2008).

O algoritmo proposto é descrito abaixo:

Entradas:

Conjunto de treinamento D_{tr} com *m* exemplos $\{x_i, y_i\} = i, \dots, m$ onde x_i é uma instância do vetor *n* dimensional do espaço X e $y_i \in Y = \{-1, 1\}$ é a classe que identifica o rótulo de x_i. Define como m_l e m_s o número de exemplos da classe majoritária e minoritária respectivamente. Então, $m_s \leq m_l$ e $m_s + m_l = m$.

Procedimento:

(1) Calcula-se o grau de desbalanceamento da classe:

$$d = \frac{m_s}{m_l} \quad (13)$$

onde $d \in (0,1]$.

(2) Se $d < d_{th}$ então (d_{th} é um limiar predefinido para o máximo grau de tolerância de relação desequilíbrio classe)

(a) Calcula-se o número de dados que precisam ser gerados para a classe minoritária

$$G = (m_l - m_s) \times \beta \quad (14)$$

onde $\beta \in [0,1]$ é um parâmetro usado para especificar o grau desejado de equilíbrio após a geração de dados sintéticos. $\beta = 1$ representa um conjunto de dados totalmente equilibrado após a geração dos dados.

(b) Para cada exemplo $x_i \in$ a classe minoritária, encontra-se os k vizinhos mais próximos baseados na distância Euclidiana e também se calcula a taxa r_i definida como:

$$r_i = \frac{\Delta_i}{k}, \quad i = 1, \dots, m_s \quad (15)$$

onde Δ_i é o número de exemplos dos k vizinhos mais próximos de x_i que pertencem à classe majoritária, então $r_i \in [0,1]$;

(c) Normaliza-se r_i de acordo com $\check{r}_i = r_i / \sum_{i=1}^{m_s} r_i$, de modo que a $\check{r}_i$ é uma distribuição de densidade ($\sum_i \check{r}_i = 1$).

(d) Calcula-se o número de dados sintéticos precisam ser gerados para cada exemplo da classe minoritária x_i:

$$g_i = \check{r}_i \times G \quad (16)$$

onde G é o número total de exemplos sintéticos que precisam ser gerados para a classe minoritária definido em (14).

(e) Para cada exemplo da classe minoritária x_i gera-se g_i exemplos sintéticos de acordo com os seguintes passos:

Laço de 1 até g_i:

(i) Aleatoriamente escolhe um dos exemplos minoritários, x_{zi}, dos k vizinhos mais próximos do exemplo x_i.

(ii) Gera-se um exemplo sintético:

$$s_i = x_i + (x_{zi} - x_i) \times \gamma \quad (17)$$

onde $(x_{zi} - x_i)$ é o vetor diferença em um espaço n dimensional e γ é um número randômico: $\gamma \in [0,1]$.

Fim do laço

A idéia principal do algoritmo ADASYN é usar a distribuição de densidade como um critério para decidir automaticamente o número de dados sintéticos que precisam ser gerados para cada exemplo da classe minoritária.

Fisicamente, ř$_i$ é uma medida da distribuição de pesos de classe minoritária para diferentes exemplos de acordo com seu nível de dificuldade na aprendizagem.

O conjunto de dados após passar pelo algoritmo ADASYN não só irá proporcionar uma representação equilibrada da distribuição dos dados (de acordo com o nível desejado de equilíbrio definido pelo coeficiente β), mas também vai forçar o algoritmo de aprendizagem para focar sobre as regiões de difícil aprendizado (HE, 2008).

A implementação deste método no trabalho propôs uma modificação do mesmo após encontrar um problema em seu algoritmo. No passo 2(b) onde é calculado o a taxa r_i, diz que Δ_i é o número de vizinhos mais próximos de x_i pertencentes à classe majoritária. Isto provoca uma criação de exemplos sintéticos próximos à x_i mesmo quando este é considerado um ruído, o que não é aconselhável. A modificação foi feita no cálculo de r_i onde é considerado que Δ_i é o número de vizinhos mais próximos de x_i que pertence à classe minoritária.

4.5 Bases de Dados Utilizadas

As bases de dados utilizadas no trabalho foram retiradas do repositório de dados do Centro de Aprendizado de Máquinas e Sistemas Inteligentes da Universidade da Califórnia, em Irvine, onde todas foram geradas por dados reais de pesquisa (UC Irvine Machine Learning Repository).

A seguir, é feita uma pequena descrição das bases selecionadas.

- *Pima Diabetes*: Foram feitas algumas restrições para a seleção de pacientes diabéticos. As candidatas eram do sexo feminino com, pelo menos, 21 anos e com herança genética da tribo indígena Pima. A tarefa é tentar predizer pessoas que têm tendências hereditárias ao Diabetes.
- *Breast Cancer* (WPBC): Cada registro representa os dados do acompanhamento de um caso de câncer de mama. Estes são pacientes consecutivamente atendidos pelo Dr. Wolberg desde 1984 e incluem somente os casos que exibem o câncer de mama invasivo e sem evidência de metástases à distância no momento do diagnóstico.
- *SPECTF Heart*: O conjunto de dados descreve o diagnóstico de doença cardíaca a partir da análise de imagens obtidas pelo método *Single Proton Emission Computed Tomography (SPECT)*. Cada um dos pacientes é classificado em duas categorias: normal e anormal.
- *Image Segmentation*: Os casos foram sorteados a partir de um banco de dados, de sete imagens ao ar livre. As imagens foram segmentadas a mão para criar uma classificação para cada pixel.
- *Glass:* Uma base de dados feita para estudo da classificação dos tipos de vidro motivado por investigações criminais. Na cena do crime, o vidro pode ser usado como prova, se ele for identificado corretamente.

- *Car*: Este banco de dados contém exemplos de informações coletadas por entrevistas feitas com motoristas sobre alguns requisitos fundamentais na avaliação de um carro: valor de compra, manutenção, número de portas, número de pessoas, tamanho do bagageiro e segurança.
- *Yeast*: Base de dados para tentar prever a localização da proteína de fermentos.
- *Abalone*: Base de dados com informações diversas sobre vários exemplos de abalone (um molusco) nas quais se tenta predizer a sua idade, o que é um trabalho demorado feito por análise de microscópio.

As características de cada base se encontram na tabela abaixo:

Tabela 3 Características das bases de dados utilizadas

Nome da Base de Dados	Nº de Atributos	Atributos Positivos	Atributos Negativos	Nº Total de Atributos	Razão de Desbalanço	% para Treinamento	% para Teste	Nº de FoldsCV [2]
Pima Diabetes	8	268	500	768	0.349	70	30	7
Breast Cancer	33	47	151	198	0.237	70	30	7
SPECTF Heart	44	55	212	267	0.206	70	30	7
Image Segmentation	19	30	180	210	0.143	70	30	7
Glass	10	29	185	214	0.136	70	30	7
Sick Euthyroid	24	238	1762	2000	0.119	70	30	7
Car	6	69	1659	1728	0.040	70	30	7
Yeast	8	51	1433	1484	0.034	70	30	7
Abalone	8	32	4145	4177	0.008	70	30	7

4.6 Método para Seleção dos Parâmetros Ótimos

[2] Número de *Folds* usados para *Cross-Validation*

Todos os experimentos foram conduzidos a partir do seguinte procedimento, que usa busca em *grid* (GESTEL, 2004) e *7-fold cross-validation* para a escolha dos parâmetros ótimos do classificador/algoritmo de aprendizado. Metodologia similar foi usada em (WU, 2005).

1. Divida o conjunto de dados em subconjunto de treinamento/validação (2/3) e subconjunto de teste (1/3).
2. Iniciando de $\mathrm{i} = 0$,
   a. Execute *7-fold cross-validation* para cada combinação de parâmetros obtida a partir de um conjunto de parâmetros candidatos.
   b. Selecione a combinação ótima de parâmetros observando o melhor desempenho médio estimado a partir dos subconjuntos (*folds*) de validação.
   c. Se $i = i_{max}$, vá ao passo 3. Senão, faça $\mathrm{i} = \mathrm{i} + 1$, construa um grid localmente refinado ao redor dos parâmetros ótimos selecionados e retorne ao passo 2(a).
3. Treine o classificador usando todo o subconjunto de treinamento/validação com a combinação ótima de parâmetros obtida a partir da busca em grid.
4. Estime o desempenho do classificador usando o subconjunto de teste independente e armazene o resultado.

Todos os subconjuntos de treinamento, validação e teste foram obtidos de forma estratificada, garantindo a mesma razão entre exemplos positivos e negativos em cada um deles. Para cada algoritmo, esse procedimento foi executado 30 vezes e o desempenho médio em relação aos subconjuntos de teste foi calculado usando métricas extraídas da matriz de confusão e comumente

usadas na literatura para avaliar o desempenho de classificadores em aplicações desbalanceadas. São elas:

- G-Mean
- AUC
- Curvas ROC

4.7 Decisões de Projetos

Os parâmetros de cada método descrito na seção 4.4 utilizados no trabalho estão descrito abaixo na tabela 4.

Tabela 4 Parâmetros de cada método utilizados no trabalho para cada base de dados

Decisões de Projeto								
Método	Base de Dados							
ADASYN	Abalone	Breast Cancer	Car	Diabetes	Euthyroid	Glass	Heart	Segmentation
K	5	5	5	5	5	5	5	5
β	1	1	1	1	1	1	1	1
Dth	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75
BED	Abalone	Breast Cancer	Car	Diabetes	Euthyroid	Glass	Heart	Segmentation
K	5	5	5	5	5	5	5	5
mintol	0.10	0.50	0.50	0.50	0.50	0.50	0.50	0.50
maxtol	0.90	0.50	0.50	0.50	0.50	0.50	0.50	0.50
Smote e Variações	Abalone	Breast Cancer	Car	Diabetes	Euthyroid	Glass	Heart	Segmentation
K	5	5	5	5	5	5	5	5
β	1	1	1	1	1	1	1	1

O trabalho foi desenvolvido no ambiente Windows Vista 32 bits, em uma máquina com processador AMD Turion™ X2 *Dual-Core Mobile* modelo TL-60 (2000 MHz), com 2 GB de memória RAM 667MHz. Para implementação dos métodos foi utilizado o software MatLab® R2009a na versão 7.8.0.347.

4.8 Conclusões do capítulo

Neste capítulo foi apresentada a metodologia dos experimentos realizados as bases de dados reais utilizadas nos treinamentos das redes neurais artificiais, os métodos de balanceamento artificial utilizados para comparação de desempenho usando a análise ROC e as decisões de projeto.

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos pelo trabalho com a apresentação de tabelas e gráficos ROC. Ao final é feita uma discussão sobre o desempenho alcançado pelos algoritmos para bases com diferentes graus de desbalanceamento.

5.1 Experimentos com Bases do Repositório UCI

A Tabela 5 e a Tabela 6 comparam, respectivamente, os valores de *G-Mean* e *AUC* obtidas pelos algoritmos quando aplicados aos subconjuntos de teste para cada base de dados contendo diferentes graus de desbalanceamentos. Para cada métrica a média e o desvio padrão foram calculados a partir de trinta execuções com diferentes subconjuntos de treinamento/validação e teste, conforme descrito na seção 4.6. Os melhores resultados estão marcados em negrito.

Tabela 5 Resultados da média do G-Mean (em %) para diferentes Métodos de Balanceamento Artificial

Média do G-mean						
Base de Dados	Método					
	Original	Smote	Smote + ENN	Smote + Tomek Link	BED	ADASYN
Abalone	0 ± 00	23.8 ± 18	**30.4 ± 18**	25.3 ± 18	08.1 ± 13	28.4 ± 17
Breast Cancer	62.6 ± 01	61.9 ± 07	61.4 ± 06	**64.1 ± 08**	60.7 ± 08	60.3 ± 09
Car	93.3 ± 05	93.5 ± 06	**95.1 ± 04**	94.6 ± 05	93.6 ± 07	94.6 ± 05
Diabetes	70.6 ± 04	68.0 ± 03	66.2 ± 03	69.0 ± 03	68.4 ± 02	**71.6 ± 04**
Euthyroid	87.8 ± 04	87.7 ± 03	**88.8 ± 02**	87.1 ± 02	88.3 ± 02	87.7 ± 02
Glass	89.3 ± 06	89.8 ± 06	90.9 ± 07	89.4 ± 07	**91.3 ± 06**	89.2 ± 08
Heart	55.8 ± 15	58.7 ± 08	61.8 ± 07	61.9 ± 07	**70.0 ± 07**	61.4 ± 07
Segmentation	94.1 ± 18	96.6 ± 03	96.4 ± 03	96.6 ± 02	**96.9 ± 02**	96.6 ± 02
Yeast	32.4 ± 25	64.5 ± 10	**68.4 ± 12**	62.8 ± 09	62.3 ± 11	67.0 ± 10

Tabela 6 Resultados da média da AUC (em %) para Diferentes Métodos de Balanceamento Artificial

Média da AUC						
Base de Dados	Método					
	Original	Smote	Smote + ENN	Smote + Tomek Link	BED	ADASYN
Abalone	**70.9 ± 01**	64.4 ± 09	68.2 ± 11	67.8 ± 10	62.3 ± 07	66.2 ± 07
Breast Cancer	72.5 ± 07	71.4 ± 06	67.3 ± 07	**73.4 ± 06**	69.7 ± 06	70.9 ± 06
Car	99.2 ± 01	99.2 ± 01	99.4 ± 01	99.5 ± 00	99.2 ± 02	**99.6 ± 00**
Diabetes	**80.2 ± 05**	75.7 ± 03	73.5 ± 03	79.8 ± 03	75.7 ± 03	76.5 ± 04
Euthyroid	94.2 ± 03	94.7 ± 01	**95.3 ± 02**	94.8 ± 02	95.1 ± 01	94.7 ± 02
Glass	94.7 ± 05	95.7 ± 05	95.7 ± 04	95.2 ± 06	**96.0 ± 04**	95.5 ± 05
Heart	72.8 ± 07	72.0 ± 07	72.7 ± 06	73.8 ± 06	**78.1 ± 06**	75.3 ± 05
Segmentation	96.7 ± 13	99.2 ± 01	99.1 ± 02	98.8 ± 01	**99.4 ± 01**	99.2 ± 01
Yeast	81.3 ± 10	84.0 ± 05	**85.8 ± 05**	83.7 ± 06	83.4 ± 06	84.0 ± 05

Curvas ROC médias foram estimadas a partir dos subconjuntos de teste para cada amostra de dados. As Figuras 11, 12, 13 e 14 ilustram, respectivamente, as curvas ROC médias (entre as faixas [0.0, 1.0] para *FPrate* e [0.0, 1.0] para

TPrate) obtidas para três conjuntos apresentando as seguintes razões de desbalanceamento, isto é, razões de 0.008 (1000 negativos : 8 positivos) para a base *Abalone*, 0.034 (1000 negativos : 34 positivos) para a base *Yeast*, 0.119 (1000 negativos : 119 positivos) para a base *Euthyroid* e 0.216 (1000 negativos : 216 positivos) para a base *Heart*.

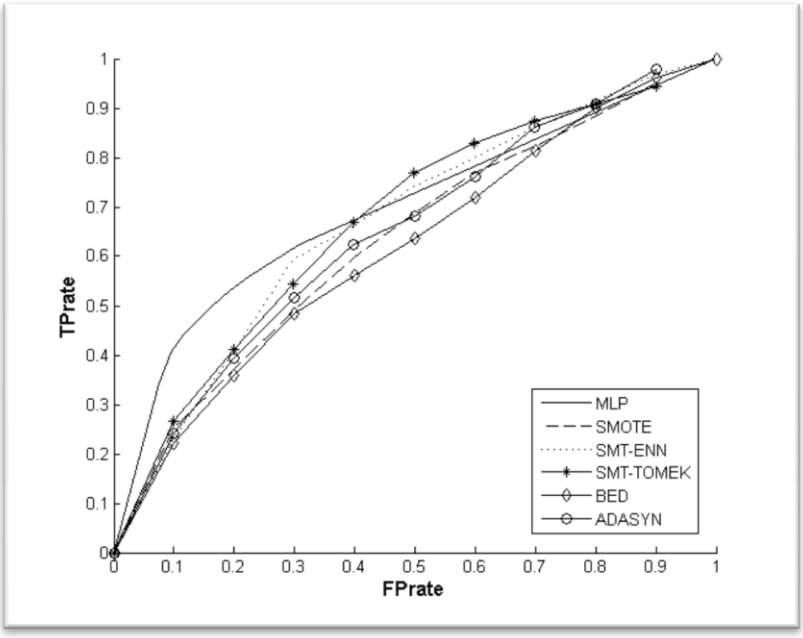

Figura 11 Curvas ROC média estimada a partir dos subconjuntos de teste para a base de dados *Abalone* com diferentes algoritmos de balanceamento artificial.

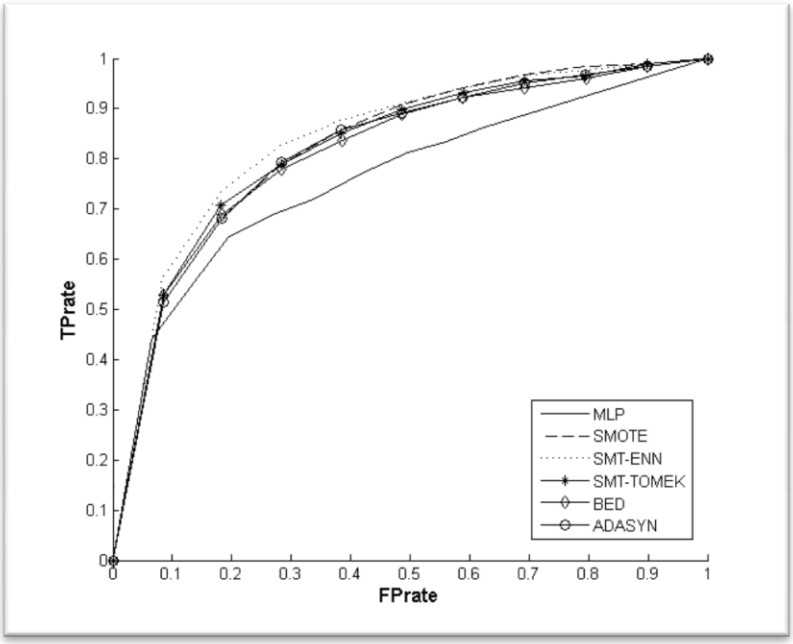

Figura 12 Curvas ROC média estimada a partir dos subconjuntos de teste para a base de dados *Yeast* com diferentes algoritmos de balanceamento artificial

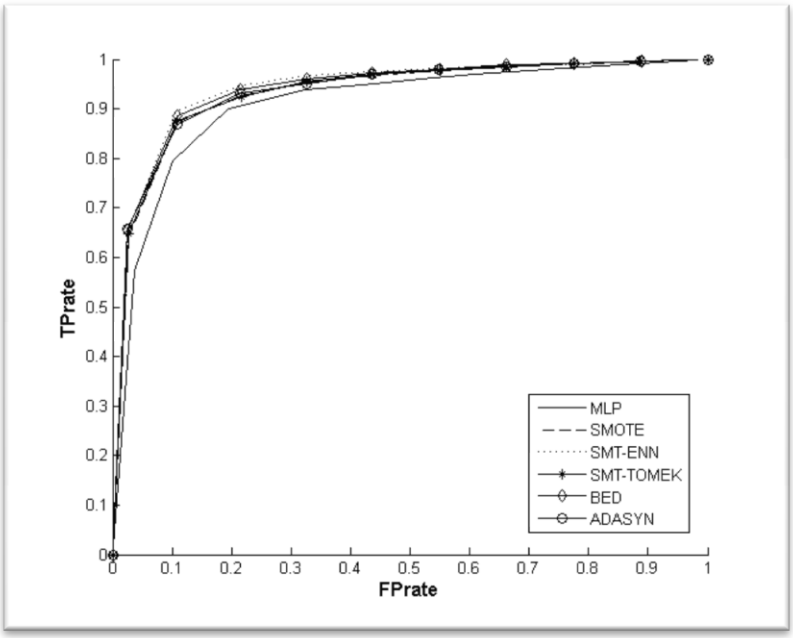

Figura 13 Curvas ROC média estimada a partir dos subconjuntos de teste para a base de dados *Euthyroid* com diferentes algoritmos de balanceamento artificial

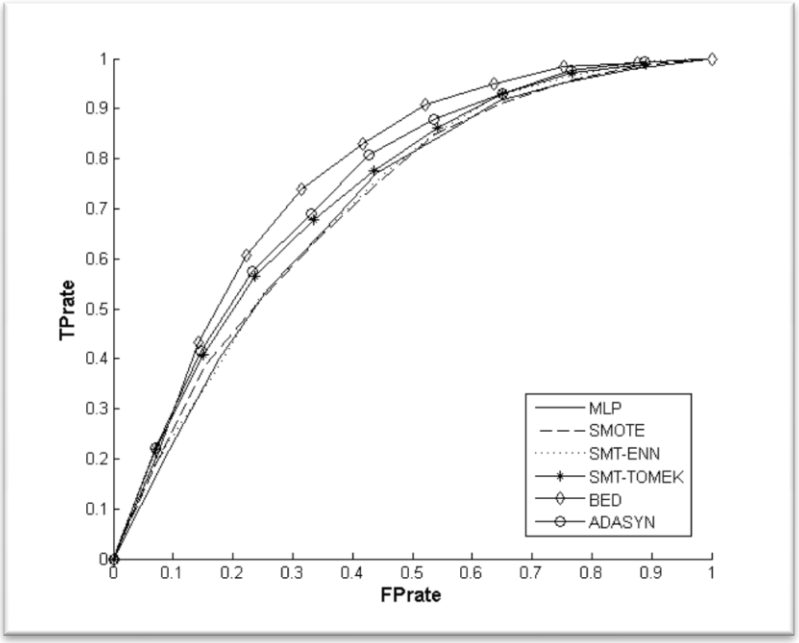

Figura 14 Curvas ROC média estimada a partir dos subconjuntos de teste para a base de dados *Heart* com diferentes algoritmos de balanceamento artificial

5.2 Discussões dos Resultados

O trabalho consistia resumidamente em estudar o efeito do desbalanceamento do conjunto de treinamento em aprendizado supervisionado de redes neurais artificiais e a comparação de alguns métodos para a realização da reamostragem deste conjunto de treinamento. Nesta seção é feita uma discussão dos resultados obtidos e mostrados na Seção 5.1.

5.2.1 Processo de Balanceamento

Um dos objetivos do trabalho era avaliar como que se comportava a desempenho de redes neurais com um conjunto de treinamento desbalanceado e após o seu balanceamento com alguns métodos de reamostragem.

A partir dos resultados obtidos descrito na Tabela 5 e na Tabela 6 é possível concluir que para todas as bases de dados o processo de reamostragem não provocou uma queda de desempenho em relação ao treinamento utilizando a base de dados original.

É possível perceber que o processo de balanceamento de algumas bases de dados não obteve um aumento muito significativo nas métricas de *G-Mean* e *AUC* como aconteceu nas bases *Car*, *Euthyroid* (Figura 13), *Glass* e *Segmentation.* Isso pode ter ocorrido devido às classes serem mais separáveis no espaço de entrada e que apresentarem um nível reduzido de sobreposição, sendo assim o desbalanceamento não é tão prejudicial (BATISTA et al., 2004).

Também é possível constatar que o processo de balanceamento de bases de dados como *Abalone* (Figura 11), *Heart* (Figura 13) e *Yeast* (Figura 12) obtiveram um acréscimo de desempenho significativo para as duas métricas, isso provavelmente ocorre, pois, essas bases possuem níveis elevados de desbalanceamento e/ou sobreposição e pode ser comprovado analisando as curvas ROC dessas três bases de dados.

5.2.2 Comparação dos Métodos de Balanceamento

Outro objetivo do trabalho era comparar os diferentes métodos de reamostragem e avaliar seu comportamento para as diferentes bases de dados.

É possível perceber que, para a maioria das bases de dados, o método Smote obteve um desempenho abaixo de suas variações, como o Smote + ENN e o Smote + Tomek Link, mostrado nas curvas ROC da Figura 11 e da Figura 14. Com isso é possível concluir que um método adicional, para a remoção de dados

ruidosos e sobreposição de dados, combinado com o método Smote proporciona uma melhor desempenho para o classificador.

Como é possível verificar nas curvas ROC do método Smote + ENN mostrado na Figura 11, Figura 12 e Figura 13, este conseguiu aumentar o desempenho do classificador para as bases de dados que apresentam uma maior taxa de desbalanceamento, ou seja, *Abalone, Yeast, Car e Euthyroid* em relação aos outro métodos de reamostragem.

Levando em consideração a Tabela 5 (média do G-Mean) o método que se consagrou mais vitorioso em relação aos outros foi o Smote + ENN que conseguiu superar a média dos outros em quatro bases seguido do método BED (três bases), Smote + Tomek Link (uma base), ADASYN (uma base) e Smote (nenhuma base).

Em relação à maximização da AUC, o método BED obteve maiores resultados em três bases (Figura 14), seguido do método Smote + ENN e do conjunto de treinamento original, ambos com duas bases e ADASYN e Smote + Tomek Link com uma base cada um.

5.3 Conclusões do Capítulo

Neste capítulo foram apresentados os resultados obtidos, ilustrados em tabelas e curvas ROC, posteriormente uma discussão destes resultados em relação ao processo de balanceamento e uma comparação dos métodos implementados como conclusão do trabalho.

6 CONCLUSÕES

Neste trabalho foram feitos experimentos com algoritmos de balanceamento artificial do conjunto de treinamento para o aprendizado de redes neurais artificiais M*ultilayer Perceptron*, utilizando bases de dados reais retiradas de repositórios na internet. Após o estudo e implementação de cinco métodos de reamostragem, (Smote, Smote + ENN, Smote + Tomek Link, BED e ADASYN) uma comparação dos resultados obtidos utilizando métricas extraídas da análise ROC foi feita, mostrando tabelas comparativas e curvas ROC a fim de ilustrar o desempenho obtido pelos classificadores após serem balanceados por cada um dos métodos selecionados.

Os resultados dos experimentos mostram que na maioria dos casos os métodos de balanceamento são eficazes e melhoram o desempenho das RNAs. Melhorias mais expressivas foram obtidas para as bases de dados com maiores razões de desbalanceamento e sobreposição. Dentre os métodos testados, aqueles que obtiveram melhores resultados foram: Smote + ENN e BED.

A maioria dos estudos sobre métodos de balanceamento artificial na literatura de aprendizado de máquina têm sido feitos para classificadores baseados em árvores de decisão. Acredita-se que os resultados obtidos nesse trabalho podem servir como base de comparação (*benchmark*) para outros estudos sobre o problema de classes desbalanceadas no contexto de RNA.

6.1 Trabalhos Futuros

Como trabalhos futuros, propõem-se:

- Aplicação de outros métodos de balanceamento artificial;
- Análise estatística da significância dos resultados obtidos visando melhorar as conclusões sobre as vatangens/desvatagens de um método em relação a outros.

7 REFERÊNCIAS BIBLIOGRÁFICAS

Bradley, A. P. **"The use of the area under the ROC curve in the evaluation of machine learning algorithms."** *Pattern Recognition*, 1997: 1145-1159.

Braga, Antônio de Pádua, André Leon de Carvalho Ponce, e Teresa Ludemir Bernarda. ***Redes Neurais Artificiais: Teorias e Aplicações.*** Rio de Janeiro: LTC, 2007.

Castro, Fernando César C. de Castro e Maria Cristina F. de. Porto Alegre, RS, 1995.

Cristiano Leite Castro, Mateus Araujo Carvalho, and Antônio Padua Braga. **"An Improved Algorithm for SVMs Classification."** *Springer-Verlag Berlin Heidelberg 2009*, 2009: 108-118.

D. E. Rumelhart, G. E. Hinton and R. J. Willians. **"Nature."** ***Learning representations by back-propagations errors***, 1986: 323:533-536.

Gustavo E. A. P. A. Batista, Ronando C. Prati, Maria Carolina Monard. **"A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data."** *SIGKDD Explorations Newsletter*, 2004: 20-29.

Haibo He, Edwardo A. Garcia. **"Leraning from Imbalanced Data."** *IEEE*, 2009.

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. "**ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced."** *International Joint Conference on Neural Networks (IJCNN 2008)*, 2008: 1322-1329.

Hand, D. J. ***Construction and Assessment of Classification Rules.*** Chichester: John Wiley & Sons, 1997.

Haykin, Simon. ***Redes Neurais - Prinícipios e Prática.*** Porto Alegre: Bookman, 2007.

Hebb, D. O. ***The Organization of Behavior***. Wiley, 1949.

Hopfield, J. J. **"Neural networks and physical systems with energent collective properties."** *Proc. Nat. Acad. Sci.*, 1982: 79:2554-2558.

Maria Carolina Monard, Gustavo E.A.P.A. Batista. **"Learning whit Skewed Class Distributions."** *Cadernos de Computação XX*, 2003.

Mendel, J. M. and McLaren, R.W. **"Reinforcement-leaning Control and Pattern recognition system."** In: *Adaptative, learning, and pattern recognition systems; theory and application*, 287-318. New York: Academic Press, 1970.

Milton García Borroto, Yenny Villuendas Rey, Miguel Ángel Medina Pérez, Juliett Martínez López, José Ruiz Shulcloper. ***Selección y construcción de objetos para el mejoramiento de un clasificador supervisado: un análisis crítico.*** 1ª Edição. Edição: Lic. Margarita Ilisástigui Avilés. Havana: Centav, 2008.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. **"SMOTE: Synthetic Minority Over-Sampling Technique."** *Artificial Intelligence Research*, 2002: 321-357.

Papert, M. Minsky and S. *Perceptrons:* ***An introduction to computacional geometry.*** Massachusetts: MIT Press, 1969.

Pitts, W. S. McCulloch and W. **"Bulletin of Mathematical Biophysics."** ***A logical calculus of the ideas immanent in neurvous activity***, 1943: 5:115-133.

PRATI, R. C., G. E. A. P. A. BATISTA, e M. C. MONARD. **"Uma Experiência no Balanceamento Artificial de Conjuntos de Dados para Aprendizado com Classes Desbalanceadas utilizando Análise ROC."** 2003.

Provost, G.M. Weiss and F. **"The Effect of Class Distribution on Classifier Learning: An Empirical Study."** Rutgers University, 2001.

Provost, T. Fawcett and F. J. **"Adaptative Fraud Detection."** *Data Mining and Knowledge Discovery*, 1997: 291-316.

R. B. Rao, S. Krishnan, and R. S. Niculescu. **"Data mining for Improved Cardiac Care."** *ACM SIGKDD Explorations Newsletter* 6, n. 14 (2006): 3-10.

R. C. Prati, G.E.A.P.A. Batista e M. C. Monard. "**Curvas ROC para avaliação de Classificadores."** *IEEE Latin America Transactions*, 2008.

Rosenblatt, F. "**The Perceptron: A Porbabilistic model for information storage and organization in the brain."** *Psychol. Rev.*, 1958: 65:386-408.

S. J. Stolfo, D. W. Fan, W. Lee, A. L. Prodromidis and P. K. Chan. **"Credit Card Fraud Detection Using Meta-Learning."** *Workshop on AI Methods in Fraud and Risk Management*, 1997.

Silva, Felipe Castro da. **"Análise ROC."** 2006.

Tomek, I. **"An Experiment with the Edited Nearest-Neighborg Rule."** *IEEE Transactions on System, Man and Communications*, June de 1976: 6:448-452.

UC Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml/index.html (acesso em 5 de 10 de 2010).

Van Gestel, T., Suykens, J.A.K., Baesens, B., Viaene, S., Vanthienen,. **"Benchmarking least squares support vector machine classifiers."** *Mach. Learn* 54 (2004): 5-32.

Windrow, B. and Hoff, M. E. ***Adaptative switching circuits.*** Western Eletronic Show and Convention: Institure of Radio Engineers, 1960.

Wu, G. & Chang, E.Y. **"Kba: Kernel boundary alignment considering."** *IEEE Transactions on Knowledge and Data* 17 (2005): 786-795.