UNIVERSIDADE DE SÃO PAULO - USP
Faculdade de Economia, Administração e Contabilidade - FEA
Instituto de Matemática e Estatística - IME
Mestrado Profissionalizante Modelagem Matemática em Finanças

PREVISÃO DE INADIMPLÊNCIA DE TRANSAÇÕES COM CARTÃO DE CRÉDITO

Um Estudo Comparativo

Sandro Sinhorigno

Orientador: Prof. Dr. Renato Vicente

São Paulo
2007

PREVISÃO DE INADIMPLÊNCIA DE TRANSAÇÕES COM CARTÃO DE CRÉDITO

Um Estudo Comparativo

Sandro Sinhorigno

Dissertação apresentada à Faculdade de Economia, Administração e Contabilidade e ao Instituto de Matemática e Estatística da Universidade de São Paulo para obtenção do Título de Mestre.

Orientador: Prof. Dr. Renato Vicente

São Paulo
2007

Dedico este trabalho à minha esposa Vanessa e aos meus pais, pois todas as conquistas obtidas em minha vida jamais seriam possíveis sem eles.

AGRADECIMENTOS

À minha esposa Vanessa, sempre presente, pelo carinho, apoio e dedicação.

Aos meus pais Lincoln e Tita por toda ajuda e lições de vida que muito me ensinaram.

Às minhas irmãs Valéria e Cláudia pelo eterno incentivo aos estudos.

Ao Professor Renato Vicente, pela orientação, condução e acompanhamento do trabalho.

Ao Professor Henrique von Dreifus e aos colegas Regina, Junior e César por todo suporte administrativo.

Aos colegas de turma Fábio, Paulo, Roberta, Bona, Han, Antonio Marcos, Vânia, Yamada, Henrique e Bess por toda ajuda, companheirismo e principalmente amizade.

Ao amigo Eduardo Prado, pela valiosa contribuição profissional e acadêmica prestada nos últimos anos.

Aos colegas de trabalho Daniel, Patrícia e Jean pelas recomendações e experiências compartilhadas.

Ao Banco Itaú, por toda a confiança e incentivo prestados.

A Deus que, em toda a minha vida, tem me dado forças para superar os obstáculos.

"A satisfação está no esforço feito para alcançar o objetivo e não somente em tê-lo alcançado."

Gandhi

ABSTRACT

The purpose of this work is to develop a behavior scoring model to recognize and predict which customers will be "good or bad payers". It proposes a comparative analysis between bankruptcy prediction methods to evaluate credit cards transaction risks. The model establishes a safe criteria to determine when the transaction must be approved or not. Bankruptcy risk evaluation is explored in this study comparing logistic regression performance with the others techniques based in machine learning: neural networks and support vector machines.

Keywords: *Credit Decision-Making; Bankruptcy Prediction; Logistic Regression; Neural Networks; Support Vector Machines*

OBJETIVO

O objetivo deste trabalho é desenvolver um modelo comportamental de classificação visando reconhecer e prever quais clientes serão "bons ou maus pagadores". Propõe-se uma análise comparativa entre as técnicas de previsão de inadimplência para avaliação do risco em transações com cartão de crédito. O modelo estabelece um critério seguro para determinarmos quando a transação deverá ser aprovada ou não. Neste estudo, o risco de inadimplência dos clientes é explorado através da comparação do desempenho do modelo regressão logística com outras técnicas baseadas em máquinas de aprendizagem: redes neurais e máquinas de vetores de suporte.

Palavras-Chave: *Tomada de Decisão de Crédito; Previsão de Inadimplência; Regressão Logística; Redes Neurais; Máquinas de Vetores de Suporte*

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Modelos de Classificação de Crédito no Brasil

Com a chegada do Plano Real em meados de 1994, proporcionando um cenário econômico de inflação controlada, as instituições financeiras, principalmente os bancos de varejo, passaram a procurar fontes alternativas para obtenção de suas receitas.

Diante de uma economia reaquecida, a expansão das ofertas de crédito ao consumidor (concessão de empréstimos, financiamentos, cartões de crédito, créditos pessoais, etc.) passou a ser identificada como uma imensa oportunidade para alavancagem de receitas, sejam elas financeiras ou de serviços.

O aumento da demanda de crédito por indivíduos não bancarizados, principalmente os de baixa renda, as aquisições de financeiras como a compra da Losango pelo Bradesco, a da carteira Hipercard pelo Unibanco e a partição do portfólio de cartonistas Credicard entre os acionistas Citibank e Itaú são apenas alguns exemplos de acontecimentos vivenciados pelo sistema financeiro brasileiro nos últimos anos.

Se por um lado tais estratégias trouxeram a possibilidade de altos lucros, por outro lado, também elevaram os níveis de risco de inadimplência nas instituições. Qualquer erro na tomada de decisão de crédito pode significar que em uma única operação ocorra a perda do ganho obtido em dezenas de outras bem sucedidas. Fatos recentes de concordata e falência, envolvendo empresas internacionalmente conhecidas, despertam a preocupação das instituições e de seus acionistas, quanto ao controle do risco sobre estas operações.

Faz-se então necessária a revisão contínua das técnicas de previsão de inadimplência, os chamados modelos de classificação ou escoragem de crédito. O investimento em instrumentos, não somente sofisticados mas principalmente velozes e eficientes, passa a ser considerado como um diferencial competitivo para a instituição.

Instrumentos para avaliação de riscos na concessão do crédito, os chamados modelos de *credit scoring*[1], são aprimorados constantemente em função do crescimento do volume de clientes novos ou desconhecidos da instituição. Por outro lado, os instrumentos comportamentais de avaliação de risco, denominados modelos de *behavior scoring*[1], passam a refletir com mais rapidez e qualidade o comportamento de risco, tanto para os clientes antigos quanto para os novos.

Nos modelos comportamentais é ideal que a decisão ótima de crédito leve em consideração outras variáveis como, por exemplo, o relacionamento do cliente com o banco, risco de imagem para a instituição e principalmente a relação risco/retorno, ou seja, não somente o controle do risco de crédito mas principalmente o resultado financeiro envolvido.

Até meados de 1994, o gerenciamento de crédito no Brasil era realizado através de uma política julgamental de decisão baseada nos cinco *Cs do Crédito (Caráter, Capacidade, Condições, Capital e Colateral)*[2] ou na experiência do analista de crédito, sem a utilização de métodos estatísticos de análise. A aplicação de métodos estatísticos de análise, não somente restringe a subjetividade na decisão, mas também permite agilizar o processo de decisão além de reduzir significativamente os índices de inadimplência.

1.2 Aplicação em Aprovação de Transações com Cartão de Crédito

Atualmente as instituições financeiras, emissoras de cartões de crédito, operam suas transações de crédito e débito em complexos sistemas computacionais. Estes sistemas, conhecidos como *sistemas de autorizações*, são estruturados, em sua grande maioria, através de gigantescas árvores de decisão. À medida em que as transações com cartão são submetidas para aprovação, o sistema de autorizações realiza consultas a outros sistemas tais como o sistema de informações cadastrais e o de controle de faturamento.

No sistema de informações cadastrais, são acessadas informações como os dados bancários do cliente, o tempo de relacionamento, limite de crédito, cartões adicionais vigentes, dentre outras. Quanto ao sistema de faturamento, são consideradas informações referentes a valores utilizados, pagamentos realizados, transações parceladas e atrasos. Os dados

consultados são então combinados com as informações referentes à transação, permitindo-se que o processo de tomada de decisão de crédito seja realizado. A idéia será avaliar se a transação deverá ser aprovada ou não.

Transações de clientes que ainda possuem limite de crédito disponível, as denominadas *transações normais*, são decididas com maior eficiência uma vez que consideram um número menor de informações, geralmente, somente informações de faturamento permitindo identificar quais são os valores utilizados até o momento, quais são os valores de pagamento processados e se há ou não incidência de atraso.

Por outro lado, existem situações em que não há mais limite de crédito disponível. Transações desta nautreza são conhecidas como *transações críticas*, e deverão ser decididas considerando também as informações que retratam o histórico do cliente junto à instituição. Desta maneira, são consideradas informações como tempo de relacionamento, histórico de pagamentos e atrasos, utilização de saques, dentre outras. Em situações como esta, em que a quantidade de informações existentes no processo, bem como as possíveis combinações entre elas é infinitamente grande, tanto a performance de resposta do sistema quanto a qualidade da decisão passam a estar comprometidos. É necessário que a decisão do sistema seja respondida com mínimos tempo de resposta e risco de inadimplência.

Uma possível alternativa para solucionar problemas como este seria desenvolver instrumentos para estimação de riscos de inadimplência, através de técnicas de classificação. Através de modelos de classificação, seria possível identificar transações de risco bem como otimizar o tempo de resposta uma vez que os modelos resultantes permitiriam associar uma classificação (escoragem) para cada transação crítica solicitada e não mais uma combinação de diversas variáveis.

Estes modelos de classificação podem ser desenvolvidos através de metodologias clássicas como a regressão logística, assim como algumas mais mais sofisticas como as redes neurais artificiais e máquinas de vetores de suporte. Desta forma, o objetivo deste estudo é desenvolver estes modelos de classificação utilizando as três metodologias e comparar os resultados obtidos.

Pelo fato de que estamos considerando transações de clientes que já tiveram a concessão do crédito aprovada, desenvolvemos modelos de classificação de crédito comportamentais, ou seja, modelos de manutenção do crédito, os modelos de *behavior scoring*. Para tais, espera-se obter um alto poder preditivo uma vez que estamos considerando a premissa de que os clientes já são conhecidos em suas instituições bem como as informações históricas sobre estes são de fácil acesso, o que nem sempre ocorre com relação às informações necessárias nos modelos de concessão de crédito, os modelos de *credit scoring*.

1.3 Conjunto de Dados para Inferência

O primeiro aspecto a ser considerado antes da seleção dos dados é identificar detalhadamente quais as principais características e particularidades envolvidas no processo de forma que o modelo possa replicar a realidade da maneira mais precisa possível. É de extrema importância conhecer o perfil dos clientes, as características de cada produto, as regulamentações envolvidas, sazonalidades, clusters já existentes, dentre outros aspectos. O contato com aqueles que atuam diretamente com os clientes bem como com os gestores dos produtos, facilita a escolha das variáveis explicativas do processo decisório de crédito, além de auxiliar a interpretação dos resultados obtidos nos modelos.

Considerando que estamos desenvolvendo modelos de classificação para transações com cartão de crédito, foram escolhidas variáveis comportamentais que retratam a história do cliente com a instituição tais como tempo de relacionamento, produtos e créditos contratados, utilização dos limites, histórico de pagamentos e atrasos. Além destas, também foram consideradas informações do negócio, coletadas no momento da solicitação da transação como valor da transação, se à vista, parcelada ou saque, se local ou internacional, qual o tipo de produto, dentre outras.

Desta maneira, foi possível utilizar um total de 16 variáveis de entrada sendo 8 variáveis contínuas como valor da transação, idade da conta em número de meses, valores de limite, saldos, dias em atraso e 8 variáveis discretas categorizadas. Estas permitem identificar a existência de atributos diferenciados por classes, por exemplo, se a transação é local ou internacional, qual o tipo de produto, se o cliente possui conta corrente junto á instituição ou não, dentre outras.

O próximo passo foi definir os critérios para organização dos dados como safra de observação, período histórico a ser considerado bem como o conceito de performance para classificação das transações em adimplentes ou inadimplentes. Como safra de observação consideramos um mês m onde foram coletadas informações referentes aos clientes, ao produto cartão de crédito e às transações. Em seguida, entre os meses $m-6$ a m, foram consideradas as informações comportamentais históricas (positivas e negativas) dos clientes. Finalmente, entre os meses m a $m+6$, identificou-se quaisquer ocorrências de inadimplência por parte dos clientes.

A idéia é que, a partir destas variáveis de entrada, o modelo classifique as transações em adimplentes ou inadimplentes, ou seja, decida se devem ser aprovadas ou não. Para tal, será associada uma escoragem ou pontuação para cada transação solicitada e, em seguida, será determinado um valor limiar de decisão, ao qual conhecemos como *ponto de corte*, onde serão separadas as classes. Desta forma, não será mais necessário realizar a combinação de

todas estas variáveis através do processo em árvore de decisão, o que possivelmente implicará em menor risco de crédito e maior agilidade na decisão.

1.4 Técnicas Utilizadas e Principais Problemas

Os problemas de classificação, também conhecidos como problemas de reconhecimento de padrões, estão presentes não somente em áreas de análise de crédito como também em áreas como processamento de imagens e diagnóstico médico.

Em problemas envolvendo classificação de crédito destaca-se a utilização das técnicas de análise discriminante e regressão logística, pois apresentam bom desempenho em problemas em que os grupos em análise podem ser separados através de uma fronteira linear. Tais técnicas se enquadram na classe de métodos estatísticos multivariados de dependência, uma vez que relacionam um conjunto de variáveis independentes com uma variável dependente categórica.

Entretanto, técnicas baseadas em máquinas de aprendizagem, como as redes neurais artificiais, têm conquistado espaço de destaque em publicações recentes na área de análise de crédito. Parte da explicação se deve ao fato de que tais metodologias possuem excelente capacidade de generalização e eficiência comprovadas em problemas onde as variáveis de entrada e saída possuem relações não lineares.

O problema de distinção, entre clientes bons ou ruins, possui alguns aspectos críticos como o fato de que, sem uma teoria sobre inadimplência, em qualquer seleção de variáveis preditivas que se faça pode ocorrer a desconsideração de variáveis importantes para os modelos. Outro ponto a ser considerado é que as variáveis preditivas podem possuir relações complexas, o que impossibilita avaliar os padrões de inadimplência através de classes linearmente separáveis, ocasionando baixo poder preditivo nos modelos.

Estudos teóricos confirmam a desvantagem em se utilizar a análise discriminante pelo fato de que a metodologia pressupõe que os padrões de inadimplência são linearmente separáveis. Nos problemas encontrados pelas instituições financeiras em análise de crédito, identifica-se que as relações entre as variáveis são essencialmente não-lineares. Sendo assim, não estaremos considerando a técnica de análise discriminante neste estudo. Maiores informações sobre a metodologia podem ser encontrados no trabalho de Altman[3].

1.5 Organização dos Demais Capítulos

O Capítulo 2 desta dissertação apresenta os principais conceitos referentes ao método de regressão logística bem como a metodologia a ser empregada no desenvolvimento do modelo. Em seguida, é realizada a apresentação e discussão dos resultados.

No Capítulo 3 é apresentado o conceito de redes neurais, a estrutura e metodologia utilizadas para a aplicação no problema proposto. Em seguida, são apresentados e discutidos os resultados.

No Capítulo 4 é contextualizada a técnica máquina de vetores de suporte descrevendo os conceitos envolvidos e resultados obtidos através da aplicação da técnica.

A metodologia de desenvolvimento, tratamento das informações, além dos principais critérios para validação e análise de performance de cada um dos modelos, são descritos detalhadamente de acordo com cada tópico.

O Capítulo 5 apresenta um quadro comparativo com os principais resultados e indicadores de desempenho obtidos em cada técnica e fornece os comentários conclusivos.

Para maior detalhamento técnico sobre os experimentos realizados, é apresentado neste trabalho um Apêndice contendo os principais resultados obtidos em cada metodologia.

Capítulo 2

Regressão Logística

A regressão logística tem se constituído em um dos principais métodos de modelagem estatística de dados. Embora conhecida desde os anos 50, somente através de Cox[4] é que a técnica tornou-se popular entre os usuários de estatística sendo que, nos dias de hoje, é considerada a mais utilizada em modelos de classificação de crédito.

A técnica baseia-se na análise de dados com resposta binária, ou seja, que admitem apenas dois resultados. Chamamos de "sucesso"o resultado mais importante da resposta ou aquele que se pretende relacionar com as demais variáveis de interesse. Mesmo em situações em que o resultado de interesse não seja originalmente do tipo binário, vários pesquisadores têm dicotomizado a resposta de modo que a probabilidade de sucesso possa ser modelada através da regressão logística.

Neste estudo, temos um modelo de resposta binária onde obtemos como resposta a decisão a ser tomada para a transação com cartão de crédito, ou seja, se esta será aprovada ou não, considerando as informações do cliente e da própria transação no exato momento da solicitação.

2.1 Aspectos Teóricos

A regressão logística consiste em uma técnica estatística utilizada na separação de dois grupos, que visa obter a probabilidade de que uma observação pertença a um conjunto determinado, em função do comportamento das variáveis independentes[5]. É então obtido

um modelo que relaciona a variável dependente Y às variáveis independentes $X_1, X_2, ..., X_p$ que supostamente influenciam as ocorrências do evento em estudo.

Em situações em que a variável dependente possua caráter não métrico, é necessário que esta seja inserida através do uso de variáveis *dummy*, que assumem valor 0 para indicar a ausência de um atributo e 1 para indicar a presença de um atributo[6].

Em aplicações envolvendo risco de crédito, a metodologia é utilizada para a avaliação da inadimplência de determinado grupo de clientes em situações relativas à concessão de crédito, assumindo que a probabilidade de inadimplência seja logisticamente distribuída, com resultado binomial 0 ou 1. De acordo com Hair et al.[5], para aplicação da regressão logística faz-se necessário conhecer sobre a ocorrência ou não de determinado evento, como por exemplo, situação de inadimplência ou não de um cliente, situação de insolvência ou não de uma empresa.

Sendo assim, utiliza-se apenas dois possíveis estados para a variável dependente ou resposta (0 ou 1), a depender da ocorrência ou não do evento considerado. A partir desse valor dicotômico, a regressão logística calcula a probabilidade desse evento acontecer ou não. O modelo de regressão logística pode então ser escrito da seguinte forma:

$$P(Y = 1) = \frac{1}{1 + e^{-\mu(x)}} \tag{2.1}$$

onde,

$$\mu(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = \beta_0 + \sum_{i=1}^{p} \beta_i X_i \tag{2.2}$$

Analisando o significado da função de distribuição logística no contexto de risco de crédito, temos que a variável dependente consiste na situação de inadimplência relacionada ao cliente, que assumirá valores 0 ou 1, a depender dos dados procederem de um cliente inadimplente ou adimplente, respectivamente. As variáveis independentes representam os fatores que supostamente influenciam a inadimplência: dados pessoais e financeiros, indicadores comportamentais, informações específicas sobre a transação, dentre outros.

A probabilidade de que a transação solicitada provenha de um cliente adimplente é representada por P, que corresponde à probabilidade condicional de Y assumir o valor 1. Os coeficientes $\beta_1, \beta_2, ..., \beta_p$ são estimados a partir do conjunto de dados através do método de máxima verossimilhança e representam medidas das variações na proporção das probabilidades. Dada uma determinada combinação destes coeficientes e variando-se os

valores de X, chega-se a uma curva logística com comportamento probabilístico de uma função sigmoidal conforme apresentado na Figura 2.1.

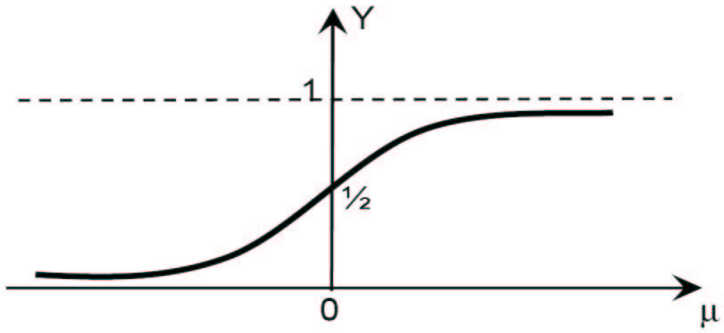


Figura 2.1: Função Característica da Regressão Logística.

Este formato característico permite um alto grau de generalidade, aliado a aspectos interessantes como o fato de que, se $\mu(x) \to +\infty$ então $P(Y = 1) \to 1$ e se $\mu(x) \to -\infty$, então $P(Y = 1) \to 0$. Como podemos estimar diretamente a probabilidade de ocorrência de um evento, a probabilidade de não ocorrência é obtida pela diferença $P(Y = 0) = 1 - P(Y = 1)$.

Na regressão logística, a principal suposição considerada é a de que o logaritmo da razão entre as probabilidades de ocorrência e não ocorrência de um evento seja linear.

$$\frac{P(Y=1)}{P(Y=0)} = e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p} \tag{2.3}$$

e, por consequência,

$$ln\left[\frac{P(Y=1)}{P(Y=0)}\right] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p \tag{2.4}$$

Na utilização do modelo para separação dos dois grupos, consideramos como critério de discriminação o fato de que se $P(Y = 1) > 0,5$, então o grupo é classificado $Y = 1$, caso contrário, classifica-se $Y = 0$. Portanto, o modelo logístico é utilizado para estimar a probabilidade de que um cliente ou operação de crédito seja associado a um grupo de "bons"ou, neste caso, não inadimplentes. A variável dependente Y indica se a transação deverá ser aprovada (quando $Y = 1$) ou recusada (quando $Y = 0$).

2.2 Aplicação e Resultados

Esta seção apresenta em detalhes a metodologia utilizada na coleta e tratamento das amostras de dados. Em seguida são apresentados os resultados e principais indicadores de desempenho obtidos com o modelo de regressão logística.

2.2.1 Seleção e Tratamento dos Dados de Inferência

Antes de iniciar o processo de modelagem, foram identificados e assinalados os pares de variáveis altamente correlacionados. Pelo fato de que o modelo de regressão logística é extremamente sensível à existência de colinearidade entre as variáveis independentes[5], podendo ocasionar estimativas extremamente exageradas dos coeficientes de regressão, duas variáveis foram excluídas amostras sendo uma contínua e outra discreta.

Outro ponto a se observar foi o tratamento das informações considerando possíveis ocorrências de *"missing"* nos dados, ou seja, ausência de conteúdo ou informação nas variáveis independentes. Desta maneira, todos os registros ou variáveis que continham *"missing"* foram desconsiderados no estudo uma vez que mesmo com tais exclusões, o tamanho da base amostral final se mostrou estatisticamente relevante, não impactando na modelagem dos dados.

Para o modelo de regressão logística foram selecionadas aleatóriamente amostras para utilização nos processos de desenvolvimento e validação preservando o critério de pareamento das bases, ou seja, uma parte da amostra contendo 50% de transações oriundas de clientes bons e 50% de ruins. Desta forma, as variáveis resposta foram construídas com domínio binário, ou seja, (0) para transações de clientes inadimplentes e (1) para transações de clientes adimplentes.

Utilizou-se um total de 10 amotras, todas disjuntas, com 10.000 registros cada, tanto na etapa de desenvolvimento como na etapa de validação. Visando assinalar os padrões de inadimplência, (0) ou (1), o conceito utilizado foi a "modelagem de transações de clientes adimplentes" de maneira que, quanto maior for a pontuação ou escoragem do modelo, menor será o risco de crédito associado. Desta forma, espera-se que a quantidade de transações ruins diminua conforme aumenta a pontuação do modelo.

2.2.2 Análise dos Resultados do Modelo

Para a implementação da metodologia, foi utilizado o módulo *Enterprise Miner* do aplicativo computacional SAS[7]. A estimação do modelo foi realizada através do método *stepwise* baseado num algoritmo misto de inclusão e eliminação de variáveis segundo a importância das mesmas de acordo com algum critério estatístico[8]. Sendo assim, as variáveis independentes são incluídas, uma por vez, de acordo com o poder discriminatório de cada uma delas.

O procedimento realizado foi variar aleatoriamente as amostras de desenvolvimento e validação, totalizando 10 simulações independentes. O resultado apurado corresponde a equação gerada pelo modelo que permite a identificação de quais variáveis possuem maior poder de explicação. O ponto de corte adotado foi 0,5, valor padronizado para a técnica de regressão logística. Esse valor representa a probabilidade de ocorrência do evento segundo o critério de aleatoriedade ou chances iguais[5].

Desta forma, a classificação das transações como adimplentes ou inadimplentes é realizada com base num vetor de probabilidades correspondente à variável resposta (1), ou seja, transações de clientes adimplentes. As transações pelas quais a probabilidade estimada de não inadimplência resultou em um valor superior a 0,5 foram classificadas como adimplentes, caso contrário como inadimplentes.

Para identificar se o poder de discrimação do modelo obtido é ou não válido, realizamos a validação deste utilizando amostras de controle diferentes das utilizadas na etapa de desenvolvimento. Como indicador de análise de performance dos modelos desenvolvidos, tivemos inicialmente as matrizes de confusão[9]. Estas matrizes consistem em tabelas que comparam a classificação realizada pelos modelos com a classificação original das observações da amostra de dados. Tais matrizes são elaboradas através da análise de cada observação, visando identificar se houve classificação correta dos modelos. Os resultados dessa análise são os percentuais de acerto e erro de classificação dos modelos.

O Erro Tipo I, conhecido como taxa de *Falsos Positivos*[9], ocorre em função da classificação dos clientes bons como ruins. A taxa de Erro Tipo I corresponde ao número de clientes bons classificados incorretamente como ruins dividido pelo número total de bons. O Erro Tipo II, ou taxa de *Falsos Negativos*[9], ocorre em função da classificação dos clientes ruins como bons. A taxa de Erro Tipo II corresponde ao número de clientes ruins classificados incorretamente como bons dividido pelo número total de ruins. Calculando a média entre os erros I e II, obtemos o Erro Geral do modelo que permite-nos identificar a precisão do modelo através da expressão $[1 - \text{Erro Geral}]$.

Desta forma, foi apurado o Erro Geral para cada uma das 10 simulações permitindo obter um valor médio de 0,2308 ou seja, uma taxa de acerto de 76,93%. A matriz de confusão referente à simulação cujo Erro Geral associado mais se aproximou do valor médio é apresentada na Figura 2.2.

		CLASSE VERDADEIRA		
CLASSE DO MODELO		RUINS	BONS	TOTAL
	RUINS	3.765	1.074	4.839
	BONS	1.235	3.926	5.161
	TOTAL	5.000	5.000	10.000

ERROS	
Erro I	0,2148
Erro II	0,2470
Erro Geral	0,2309

Figura 2.2: Matriz de Confusão - Modelo Logístico.

Uma técnica bastante simples, porém extremamente eficiente na avaliação de modelos de classificação, é verificar graficamente se existe ordenação adequada dos dados segundo a qualidade de crédito.

Inicialmente, os dados são divididos em percentis igualmente distribuídos. Neste estudo, adotamos dividir os dados em 10 percentis com 1.000 registros cada. Estes percentis são interpretados como faixas de escoragem e são apresentados através do eixo das abscissas. No eixo das ordenadas são apresentados os percentuais referentes aos dados classificados como inadimplentes. Considerando o eixo das abscissas, espera-se encontrar maior concentração de dados inadimplentes nas faixas mais baixas. Desta maneira, um modelo pode ser considerado consistente quando o percentual de inadimplentes decrescer monotonicamente da faixa mais baixa para a mais alta, produzindo um alto poder de separação entre as classes.

Outro medida utilizada como indicador de desempenho pode ser obtida através de um teste de qualidade de ajuste conhecido como teste de Kolmogorov-Smirnov[10], ou abreviadamente KS. Esta é uma técnica não paramétrica que permite verificar se diferentes amostras são provenientes de uma população comum.

Para uma variável aleatória X, o teste KS tem por base a análise da proximidade entre uma função de distribuição populacional teórica $F_0(x)$, que é admitida na hipótese nula H_0 e a função de distribuição empírica da amostra $S(x)$. Para uma amostra de tamanho n, a função $S(x)$ expressa a soma das freqüências relativas dos dados com valores menores ou iguais a um valor qualquer x da variável X.

Sendo $(X_1, X_2, \ldots, X_n)$ uma amostra aleatória de uma população contínua X e $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$ a respectiva amostra ordenada, tem-se que a função distribuição empírica $S(x)$ é dada por

$$S(x) = \begin{cases} 0\,, & x < X_{(1)} \\ \frac{k}{n}\,, & X_{(k)} < x < X_{(k+1)} \\ 1\,, & x \leq X_{(n)} \end{cases}$$

A função de distribuição empírica $S(x)$ representa uma função em degrau que cresce na fração $1/n$. A estatística de teste, denotada por D_{obs}, corresponde ao supremo (ou máximo) da diferença, em valor absoluto, entre $S(x)$ e $F_0(x)$, quando são considerados todos os valores possíveis de X. Em notação simbólica,

$$D_{obs} = \max|S(x) - F_0(x)|.$$

A hipótese nula é então formulada supondo que a função de distribuição da população da qual provém a amostra é idêntica a uma função de distribuição $F_0(x)$ que se assume conhecida. Simbolicamente,

$$S(x) = \begin{cases} H_0 : F_x = F_0(x), \text{ para todos os valores de} X; \\ H_1 : F_x \neq F_0(x), \text{ para algum valor de} X. \end{cases}$$

Considerando o problema de classificação abordado no estudo, onde buscamos obter a melhor separação entre as classes, a tendência esperada é que todos os modelos obtidos rejeitem a hipótese de igualdade nas distribuições. Sendo assim, a aplicação do KS em nosso modelo pode ser realizada através dos resultados já utilizados no indicador de ordenação. A idéia é inicialmente ordenar as classes de bons e de ruis através dos percentis. Em seguida, são apurados separadamente os percentuais acumulados por faixa e calculadas as diferenças entre o percentual de bons e ruins. O valor percentual máximo encontrado entre as diferenças representará o valor do teste KS. Será considerado como modelo mais eficiente o que obtiver o maior valor no teste pois este percentual representa o poder de separação entre as classes bons e ruins.

De acordo com Picinini et al.[11], considera-se um modelo de *credit scoring* eficiente aquele que proporcionar um valor de KS igual ou superior a 30. Em modelos de *behavior scoring*, adotamos utilizar os valores praticados pelo mercado, ou seja, valores de

KS próximos a 70. Neste estudo, pelo fato de utilizar-se informações pertinentes aos dois tipos de modelos, cadastrais e comportamentais, consideramos modelos eficientes àqueles em que se obtenham valores da estatística no intervalo $30 \leq KS \leq 70$. A estatística KS assim como a curva de ordenação são apresentados na Figura 2.3.

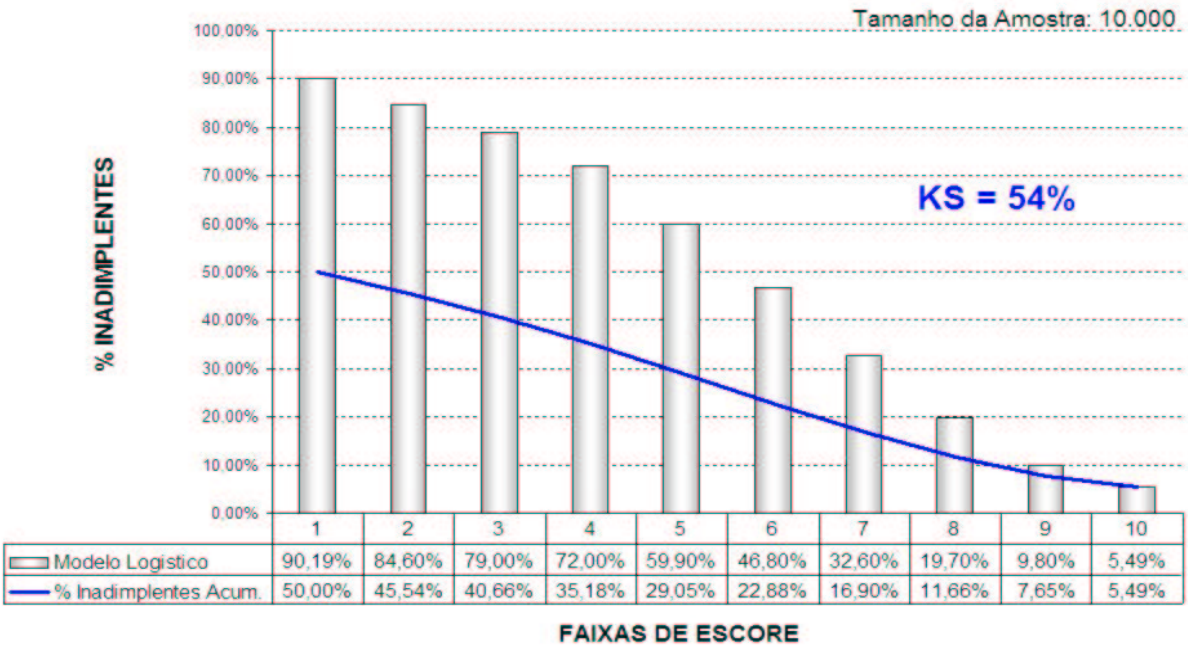


Figura 2.3: Desempenho do Modelo de Classificação com Regressão Logística.

O formato da curva de ordenação apresentada na Figura 2.3 permite-nos identificar um bom poder de separação entrea as classes. O valor de 54% obtido para a estatística KS pode também ser considerado bastante aceitável. Tais indicadores, assim como a taxa de acerto do modelo de 76,91% obtida através da matriz de confusão da Figura 2.2, comprova consistência em nossos resultados.

Um aspecto importante a se observar, também a através da matriz de confusão da Figura 2.2, é que o modelo apresentou melhor desempenho na classificação das transações inadimplentes, ou seja erros tipo I menor que o erro tipo II.

Outro fato observado refere-se às variáveis selecionadas pelo modelo logístico. Os resultados obtidos em todas as simulações permite-nos observar que variáveis valor da transação, saldos referentes a saques bem como algumas relacionadas à conta corrente do cliente não foram consideradas como variáveis explicativas do modelo e, de maneira geral, podem ser descartadas.

Capítulo 3

Redes Neurais Artificiais

De maneira a familiarizar-nos com o uso das técnicas baseadas nas chamadas máquinas de aprendizagem (*Machine Learning*)[12] assim como comparar a performance destes modelos com os resultados obtidos através do método estatístico, desenvolvemos modelos utilizando redes neurais artificiais. A teoria envolvida e os resultados obtidos são apresentados neste capítulo.

3.1 Aspectos Teóricos

As redes neurais artificiais (RNAs) são técnicas computacionais que representam um modelo matemático inspirado em sistemas neurobiológicos presentes em organismos inteligentes e que adquirem conhecimentos por intermédio de experiências. Mesmo possuindo um histórico de aproximadamente seis décadas, as metodologias baseadas em RNAs somente passaram a ser utilizadas de maneira consistente nos últimos vinte anos em decorrência da crescente evolução dos sistemas computacionais.

O primeiro modelo baseado em redes neurais foi idealizado pelo neurofisiologista McColluch e o matemático Walter Pitts em 1943 [13]. Inspirados em características biológicas do cérebro humano, construíram um modelo que simulava o comportamento de um neurônio real que possuía apenas uma saída sendo que esta saída correspondia a uma função soma dos valores da entrada.

Somente em 1962, pesquisando problemas de classificação com dados de entrada simples como as imagens binárias, Rosenblatt[14] criou o *perceptron* constituído por uma única camada de neurônios. Entretanto, trabalhos posteriores permitiram perceber que este tipo de estrutura não era capaz de solucionar problemas em que as classes envolvidas não eram linearmente separáveis. Com o aperfeiçoamento da idéia original do *perceptron* através da criação do algoritmo *back-propagation*[15] e do avanço computacional, ambos ocorridos a partir dos anos 80, cresceu substancialmente o número de pesquisas envolvendo a metodologia.

Atualmente, as RNAs têm sido empregadas em diversos campos de ciência e tecnologia, tais como reconhecimento e classificação de padrões, aproximação de funções complexas, diagnóstico de doenças, dentre outros[16]. Características importantes como capacidade de auto-aprendizado, habilidade em problemas de separação de padrões não lineares e, principalmente, alto poder de generalização são algumas das vantagens associadas à metodologia.

A idéia central da metodologia é tentar reproduzir a capacidade de aprendizado do ser humano. Toda capacidade de raciocínio e aprendizado ocorre em nosso sistema nervoso através de células complexas denominadas neurônios. Os neurônios, por sua vez, são compostos pelos dentritos, pelo corpo celular e pelos axônios. Os dentritos representam um conjunto de terminais de entrada e são responsáveis pela recepção dos estímulos. O corpo celular tem por finalidade capturar e combinar as informações e os axônios representam terminais que propagam os estímulos para outras células. A região onde dois neurônios entram em contato e através da qual são transmitidos os impulsos nervosos é chamada de sinapse.

A transmissão dos impulsos ocorre da seguinte forma: os pulsos recebidos por um neurônio são continuamente processados até que, atingido um limiar de ação, são disparados através da produção de uma substância neurotransmissora que flui do corpo celular para o axônio. Este está conectado a um dentrito de um outro neurônio, onde ocorrerá o mesmo processo de propagação, adicionadas novas modificações no pulso.

O neurônio que propaga o pulso pode controlar a intensidade, frequência e polaridade de emissão através de alterações na membrana pós-sináptica. No entanto, este processo depende de alguns fatores como a geometria da sinapse e do tipo de neurotransmissor.

De maneira similar ao processo biológico, as RNAs possuem habilidade de aprendizagem sobre possíveis padrões através de exposição de exemplos dos mesmos. Os dentritos são representados pelas *unidades de entrada*, cujas ligações com o corpo celular artificial são realizadas através de elementos denominados *pesos*, simulando as sinapses. Os estímulos são processados por uma *função soma*, e o limiar de disparo por uma *função de ativação*, que propaga os pulsos através de *unidades de saída*, representando os axônios.

O processo para estruturação de um modelo baseado nas RNAs pode ser realizado através das seguintes etapas: inicialmente realizamos a coleta dos dados de entrada, separando um grupo para treinamento da rede e outro para teste, lembrando que ambos devem ser disjuntos.

O próximo passo é definir a topologia da RNA para realização do treinamento. Este processo irá gerar uma matriz de pesos. Estes por sua vez, são utilizados no processo de teste da RNA onde são propagadas novas informações de maneira a permitir a avaliação de performance da rede treinada perante apresentação de padrões inéditos.

De maneira geral, temos que uma RNA consiste de um determinado número de elementos de processamento ou neurônios que, por sua vez, são dispostos através de camadas. Para cada camada são associados pesos cujo cálculo dos valores é conhecido como processo de treinamento ou aprendizagem da rede.

Existem dois tipos de treinamento, o supervisionado e o não supervisionado. Quando é indicada para a rede qual é a saída esperada e o objetivo é determinar a resposta correta para todos os vetores de entrada, estamos nos referindo a um processo de treinamento supervisionado. Por outro lado, quando a rede baseia-se apenas no conjunto de entradas fornecidas, sendo necessário agrupar os estímulos ou extrair propriedades de acordo com determinadas representações internas, estamos nos referindo a um processo de treinamento não supervisionado. Neste estudo, trabalhamos somente com os processos supervisionados.

Em seguida, para a definição de um modelo de RNA, é necessário observar a arquitetura da rede, o método de determinação dos pesos e a função de ativação. As camadas de uma RNA são classificadas como *camadas de entrada*, *camadas intermediárias* ou *ocultas* e *camadas de saída*. Os padrões são apresentados à rede através da *camada de entrada* enquanto que o processamento dos dados é realizado na *camada oculta*. A *camada de saída* corresponde ao elemento onde os resultados são finalizados e apresentados.

De acordo com Haykin[16], existem três tipos de arquitetura: redes *feedforward* com uma única camada, redes *feedforward* com múltiplas camadas e redes recorrentes. Dizemos que uma rede é do tipo *feedforward* se, em uma rede com todas as unidades ordenadas, desde as entradas até as saídas, cada unidade receba somente conexões provenientes de unidades anteriores, ou seja, que possuam ordem inferior à sua.

As redes *feedforward* com uma única camada, ou também conhecidas como *Linear Perceptrons* (LPs), são consideradas como o caso mais simples de rede, uma vez que existem somente uma camada de entrada e uma camada de saída. Um exemplo de rede *feedforward* com camada única é apresentada na Figura 3.1.[1]

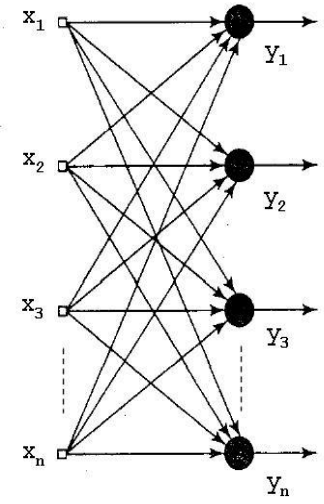


Figura 3.1: Exemplo de rede *feedforward* com uma única camada.

Na Figura 3.1, x_i representa as variáveis na camada de entrada e y_j as unidades na camada de saída, lembrando que a cada conexão entre os neurônios é atribuído um peso. As redes *feedforward* com camada única baseiam-se em uma combinação linear das variáveis de entrada, que são transformadas por uma função de ativação linear. Dependendo do problema a ser modelado, esta arquitetura pode resultar em redes muito limitadas com respeito ao tipo de função que esta consegue representar.

De forma a permitir um processo de mapeamento mais genérico, é necessário que existam sucessivas transformações correspondentes às redes contendo mais camadas e pesos ajustáveis, como é o caso das redes *feedforward* com múltiplas camadas ou também denominadas *Multilayer Perceptrons* (MLPs). Nas redes MLPs, a saída de cada camada é utilizada como entrada para a próxima camada.

[1]Fonte: Haykin, S., *Neural Networks - A Comprehensive Foundation*, 1999

A arquitetura de uma rede *feedforward* com múltiplas camadas pode ser identificada através na Figura 3.2.[2]

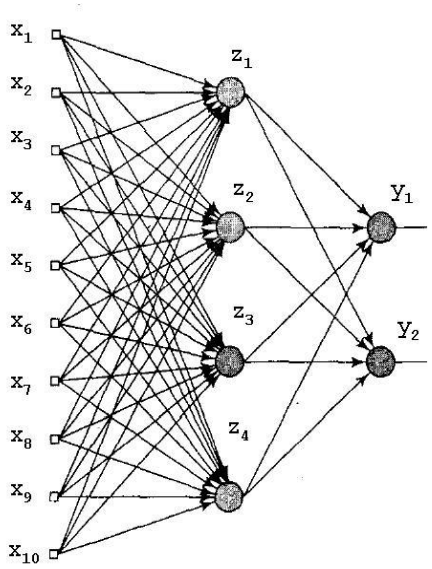


Figura 3.2: Exemplo de rede *feedforward* com múltiplas camadas.

Neste tipo de arquitetura, x_i representa as variáveis na camada de entrada, y_j as unidades na camada de saída e z_k as unidades ocultas ou intermediárias. Como na rede de camada única, a cada conexão é atribuído um peso. Analogamente à arquitetura anterior, este tipo de rede caracteriza-se apenas por alimentação adiante.

Nas redes recorrentes, a camada de saída possui ao menos uma ligação que realimenta a rede. Entretanto, neste trabalho somente estaremos considerando modelos baseados em redes *feedforward* com camada única e em redes *feedforward* com múltiplas camadas. Maiores detalhes sobre redes recorrentes podem ser encontrados em Haykin[16].

O passo seguinte, considerando o modelo baseado em RNA, é definir o método de determinação dos pesos, ou seja, o processo de aprendizagem da rede. Sabemos que as informações são processadas nos neurônios e estes são conectados entre si mediante padrões que definem a arquitetura da rede. Cada conexão possui um peso associado que é multiplicado por um estímulo recebido e cada neurônio possui um estado, denominado ativação, que representa uma função das entradas recebidas por ele. Esta ativação é responsável pelo envio de um sinal para outros neurônios, permitindo um estímulo na camada de saída.

[2]Fonte: Haykin, S., *Neural Networks - A Comprehensive Foundation*, 1999

Considerando novamente o modelo pioneiro de McCulloch e Pitts 1943 [13], podemos resumir a mecânica operacional do modelo neural da seguinte maneira: os sinais são apresentados à entrada e em seguida são multiplicados por pesos que indicam a influência na saída da unidade de processamento (neurônio); em seguida, é realizada a soma ponderada dos sinais permitindo produzir um nível de atividade; caso este nível exceda um determinado limite, a unidade produzirá uma saída.

Sejam $x_1, x_2, ..., x_p$ os sinais de entrada, $w_1, w_2, ..., w_p$ os pesos correspondentes e k o limite de disparo do estímulo. Podemos então definir o nível de atividade como:

$$\alpha = \sum_{i=1}^{p} w_i x_i \tag{3.1}$$

No modelo de RNA utilizado neste trabalho, definimos a variável resposta de maneira a assumir apenas os valores 0 ou 1. Desta forma, a saída será dada por $y - 1$, se $\alpha \geq k$ e $y - 0$, se $\alpha < k$

Segundo Haykin[16], a função de ativação desempenha o papel de restringir a amplitude de saída de um neurônio, podendo assumir valores binários ou contínuos, em geral [0,1] ou [-1,1]. Podemos citar alguns exemplos de funções de ativação utilizadas:

- Limiar ou *Heaviside*: $f(x) = \begin{cases} 1\,, & \text{se } x \geq 0 \\ 0\,, & \text{se } x < 0 \end{cases}$
- *Piecewise* Linear: $f(x) = \begin{cases} 1\,, & x \geq +\frac{1}{2} \\ x\,, & -\frac{1}{2} < x < +\frac{1}{2} \\ 0\,, & x \leq +\frac{1}{2} \end{cases}$
- Linear: $f(x) = x$
- Logística: $f(x) = \frac{1}{1+e^{-\mu(x)}}$
- Tangente Hiperbólica: $f(x) = \tanh(x)$

A função utilizada neste estudo foi a função logística pelo fato de que suas respostas podem ser interpretadas como probabilidades a posteriori, ou seja, permite-nos a interpretação estatística do modelo dado que estamos diante de um problema de separação em duas classes[12].

3.2 Aplicação e Resultados

Esta seção apresenta a metodologia bem como os resultados obtidos através da utilização de modelos baseados em RNAs. Para nossa familiarização com a metodologia, escolhemos realizar a modelagem utilizando inicialmente uma RNA *feedforward* com uma única camada e, em seguida, uma RNA *feedforward* multicamada.

3.2.1 Seleção e Tratamento dos Dados de Inferência

A metodologia de coleta e tratamento dos dados ocorreu de maneira análoga ao modelo logístico. Desta maneira, utilizamos as mesmas 14 variáveis de entrada (7 variáveis contínuas e 7 variáveis discretas) e os com os mesmos critérios para definição de safra de observação, período histórico e performance.

Foram selecionadas 10 amotras, todas disjuntas, com 10.000 registros cada, visando a apresentação destas no processo de treinamento e teste das redes. Foram mantidos todos os critérios de aleatoriedade e pareamento das bases, ou seja, variável resposta (0) para 5.000 transações de clientes inadimplentes e (1) para 5.000 transações de clientes adimplentes.

Alguns testes foram realizados utilizando-se amostras de dados normalizados e não normalizados. A idéia destes testes se deve ao fato de que as variáveis de entrada podem possuir diferentes dimensões em seus domínios. De maneira a minimizar estas diferenças e facilitar o processo de aprendizagem da rede, dividimos cada valor das variáveis de entrada pelo valor médio de suas observações. Embora os resultados obtidos em ambos os casos sejam muito próximos, tivemos pequena melhora quando utilizamos os dados normalizados e, desta forma, adotamos a prática nos experimentos seguintes.

Nos modelos neurais desenvolvidos neste trabalho, novamente consideramos a "modelagem de transações de clientes adimplentes"de maneira a possibilitar maior escoragem para transações com menor risco de crédito.

3.2.2 Método de Aprendizagem

O processo de aprendizagem de uma RNA baseia-se em um algoritmo de treinamento, que é definido por um conjunto de regras que determinam como os pesos iniciais podem ser adaptados para que a rede possa assimilar os padrões apresentados. Este processo está

diretamente ligado à definição de uma função erro e na escolha apropriada de um método de minimização deste erro.

Nos problemas envolvendo classificação de padrões, é tarefa difícil estabelecer valores apropriados para os pesos. A solução geral utilizada nos algoritmos de aprendizagem é fazer com que a rede aprenda, treinando-a com padrões. Desta forma, consideramos que a aprendizagem da rede refere-se a um processo iterativo que busca minimizar a função erro através dos ajustes dos pesos sendo realizado em duas etapas. Na primeira etapa é calculada a derivada da função erro em relação aos pesos, enquanto que na segunda etapa estas são utilizadas para computar os possíveis ajustes nos pesos.

A função erro estabelece uma medida de eficiência da RNA, sendo que o processo de otimização baseia-se em sua minimização. No entanto, a escolha da função erro está diretamente ligada ao problema a ser modelado bem como na função de ativação escolhida. Neste estudo, onde o objetivo principal é modelar a probabilidade a posteriori dos elementos de uma classe, condicionados às variáveis de entrada, a escolha da função soma dos quadrados dos erros é apropriada.

O passo seguinte é escolher um método de otimização eficiente para a minimização do erro. Através do trabalho de Bishop[17] identificamos a existência de várias técnicas de otimização que podem ser utilizadas para minimização dos erros tais como *Gradient Descent*, Gradiente Conjugado, Gradiente Conjugado Escalado, Método de Newton, *Quasi-Newton*, dentre outras.

As técnicas de otimização acima baseiam-se na definição de um passo em direção à maior taxa de decrescimento do erro em relação aos pesos computados, ou seja, na direção do gradiente negativo. No entanto, a escolha do método deve também considerar aspectos importantes como desempenho da rede, eficiência computacional, escolha apropriada de parâmetros e principalmente convergência.

Sendo assim, escolhemos utilizar um método simples, porém muito eficiente, chamado Gradiente Conjugado Escalado. Este algoritmo foi introduzido por Moller[18] e possui vantagens consideráveis, tais como não dependência da escolha de parâmetros bem como baixo custo computacional.

3.2.3 Análise dos Resultados do Modelo

Para a implementação desta técnica ao problema abordado, utilizamos o pacote computacional *Netlab*[34]. Esta é uma *toolbox* de redes neurais em *Matlab* disponibilizada gratuitamente pelo *Neural Computing Research Group* da Universidade de Aston.

Redes Neurais Camada Única

No processo de treinamento das redes, utilizamos as 14 variáveis de entrada, 1 variável resposta e a função logística como função de ativação. O *perceptron* linear que representa esta arquitetura é apresentado na Figura 3.3.

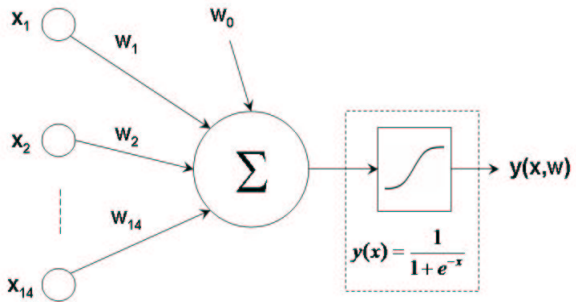


Figura 3.3: Arquitetura do Modelo RNA Camada Única.

onde,

$$y(x, w) = \sum_{i=1}^{M} w_i \, x_i + w_0. \tag{3.2}$$

O valor de w_0, também conhecido como "*vicio*", tem a finalidade de aumentar o número de graus de liberdade do modelo, permitindo um aumento na capacidade da RNA para se ajustar ao conhecimento que lhe é fornecido.

O procedimento realizado no desenvolvimento do modelo foi novamente variar aleatoriamente as amostras de treino e teste, duas a duas, totalizando 10 simulações independentes. Na modelagem com RNAs, não se permite identificar quais variáveis possuem maior poder de explicação, pois todas elas são consideradas como padrões.

O ponto de corte adotado foi novamente 0,5 de maneira a permitir a separação das duas classes de transações a partir do vetor de probabilidades fornecido pelo modelo. Transações cujo valor de probabilidade resultou em um valor superior a 0,5 foram classificados como adimplentes e, caso contrário, como inadimplentes.

Outro aspecto importante na modelagem neural, foi utilizar um critério de parada para o treinamento da rede. Neste estudo, utilizamos o número de iterações como métrica de interrupção do treinamento. Para determinar o limiar de parada, realizamos o treinamento de algumas redes possibilitando identificar que o erro de treinamento praticamente não se alterava a partir de 100 iterações, conforme ilustrado na Figura 3.4. Considerando a eficiência e simplicidade do algoritmo de otimização bem como o baixo custo computacional, foi considerado apropriado fixar o limiar em 500 iterações.

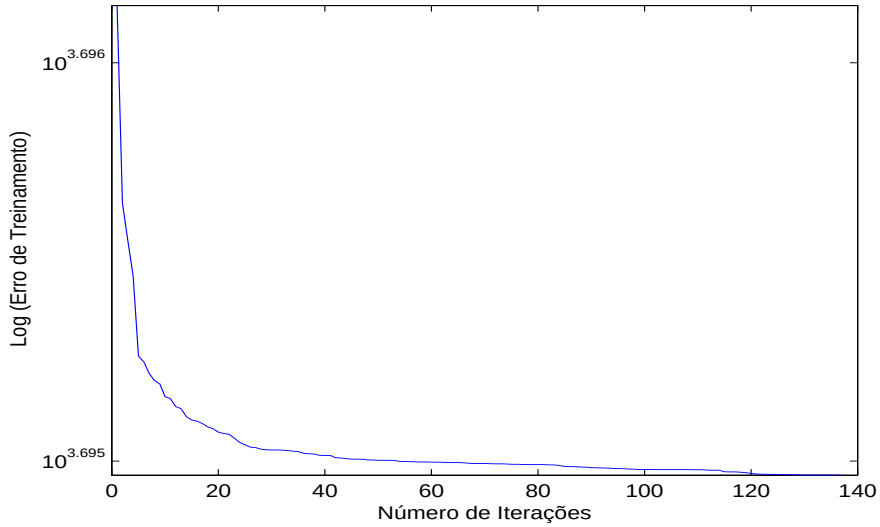


Figura 3.4: Evolução do Erro de Treinamento - RNA Camada Única.

Para avaliação da performance do modelo neural, utilizamos os mesmos indicadores considerados no modelo de regressão logística: matrizes de confusão, taxas de acerto, KS e ordenação das classes. A partir das 10 simulações de treinamento e teste, foram apurados os resultados permitindo obter um valor médio de 76,59% de taxa de acerto.

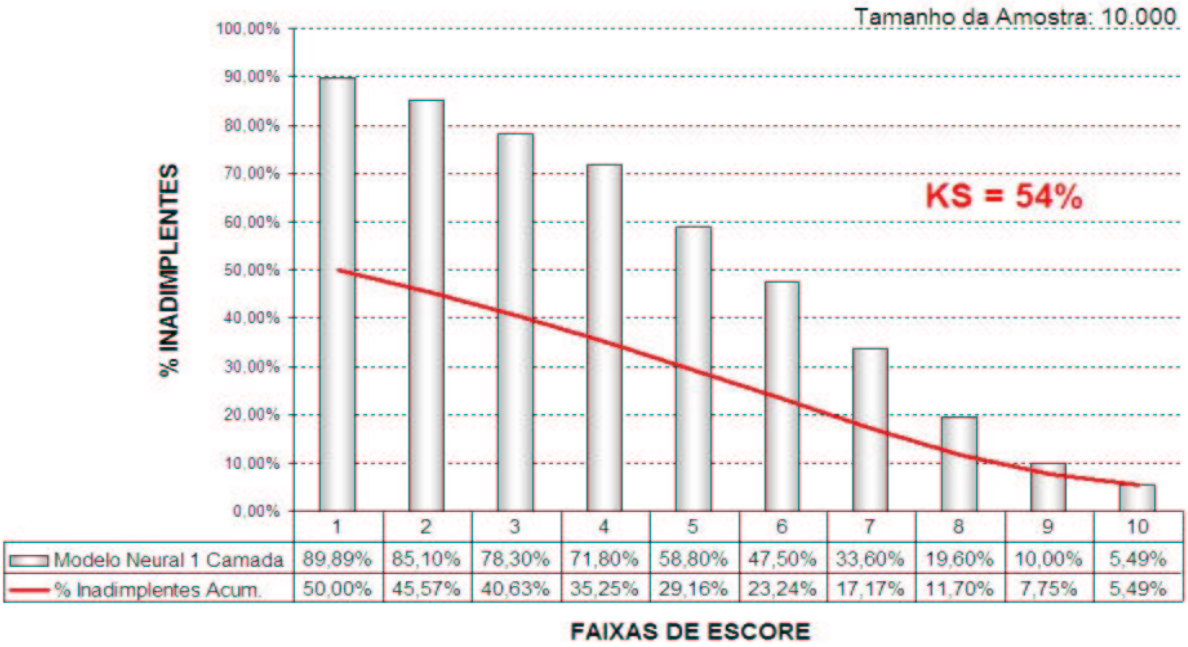
Confrontando a taxa de acerto obtida através deste modelo com o valor apurado através da regressão logística (76,91%), confirmamos a consistência do modelo tendo em vista também o aspecto teórico de que o modelo de RNA com uma camada pode ser interpretado como um modelo de regressão logística. A matriz de confusão referente à simulação, cujo erro geral coincidiu com o valor médio de 0,2341 é apresentada na Figura 3.5.

		CLASSE VERDADEIRA		
		RUINS	BONS	TOTAL
CLASSE DO MODELO	RUINS	3.725	1.066	4.791
	BONS	1.275	3.934	5.209
	TOTAL	5.000	5.000	10.000

ERROS	
Erro I	0,2132
Erro II	0,2550
Erro Geral	0,2341

Figura 3.5: Matriz de Confusão - Modelo RNA Camada Única.

A estatística KS bem como a curva de ordenação são apresentados na Figura 3.6.



	1	2	3	4	5	6	7	8	9	10
Modelo Neural 1 Camada	89,89%	85,10%	78,30%	71,80%	58,80%	47,50%	33,60%	19,60%	10,00%	5,49%
% Inadimplentes Acum.	50,00%	45,57%	40,63%	35,25%	29,16%	23,24%	17,17%	11,70%	7,75%	5,49%

Figura 3.6: Desempenho do Modelo de Classificação com RNA Camada Única.

Através dos indicadores apurados no modelo, confirmamos consistência nos resultados. O valor de KS de 54%, considerado apropriado para o modelo bem como o indicador de ordenação, novamente confirmam a similaridade do modelo neural camada única com o modelo de regressão logística. Como também ocorreu no modelo de regressão logística, observamos erro tipo I menor que o erro tipo II.

Redes Neurais Multicamada

Na simulação do modelo utilizando RNA multicamada, consideramos inicialmente o problema de se encontrar a arquitetura apropriada para a rede. Considerando que a topologia em questão difere da anterior pela existência de camadas escondidas, propomos primeiramente encontrar o número adequado de neurônios em função da estabilidade do erro em cada simulação.

Desta maneira, foi proposto realizar a simulação do modelo, utilizando as mesmas variáveis de entrada e resposta porém, variando-se a quantidade de unidades ocultas.

Foram realizadas 20 simulações sendo que em cada simulação, consideramos 10 combinações aleatórias de amostras de treino e teste. A cada simulação variamos o número de unidades na camada escondida, de 1 a 20, e apuramos os erros médios e desvios das 10 combinações. O erros e os desvios foram coletados através das matrizes de confusão. A função de ativação logística bem como o método minimização, Gradiente Conjugado Escalado, foram mantidos. A Figura 3.7. apresenta os resultados.

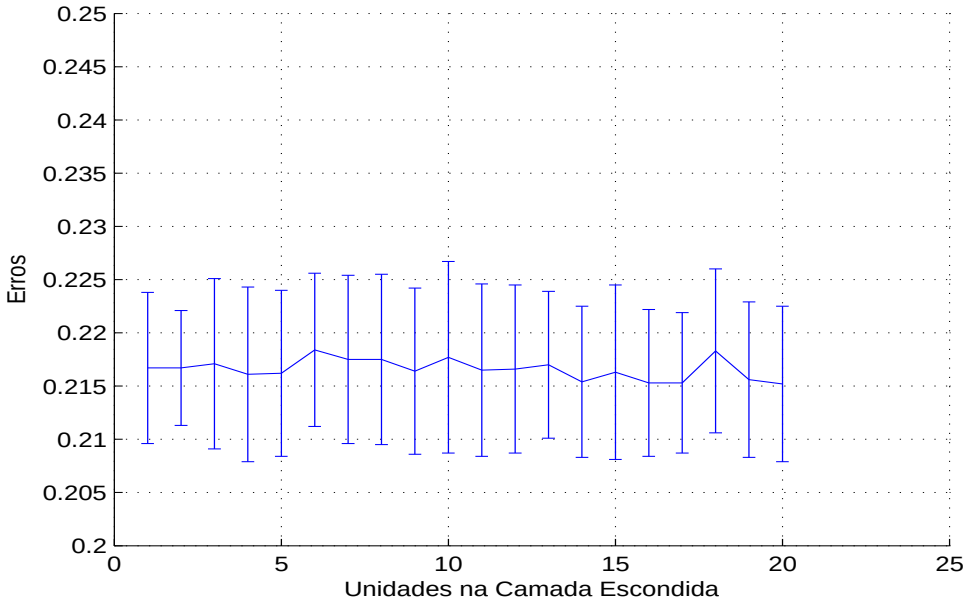


Figura 3.7: Evolução dos Erros de Classificação - RNA Multicamada.

Analisando a Figura 3.7, podemos observar a característica de estabilidade nos erros, o que nos leva a concluir que, para o problema em questão, o número de unidades na camada escondida não influencia os resultados. Outro ponto importante a ser identificado é que, para a escolha da arquitetura apropriada para o modelo multicamada, também não seria necessário realizar uma estratégia de combinação de todas as respostas geradas, as chamadas máquina de comitês, propostas por Haykin[16].

Sendo assim, considerando a estabilidade do erro de classificação bem como o menor custo computacional envolvido na arquitetura mais simples, escolhemos trabalhar com uma RNA multicamada com 14 variáveis na camada de entrada, 1 variável resposta na camada de saída e somente 1 neurônio na camada escondida. O *perceptron* não-linear que representa a arquitetura escolhida por parcimônia é apresentado na Figura 3.8.

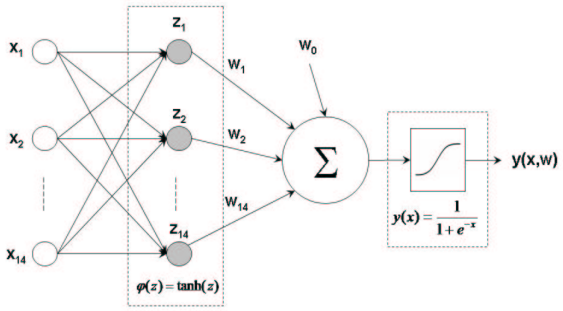


Figura 3.8: Arquitetura do Modelo RNA Multicamada - 1 Unidade na Camada Escondida.

A arquitetura apresentada na Figura 3.8 permite a implementação da seguinte família de funções[20]:

$$y(x,w) = \varphi_0 \left[\sum_{m=1}^{M} w_{m0}\, \varphi \left(\sum_{j=1}^{N} w_{jm}\, x_j + w_0 \right) + {w_0}^{out} \right] \tag{3.3}$$

A rede representada pela Equação (3.3) é treinada de maneira que seja obtido um hiperplano que divide o espaço em duas regiões. Desta maneira, uma RNA multicamada com uma unidade na camada escondida pode representar qualquer região convexa no espaço sendo que cada neurônio da camada escondida define un hiperplano de classificação. Para o nosso modelo, adotamos utilizar a função $\varphi(z) = \tanh(z)$ para a camada interna e $\varphi_0(z) = \frac{1}{1+e^{-z}}$ para a camada externa.

O procedimento realizado para treinamento da rede foi novamente variar aleatoriamente as amostras de treino e teste, duas a duas, totalizando 10 simulações independentes. O ponto de corte adotado para definir a classificação das transações em boas ou ruins foi novamente o valor 0,5.

Com respeito ao critério de parada para o treinamento da rede, o procedimento foi análogo ao realizado nas simulações com RNA camada única. Realizamos o treinamento de algumas redes e identificamos que o erro de treinamento praticamente não se alterava a partir de 250 iterações, conforme ilustrado na Figura 3.9.

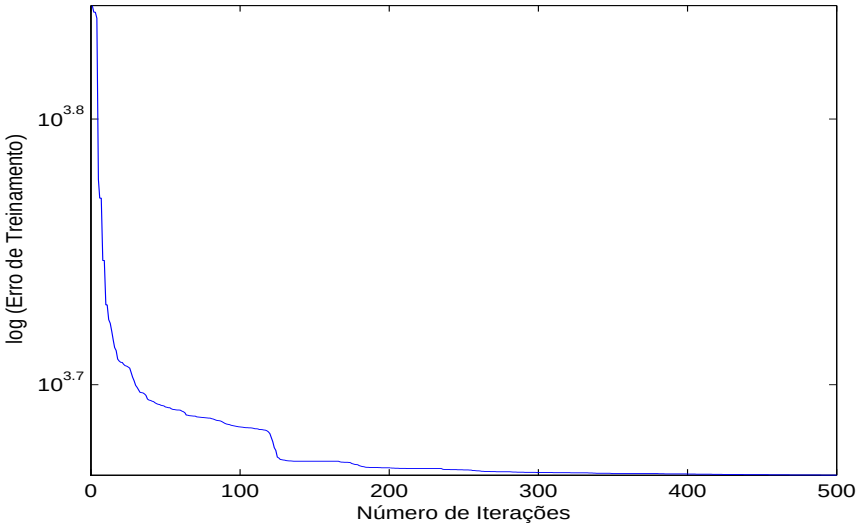


Figura 3.9: Evolução do Erro de Treinamento - RNA Multicamada.

Desta maneira, considerando também o rápido e eficiente processamento realizado em apenas alguns segundos, novamente definimos limitar o processo em 500 iterações.

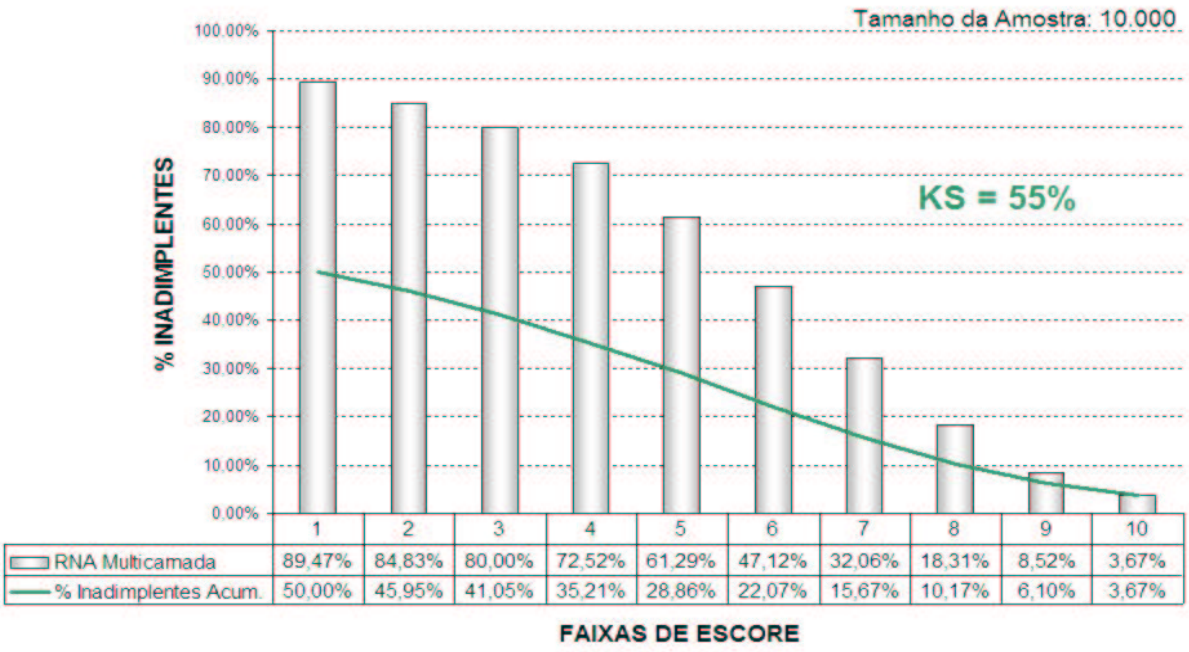
Para avaliação da performance do modelo neural multicamada, utilizamos novamente os indicadores apurados através das matrizes de confusão assim como a estatística KS e ordenação das classes. A partir das 10 simulações de treinamento e teste, foram apurados os resultados permitindo obter um valor médio de 78,33% de taxa de acerto. A matriz de confusão referente à simulação, cujo erro geral mais se aproximou do valor médio é apresentada na Figura 3.10.

		CLASSE VERDADEIRA		
CLASSE DO MODELO		RUINS	BONS	TOTAL
	RUINS	4.029	1.183	5.212
	BONS	971	3.817	4.788
	TOTAL	**5.000**	**5.000**	**10.000**

ERROS	
Erro I	0,2366
Erro II	0,1942
Erro Geral	**0,2154**

Figura 3.10: Matriz de Confusão - Modelo RNA Multicamada.

A estatística KS bem como a curva de ordenação são apresentados na Figura 3.11.



	1	2	3	4	5	6	7	8	9	10
RNA Multicamada	89,47%	84,83%	80,00%	72,52%	61,29%	47,12%	32,06%	18,31%	8,52%	3,67%
% Inadimplentes Acum.	50,00%	45,95%	41,05%	35,21%	28,86%	22,07%	15,67%	10,17%	6,10%	3,67%

Figura 3.11: Desempenho do Modelo de Classificação com RNA Multicamada.

Através dos indicadores apurados no modelo, confirmamos melhores resultados para o modelo desenvolvido através de aquitetura multicamada. A taxa de acerto de 78,46% bem como um KS de 55% comprova sensível melhora comparado ao modelo logístico ou, da mesma maneira, RNA com uma camada. O indicador de ordenação também mostra consistência nos resultados.

O aspecto interessante a se observar é que, comparando os erros tipo I e II, agora tivemos erro tipo I maior que o erro tipo II. Tanto no modelo de regressão logística quanto no modelo RNA com uma camada, tivemos resultados mais consistentes em relação ao erro tipo I.

Embora obtendo-se indicadores superiores na modelagem com RNA multicamada, ambas metodologias apresentaram resultados apropriados para o nosso problema de classificação. O fato importante a se destacar é que pudemos nos familiarizar com teoria de aprendizagem de máquina em que se baseia o modelo neural. Sendo assim, prosseguimos com os nossos experimentos, visando agora identificar o desempenho do modelo aplicando a metodologia máquina de vetores de suporte, as SVMs.

Capítulo 4

Máquinas de Vetores de Suporte

As máquinas de vetores de suporte, do termo inglês *Support Vector Machines (SVMs)*, são máquinas de aprendizagem baseadas em treinamento supervisionado e foram desenvolvidas originalmente por Boser, Vapnik e Guyon[21]. De acordo com Vapnik[22], esta técnica pode ser comparada a uma máquina de aprendizagem estruturada com apenas uma unidade na camada escondida. A principal característica das SVMs é a determinação automática dos dados de treinamento mais relevantes para o problema em questão, os chamados vetores de suporte.

As SVMs são ferramentas adequadas para situações onde encontramos sobreposição de dados, ou seja, problemas onde as classes não podem ser separadas linearmente. A excelente capacidade de generalização bem como a possibilidade de superar tendências de excesso de ajustes *(overfitting)* são algumas das vantagens associadas à metodologia.

A idéia deste capítulo é fundamentar o problema em questão, descrevendo inicialmente o problema e os conceitos envolvidos nos casos em que é possível separar os dados através de uma fronteira linear. Em seguida, será apresentada toda a teoria da abordagem SVM para aplicação nos problemas em que as classes não são separáveis linearmente. Após a apresentação de toda a base teórica, a aplicação da técnica SVM na modelagem do problema de classificação de transações com cartão de crédito, a metodologia envolvida bem como os resultados obtidos, serão apresentados e discutidos.

4.1 Classes Linearmente Separáveis

Nos capítulos anteriores, verificamos que as abordagens baseadas no método logístico e, consequentemente, das RNAs com uma camada permitem-nos estimar uma fronteira linear de decisão para o problema de classificação. Utilizaremos a teoria envolvida nestes processos de maneira a nos fornecer toda a base inicial para a compreensão do modelo de classificação por vetores de suporte.

Considerando a Figura 4.1[3], são apresentados 20 dados representados através de pontos, sendo que a separação entre eles ocorre através de duas classes na dimensão $\mathbb{R}^2$.

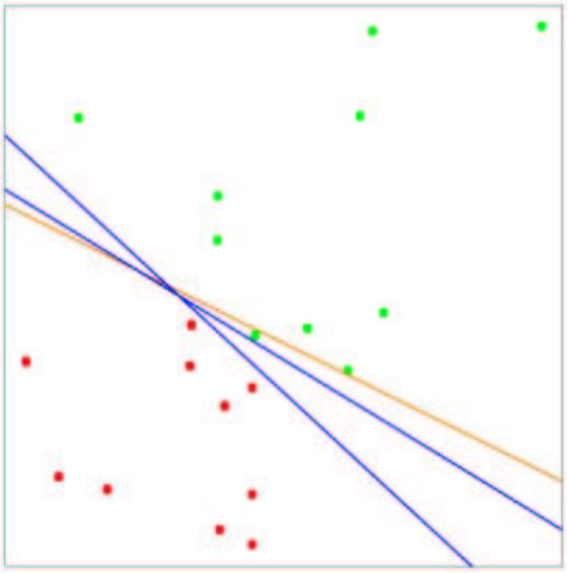


Figura 4.1: Exemplo de 2 Classes Separáveis Linearmente.

As retas em azul representam dois dos infinitos hiperplanos de separação possíveis para o problema enquanto que a reta em laranja representa a solução por mínimos quadrados representada por:

$$\{x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0\} \tag{4.1}$$

Entretanto, verificamos que a solução obtida através de mínimos quadrados ou, de maneira equivalente, a fronteira do modelo logístico, não permite uma perfeita separação dos dados.

[3]Fonte: Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, 2001.

Soluções análogas à representada pela Equação (4.1), que calculam uma combinação linear das variáveis de entrada e devolvem um sinal, são conhecidas como *perceptrons*, cuja a idéia já foi mencionada no capítulo referente às RNAs. As retas em azul, na Figura 4.1, representam as soluções obtidas através da abordagem dos *perceptrons*.

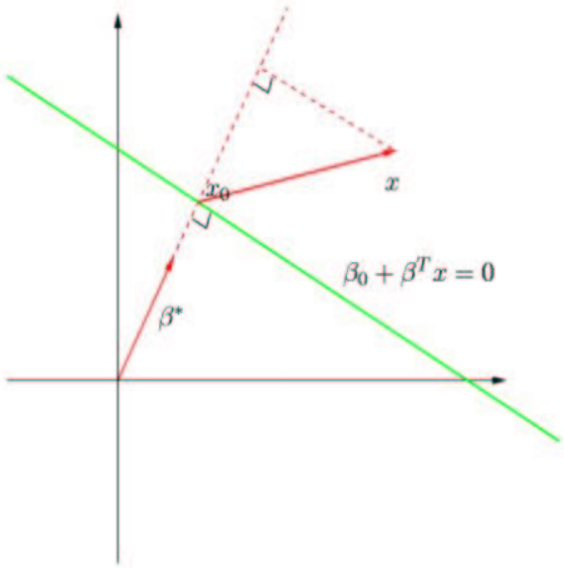


Figura 4.2: Álgebra Linear de um Hiperplano.

Visando facilitar a compreensão do problema, vamos utilizar algumas definições de álgebra vetorial. Considere a reta em verde na Figura 4.2[4] como sendo um hiperplano L, representado por $f(x) = \beta_0 + \beta^T x = 0$. Algumas propriedades importantes:

1. Para quaisquer dois pontos x_1 e x_2 em L, $\beta^T(x_1 - x_2) = 0$, que implica que $\beta^* = \beta/\|\beta\|$ representa o vetor normal à superfície de L;

2. Para qualquer ponto x_0 em L, $\beta^T x_0 = -\beta_0$;

3. A distância de qualquer ponto x a L é dada por:

$$\begin{aligned} \beta^{*T}(x - x_0) &= \frac{1}{\|\beta\|}(\beta^T x + \beta_0) \\ &= \frac{1}{\|f'(x)\|} f(x) \end{aligned} \tag{4.2}$$

Portanto, $f(x)$ é proporcional à distância entre x e o hiperplano definido por $f(x) = 0$.

[4]Fonte: Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, 2001.

Vamos agora considerar o problema de se encontrar um hiperplano de separação através das técnicas neurais. O algoritmo de aprendizagem utilizado nos *perceptrons* tenta minimizar a distância entre os pontos classificados incorretamente à fronteira de decisão. Caso uma saída $y_i = 1$ seja classificada incorretamente, então ${x_i}^T\beta+\beta_0 < 0$ e, caso contrário, ${x_i}^T\beta + \beta_0 > 0$ quando $y_i = -1$. Sendo assim, o problema de minimização será:

$$D(\beta, \beta_0) = -\sum_{i \in M} y_i({x_i}^T\beta + \beta_0), \tag{4.3}$$

onde M representa o conjunto de pontos mal classificados. A quantidade é considerada não negativa e proporcional à distância dos pontos mal classificados até a fronteira de decisão definida por $\beta^T x + \beta_0$. Assumindo que M é fixo, os gradientes serão dados por

$$\partial\frac{D(\beta, \beta_0)}{\partial\beta} = -\sum_{i \in M} y_i x_i, \tag{4.4}$$

$$\partial\frac{D(\beta, \beta_0)}{\partial\beta_0} = -\sum_{i \in M} y_i. \tag{4.5}$$

Ou seja, ao invés de realizar a soma das contribuições de cada observação e, em seguida, operar na direção negativa do gradiente, o algoritmo trabalha de maneira a realizar apenas um passo à medida que cada observação é considerada. Sendo assim, as observações classificadas incorretamente são tratadas através de alguma sequência e os parâmetros β são atualizados como

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} \leftarrow \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} + \rho \begin{pmatrix} y_i x_i \\ y_i \end{pmatrix}. \tag{4.6}$$

A taxa de aprendizagem do processo é representada por ρ que, neste caso, pode ser considerada como 1 sem perda de generalidade. Se as classes são linearmente separáveis, após um número finito de passos, um hiperplano separável será obtido conforme representado na Figura 4.1 através das linhas em azul.

No entanto, são observados alguns problemas no algoritmo. O primeiro deles é que, nas situações em que os dados são separáveis, existem muitas soluções sendo que a solução encontrada depende dos valores iniciais. Outro fator é que este número finito de passos pode

ser consideravelmente grande, ocorrendo em esforço computacional elevado para obtenção da solução. Contudo, o fato é que em problemas em que os dados não são separáveis, não haverá convergência.

A maneira descoberta para contornar estes problemas foi encontrar um hiperplano, não mais no espaço original e sim, em um espaço de dimensão elevada através da transformação do conjunto de dados inicial em um conjunto de padrões linearmente separáveis.

4.1.1 O Hiperplano Ótimo

O trabalho de Vapnik[22] apresenta que, através de um hiperplano ótimo, é possível separar duas classes maximizando a distância do ponto mais próximo ao hiperplano obtido. Estaremos nos referindo a esta distância como *margem de separação* que será representada por C. Para tal, é necessário generalizar o critério descrito na Equação (4.3) através da formulação do problema de otimização

$$\begin{aligned} &\max_{\beta,\beta_0,\|\beta\|=1} \quad C \\ &\text{sujeito a } \; y_i({x_i}^T\beta + \beta_0) \geq C, \; i = 1, \ldots, N. \end{aligned} \tag{4.7}$$

O conjunto de condições assegura que todos os pontos permaneçam, no mínimo, a uma distância C a partir da fronteira de decisão definida por β e β_0. De maneira a eliminarmos a restrição $\|\beta\| = 1$, substituímos as condições por

$$\frac{1}{\|\beta\|}\, y_i({x_i}^T\beta + \beta_0) \geq C, \tag{4.8}$$

ou, de maneira equivalente

$$y_i({x_i}^T\beta + \beta_0) \geq C\,\|\beta\|. \tag{4.9}$$

Uma vez que quaisquer β e β_0 satisfazem a desigualdade, podemos arbitrariamente escolher $\|\beta\| = 1/C$. Desta forma, o problema de otimização descrito na Equação (4.7) é equivalente a

$$\begin{aligned} &\min_{\beta,\beta_0} \; \frac{1}{2}\,\|\beta\|^2 \\ &\text{sujeito a } \; y_i({x_i}^T\beta + \beta_0) \geq 1, \; i = 1, \ldots, N. \end{aligned} \tag{4.10}$$

Considerando a Equação (4.2), identificamos que as restrições do problema definem a margem de separação sendo que sua espessura é definida por $1/\|\beta\|$. A escolha de β e β_0 permite maximizar esta espessura. Desta forma, estamos diante de um problema de otimização convexa, descrito por uma função quadrática e com restrições de desigualdade.

O problema primal é descrito através do Lagrangiano

$$L_P = \min_{\beta,\beta_0} \frac{1}{2} \|\beta\|^2 + \sum_{i=1}^{N} \alpha_i [y_i({x_i}^T \beta + \beta_0) - 1]. \tag{4.11}$$

Derivando (4.11) em relação a β, β_0 e igualando os resultados a zero, obtemos

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{4.12}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{4.13}$$

Substituindo as equações anteriores em (4.11), chegamos ao problema dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k {x_i}^T x_k$$
$$\text{sujeito a} \quad \alpha_i \geq 0. \tag{4.14}$$

Desta maneira, chegamos ao problema de otimização convexa descrito por L_D e que pode ser resolvido numericamente sem grandes implicações. Em adição, a solução do problema deve satisfazer as condições de Karush-Kuhn-Tucker[23] descritas em (4.12), (4.13), (4.14) e

$$\alpha_i [y_i({x_i}^T \beta + \beta_0) - 1] = 0 \quad \forall i. \tag{4.15}$$

A partir da equação anterior, podemos extrair as seguintes propriedades

- se $\alpha_i > 0$ então $y_i({x_i}^T \beta + \beta_0) = 1$, ou seja, x_i está na margem;
- se $y_i({x_i}^T \beta + \beta_0) > 1$, x_i não está na margem e $\alpha_i = 0$;

Através da Equação (4.15), observamos que o vetor solução β é definido em termos de uma combinação linear de pontos de suporte x_i. A Figura 4.3[5] apresenta o hiperplano ótimo sendo que os pontos de suporte são representados pelos 3 pontos em azul. O aspecto interessante a se observar é que, quando os dados são separáveis, é obtido um hiperplano ótimo que apresenta máxima margem de separação.

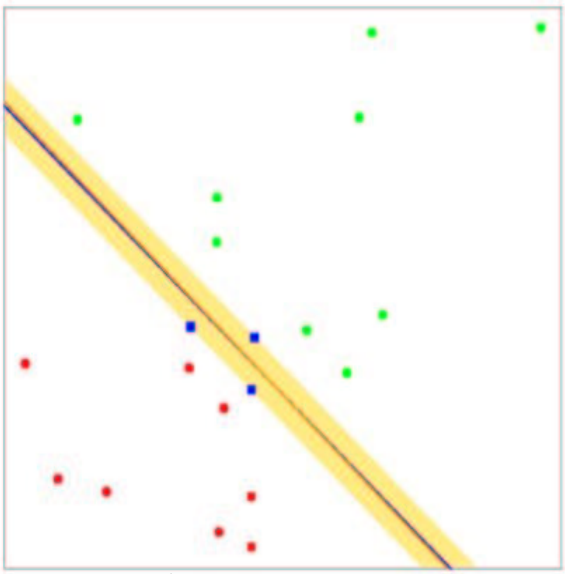


Figura 4.3: Hiperplano Ótimo de Separação e Pontos de Suporte.

Para quaisquer pontos de suporte, podemos obter o valor de β_0 através da Equação (4.15). Assim, o hiperplano ótimo de separação produz uma função $f(x) = x^T\hat{\beta} + \hat{\beta}_0$ para a classificação de novas observações:

$$\hat{G}(x) = \text{sign}\,\hat{f}(x). \tag{4.16}$$

A solução descrita em termos dos pontos de suporte sugere que o hiperplano ótimo prioriza tratar os dados mais relevantes. No entanto, é importante observar que, para identificar os pontos de suporte, a metodologia requer o uso de todas as observações do problema.

Outro aspecto importante a se confirmar é que, diante de problemas em que as classes podem ser separadas linearmente, a regressão logística também fornecerá um hiperplano ótimo de separação. Quando os dados não são separáveis, não existirá uma solução factível para o problema e, desta forma, será necessário ampliar o espaço de dimensões utilizando as funções baseadas nos produtos inernos kernel, cuja abordagem será discutida adiante. Entretanto, podem ocorrer situações de excesso de ajustes *(overfitting)*.

[5]Fonte: Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, 2001.

A motivação para resolução do problema de *overfitting* foi identificada como base da teoria envolvendo as máquinas de vetores de suporte (SVMs). A metodologia propõe permitir situações de sobreposição dos dados porém, sua formulação busca minimizar esta medida. Os conceitos relacionados ao tratamento de dados não separáveis tais como a base utilizada na construção de fronteiras não lineares são apresentados na seção seguinte.

4.2 Classes Não Separáveis Linearmente

Sejam os dados de entrada do problema definidos através de N pares $(x_i, y_i)_{i=1}^N$, com $x_i \in \mathbb{R}^p$ e $y_i \in \{-1, +1\}$. Um hiperplano pode ser definido como

$$x : f(x) = x^T\beta + \beta_0 = 0, \tag{4.17}$$

onde $\|\beta\| = 1$. Uma regra induzida por $f(x)$ é

$$G(x) = \text{sign}\,[x^T\beta + \beta_0]. \tag{4.18}$$

Na seção anterior, observamos que $f(x)$ representa a distância de um ponto x até o hiperplano $f(x) = x^T\beta + \beta_0 = 0$. Considerando o caso separável, podemos obter uma função $f(x) = x^T\beta + \beta_0$ com $y_i f(x_i) > 0 \ \forall i$. Ou seja, é possível encontrar um hiperplano que cria a maior margem de separação entre os pontos de treinamento para as classes 1 e -1 conforme mostra a Figura 4.4.[6]

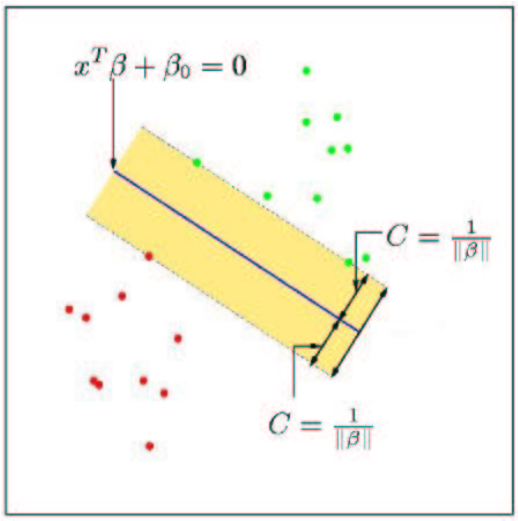


Figura 4.4: Classificação por Vetores de Suporte - Caso Separável.

[6]Fonte: Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, 2001.

O problema de otimização descrito na Equação (4.19) captura este conceito.

$$\begin{aligned} &\max_{\beta,\beta_0,\|\beta\|=1} \quad C \\ &\text{sujeito a } \; y_i({x_i}^T\beta + \beta_0) \geq C, \; i = 1, \ldots, N. \end{aligned} \tag{4.19}$$

Na Figura 4.4, a espessura da margem de separação é dada por $2C = 2/\|\beta\|$. Desta maneira, utilizando a relação $C = 1/\|\beta\|$ vamos, por conveniência, reformular o problema de maximização descrito em (4.19) como um problema de minimização

$$\begin{aligned} &\min_{\beta,\beta_0} \quad \|\beta\| \\ &\text{sujeito a } \; y_i({x_i}^T\beta + \beta_0) \geq 1, \; i = 1, \ldots, N. \end{aligned} \tag{4.20}$$

O problema de otimização quadrática com restrições de desigualdade descrito na Equação (4.20) representa a base conceitual dos vetores de suporte para os casos linearmente separáveis sendo que a solução foi apresentada na seção anterior.

Vamos agora considerar o caso não separável. Uma maneira de tratar a sobreposição dos dados continua sendo maximizar $\|C\|$ permitindo que alguns deles estejam no lado incorreto da margem de separação. Vamos então definir algumas variáveis de folga, denotadas por $\xi = (\xi_1, \xi_2, \ldots, \xi_n)$, possibilitando modificar o problema descrito em (4.19):

$$y_i({x_i}^T\beta + \beta_0), \quad \geq \quad C - \xi_i, \tag{4.21}$$

ou

$$y_i({x_i}^T\beta + \beta_0), \quad \geq \quad C(1 - \xi_i), \tag{4.22}$$

$\forall i, \xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq$ constante. As duas formas conduzem a diferentes soluções sendo que a segunda escolha representa a abordagem padrão para classificação através do uso dos vetores de suporte. Sendo assim, adotaremos esta escolha.

Desta maneira, as variáveis de folga ξ_i medem os desvios dos pontos $(x_i, y_i)_{i=1}^{N}$ para a condição ideal de separação das classes. A interpretação para o problema é de que, quando $\xi_i > 1$, o ponto encontra-se no lado incorreto do hiperplano de separação.

Conforme critério adotado para a Equação (4.10) da seção anterior, vamos alterar a restrição em β definindo $C = 1/\|\beta\|$. Desta forma, a Equação (4.20) pode ser escrita na forma equivalente

$$\min \|\beta\| \; \text{ sujeito a } \; \begin{cases} y_i({x_i}^T\beta + \beta_0) \geq 1 - \xi_i \forall i, \\ \xi_i \geq 0, \sum \xi_i \leq \text{constante}. \end{cases} \tag{4.23}$$

A Equação (4.23) representa a definição usual para a abordagem de classificação através dos vetores de suporte considerando o caso não separável. Desta maneira, as variáveis de folga ξ_i medem os desvios dos pontos $(x_i, y_i)_{i=1}^N$ para a condição ideal de separação das classes. A interpretação para o problema é de que, quando $\xi_i > 1$, o ponto encontra-se no lado incorreto do hiperplano de separação conforme ilustra a Figura 4.5.[7]

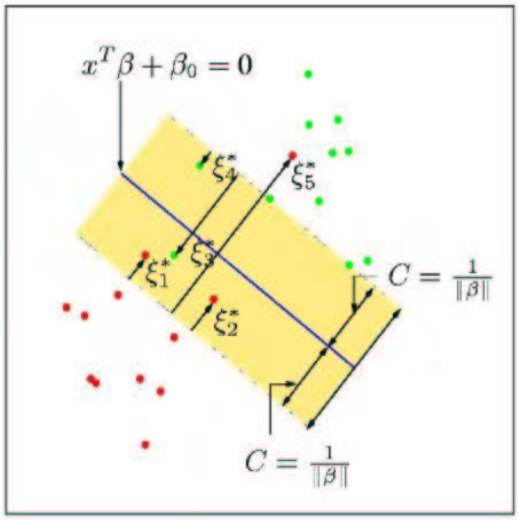


Figura 4.5: Classificação por Vetores de Suporte - Caso Não Separável.

Os pontos representados por ${\xi_i}^*$ encontram-se no lado incorreto de suas margens a uma quantia ${\xi_j}^* = C\xi_j$. Para os pontos presentes no lado correto de suas margens temos ${\xi_j}^* = 0$. A margem de separação é maximizada sujeita a um total de $\sum \xi_i \leq$ constante. Desta forma, ${\xi_j}^*$ corresponde à distância total definida pelos pontos localizados no lado incorreto de suas margens.

Considerando a Equação (4.23), retornamos ao problema de otimização convexa que, como também formulado no caso separável, é representado por uma função quadrática com restrições de desigualdade. Para solução deste problema, vamos utilizar os multiplicadores de Lagrange. Computacionalmente, é conveniente reescrever (4.23) na forma

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$
$$\text{sujeito a } \xi_i \geq 0,\ y_i({x_i}^T\beta x_i + b) \geq 1 - \xi_i \forall i, \tag{4.24}$$

[7]Fonte: Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, 2001.

onde γ substitui a constante em (4.23). Para o caso separável, $\gamma = \infty$. O problema primal é então descrito através do Lagrangiano

$$L_P = \frac{1}{2}\|\beta\|^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[y_i({x_i}^T \beta + \beta_0) - (1 - \xi_i) \right] - \sum_{i=1}^{N} \mu_i \xi_i. \tag{4.25}$$

Calculando as respectivas derivadas em relação a β, β_0, ξ_i e igualando os resultados a zero, obtemos

$$\begin{aligned} \beta &= \sum_{i=1}^{N} \alpha_i y_i x_i, & (4.26) \\ 0 &= \sum_{i=1}^{N} \alpha_i y_i, & (4.27) \\ \alpha_i &= \gamma - \mu_i, \forall i, & (4.28) \end{aligned}$$

onde α_i, μ_i, ξ_i são todas positivas $\forall i$.

Substituindo as equações anteriores em (4.25), chegamos ao problema dual

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} {x_i}^T x_{i'}. \tag{4.29}$$

A idéia é maximizar L_D sujeito a $0 \leq \alpha_i \leq \gamma$ e $\sum_{i=1}^{N} \alpha_i y_i = 0$. Somada às equações (4.26), (4.27) e (4.28), as condições de Karush-Kuhn-Tucker também consideram as restrições

$$\begin{aligned} \alpha_i \left[y_i({x_i}^T \beta + \beta_0) - (1 - \xi_i) \right] &= 0, & (4.30) \\ \mu_i \xi_i &= 0, & (4.31) \\ y_i({x_i}^T \beta + \beta_0) - (1 - \xi_i) &\geq 0, & (4.32) \end{aligned}$$

para $i = 1, \ldots, N$. Juntas, as equações (4.26) a (4.32), são suficientes para caracterizar a solução, tanto do problema primal quanto do problema dual.

A interpretação para o problema dual é que este permite trabalharmos em um espaço de dimensão elevada dado que os número de parâmetros ajustados independem do número de atributos utilizados (dimensão dos dados de entrada do problema).

A partir da Equação (4.26), observamos que a solução para β possui a forma

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha_i}\, y_i\, x_i, \tag{4.33}$$

cujos coeficientes α_i, considerando apenas as observações i nos quais as restrições em (4.32) são satisfeitas, são todos não nulos.

Tais observações são conhecidas como *vetores de suporte*, pelo fato de que $\hat{\beta}$ é representado unicamente em termos destas observações. Alguns pontos de suporte podem se localizar exatamente na linha que determina a margem da fronteira (caso em que $\xi_i = 0$) e, portanto, (4.31) e (4.28) serão caracterizadas por $0 \leq \alpha_i \leq \gamma$. Para as demais observações ($\xi_i = 0$) temos $\alpha_i = \gamma$. A partir da Equação (4.30), é possível observar que quaisquer pontos presentes na margem (caso $\xi_i = 0$, $0 \leq \alpha_i$) podem ser utilizados para a solução de β_0.

Tanto expressão dual descrita por L_D, quanto a primal descrita por L_P são formulações simples de problemas otimização convexa em funções quadráticas. Sendo assim, podem ser resolvidos numericamente através de técnicas conhecidas[24] sem grandes implicações.

Dadas as soluções $\hat{\beta}$ e $\hat{\beta}_0$, a função de decisão do problema pode ser escrita como

$$\begin{aligned} \hat{G}(x) &= \text{sign}\,[\hat{f}(x)] && (4.34)\\ &= \text{sign}\,[x_i{}^T\beta + \beta_0] && (4.35) \end{aligned}$$

cuja interpretação está relacionada ao sinal devolvido pela função que representa a equação da margem de separação, ou seja, $f(x) = \pm 1$. A partir deste sinal, é possível identificar como a observação foi classificada e, desta maneira, avaliar a capacidade de generalização do modelo.

O parâmetro de ajuste associado à metodologia é representado por γ. Para estimação do valor ótimo de γ, pode ser utilizada a metodologia de *cross-validation*[12] cuja abordagem não será considerada neste trabalho.

4.2.1 Derivação da SVM para o Problema de Classificação

A metodologia de classificação por vetores de suporte busca resolver o problema de separação de classes não-lineares através da obtenção de fronteiras lineares em um espaço característico de dimensão elevada. Para a realização destas transformações, a técnica SVM faz o uso de funções baseadas no produto interno kernel, originalmente utilizado por Aizerman et al.[25], [26]. O produto interno kernel permite a realização de um mapeamento não-linear do conjunto de dados de entrada para um espaço característico de alta dimensão, no qual é possível construir o hiperplano ótimo de separação.

Visando assimilar esta idéia, ilustramos um exemplo simples de mapeamento não-linear dos dados de entrada pertencentes ao espaço original para um espaço característico conforme Figura 4.6. Para fins de visualização, tanto o espaço original (a) quanto o espaço característico (b) são supostamente bidimensionais. Entretanto, sabe-se na prática que o espaço característico possui dimensão muito superior à do espaço original.

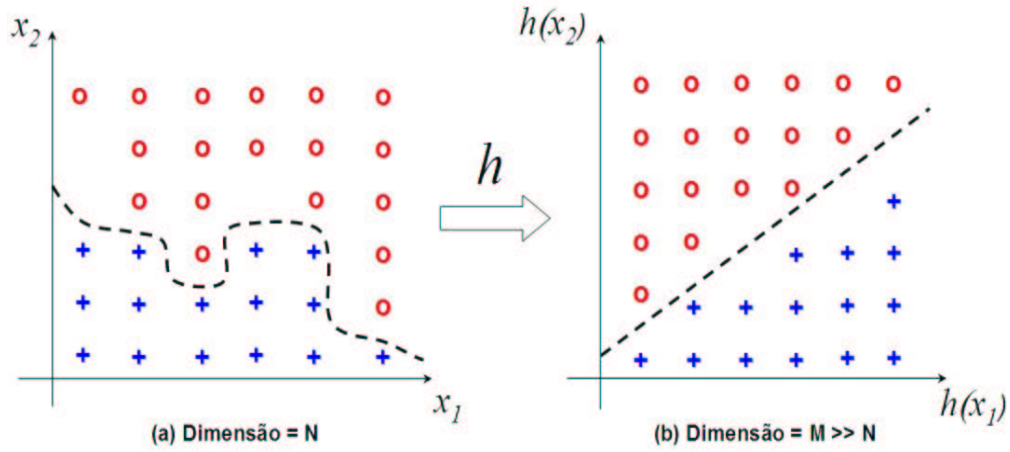


Figura 4.6: Exemplo de Mapeamento para o Espaço Característico.

A idéia é ajustar os dados originais $x_i = (x_1, x_2, \ldots, x_N)$, $i = 1, \ldots, N$ de maneira que estes sejam transformados em entradas do tipo $h_m(x_i) = (h_1(x_i), h_2(x_i), \ldots, h_M(x_i))$, $m = 1, \ldots, M$ para um espaço característico onde $M >> N$.

A partir destas transformações, são obtidas funções não lineares $\hat{f}(x) = h(x)^T\hat{\beta} + \hat{\beta}_0$ sendo que o classificador $\hat{G}(x) = \text{sign}\,(\hat{f}(x))$, já apresentado em (4.34), permanece o mesmo. A abordagem de classificação utilizada nas SVMs é uma extensão desta idéia e será formulada nos parágrafos seguintes.

Vamos novamente considerar a Equação (4.25) que descreve o problema dual. Tanto o problema de otimização quanto a sua solução podem ser representados em termos de funções $h(x_i)$ baseadas em produto interno kernel. Sendo assim, o problema dual (4.29) pode ser expresso na forma

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle. \tag{4.36}$$

Considerando a Equação (4.26), a solução de $f(x)$ pode ser expressa como

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i \, y_i \, \langle h(x), h(x_i) \rangle + \beta_0 \end{aligned} \tag{4.37}$$

Novamente, dados os valores de α_i, o valor de β_0 pode ser determinado resolvendo-se $f(x) = 0$ na Equação (4.37) para quaisquer valores x_i nos quais $0 < \alpha_i < \gamma$. Entretanto, é necessário especificar os as funções $h(x)$ envolvidas nas equações (4.36) e (4.37). Para tal, é apenas necessário conhecer as funções kernel

$$K(x, x') = \langle h(x), h(x') \rangle \tag{4.38}$$

responsáveis pelo cálculo dos produtos internos no espaço transformado. Para utilização da função no modelo de classificação, é necessário que K seja uma função positiva semi-definida. Os tipos mais comuns de produto interno utilizados nas SVMs são:

- Polinomial : $K(x, x') = \left(x'^T x + 1\right)^d$ (4.39)
- Base Radial : $K(x, x') = \exp(-\|x - x'\|^2 / 2\sigma^2)$ (4.40)
- *Perceptron* : $K(x, x') = \tanh(k_1 \, x'^T x + k_2)$ (4.41)

onde os parâmetros d, σ, k_1 e k_2 podem ser especificados a priori pelo usuário.

Vamos supor uma função kernel polinomial de grau 2 em nosso exemplo, apresentado na Figura 4.6, com apenas as entradas x_1 e x_2 no espaço original. Então

$$\begin{aligned} K(x, x') &= \left(x'^T x + 1\right)^2 \\ &= \left(x_1 \, x_{1'} + x_2 \, x_{2'} + 1\right)^2 \\ &= 2 \, x_1 \, x_{1'} \, x_2 \, x_{2'} + \left(x_1 \, x_{1'}\right)^2 + \left(x_2 \, x_{2'}\right)^2 + 2 \, x_1 \, x_{1'} + 2 \, x_2 \, x_{2'} + 1. \end{aligned} \tag{4.42}$$

Desta maneira, podemos expressar a imagem do vetor de entrada x induzido no espaço característico como

$$\begin{aligned} h^T(x) &= (1, \quad {x_1}^2, \quad \sqrt{2}\,x_1\,x_2, \quad {x_2}^2, \quad \sqrt{2}\,x_1, \quad \sqrt{2}\,x_2) \\ h^T(x') &= (1, \quad {x_{1'}}^2, \quad \sqrt{2}\,x_{1'}\,x_{2'}, \quad {x_{2'}}^2, \quad \sqrt{2}\,x_{1'}, \quad \sqrt{2}\,x_{2'}) \end{aligned}$$

Portanto, o espaço bidimensional das entradas do nosso exemplo, quando aplicado a um produto interno kernel do tipo polinomial de grau 2, é mapeado para um espaço característico de dimensão 6. Podemos então, escrever a representação final da solução, descrita em (4.37) em termos do produto interno

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i\, y_i\, K(x, x_i) + \hat{\beta}_0. \tag{4.43}$$

Resta-nos apenas converter o resultado acima para a abordagem probabilística uma vez que o interesse é classificar a amostra de dados em duas classes. Utilizando o classificador definido por $\hat{G}(x) = \text{sign}\,(\hat{f}(x))$ como nossa função de decisão, obtemos a resposta da SVM que pode ser interpretada da seguine maneira:

- Se é retornado um valor negativo, o ponto x pertence à classe negativa (-1);
- Se é retornado um valor positivo, o ponto x pertence à classe positiva (+1).

4.3 Aplicação e Resultados

Através desta seção, descreveremos a metodologia empregada na construção dos modelos de classificação baseados nas SVMs. O processo de familiarização com a técnica bem como todos os resultados apurados são apresentados.

Para a implementação desta técnica utilizamos uma *toolbox* de SVM em *Matlab* disponibilizada gratuitamente pelo Dr. Gavin Cawley, University of East Anglia[27].

4.3.1 Simulação da Técnica SVM com Dados Fictícios

Uma etapa anterior à modelagem dos dados de entrada, através da técnica SVM, foi simular diversos modelos de separação com dados arbitrários. Este processo teve grande importância, uma vez que nos forneceu familiaridade com o algoritmo e com os parâmetros ajustáveis do modelo. Para a realização das simulações, utilizou-se como produtos internos kernel as funções de base radial (RBF) e a polinomial. Este processo é descrito em detalhes nesta seção.

Inicialmente definimos, arbitrariamente, um exemplo de conjunto de dados no $\mathbb{R}^2$ para utilização da técnica. A Figura 4.7 apresenta estes dados representados por 48 pontos, supostamente classificados como *bons* e *ruins*. Para o exemplo em questão, existem sobreposições de dados e, desta maneira, podemos supor que não sejam separáveis linearmente.

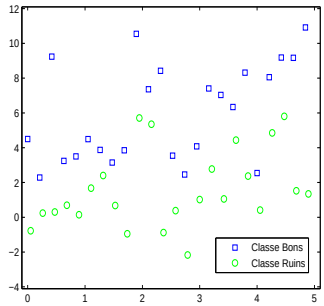


Figura 4.7: Exemplo de Classes Não Separáveis Linearmente.

A idéia inicial foi simular a técnica utilizando, como produto interno kernel, a função de base radial (RBF). Realizamos 5 simulações, variando-se o parâmetro σ da Equação (4.40). As margens de separação assim como os vetores de suporte são apresentados na Figura 4.8.

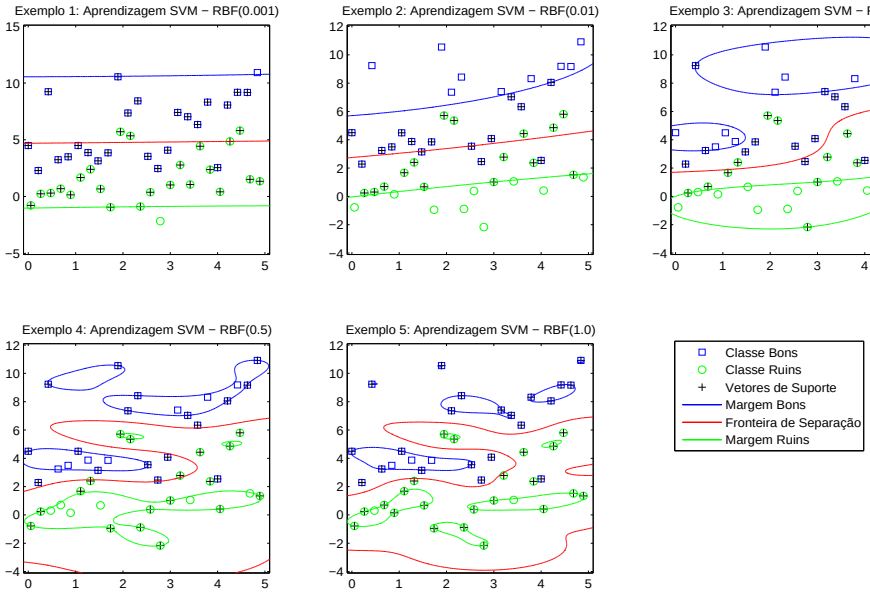


Figura 4.8: Exemplos de SVMs Utilizando Funções de Base Radial (RBF).

A cada simulação realizada, foram coletadas informações referentes ao número de vetores de suporte assim como as taxas de acerto para cada parâmetro σ variado. Tais informações são são apresentadas na tabela seguinte.

σ	Número de Vetores de Suporte	Taxa de Acerto (%)
0.001	46	81.3
0.01	30	93.8
0.1	28	97.9
0.5	35	95.8
1.0	43	97.9

Analogamente, agora utilizando como produto interno kernel a função polinomial, realizamos 5 simulações variando-se o grau do polinômio d da Equação (4.39). As margens de separação assim como os vetores de suporte são apresentados na Figura 4.9.

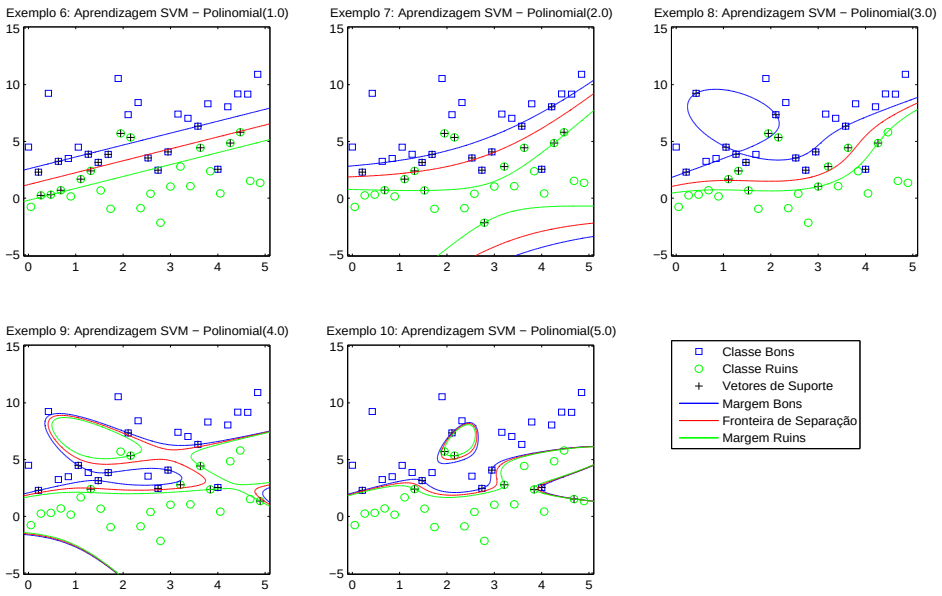


Figura 4.9: Exemplos SVMs Utilizando Funções Polinomiais.

Novamente realizamos a coleta das informações referentes ao número de vetores de suporte assim como as taxas de acerto para cada parâmetro d variado. Tais informações são são apresentadas na tabela seguinte.

d	Número de Vetores de Suporte	Taxa de Acerto (%)
1.0	20	97.9
2.0	20	95.8
3.0	20	91.7
4.0	15	97.9
5.0	12	100.0

Aplicando o modelo SVM neste exemplo de problema de classificação identificamos que, a cada acréscimo nos parâmetros ajustáveis, σ ou d, a técnica busca cada vez mais encontrar uma fronteira não linear para separação dos dados. No entanto, quanto ao número de vetores de suporte e às taxas de acerto, identificamos sensível melhora porém, não sendo suficiente identificar uma relação com os valores dos parâmetros. A princípio, observamos taxa de acerto superior quando utilizadas as funções polinomiais como produtos internos.

Entretanto, temos que considerar que em nosso modelo original existem 14 variáveis de entrada e portanto, 14 dimensões. Isto nos permite identificar que a aplicação da técnica em nossos dados reais pode retornar resultados nada similares aos exemplificados. Outro aspecto importante observado foi o fato de que o algoritmo, principalmente quando adotadas as funções polinomiais, já apresentava sinais de performance crítica, ainda que com apenas 48 pontos num espaço de entradas bidimensional.

4.3.2 Seleção e Tratamento dos Dados de Inferência

Os processos de coleta e tratamento dos dados de entrada do modelo foram os mesmos adotados na modelagem com RNAs. Utilizamos as mesmas amostras com dados normalizados contendo, em cada uma delas, as 14 variáveis de entrada já conhecidas. Os critérios safra de observação, período histórico, performance e escoragem permaneceram os mesmos.

O único ponto a ser observado é que, em função da performance computacional já identificada em nossos dados fictícios, adotamos utilizar amostras não mais com 10.000 registros cada, conforme simulações realizadas através dos métodos logístico e neural. Desta maneira, foram selecionadas aleatóriamente 10 amotras disjuntas e pareadas, agora com 5.000 registros cada. A variável resposta para as 2.500 transações decorrentes de clientes inadimplentes foram definidas como (-1) e para as 2.500 transações de clientes adimplentes como (+1).

4.3.3 Análise dos Resultados do Modelo

Todas as simulações deste estudo foram realizadas utilizando-se um computador pessoal com processador Intel Centrino Duo de 1,6GHz e 1GB de memória. O tempo médio apurado para as simulações envolvendo tanto a regressão logística quanto as redes neurais foi de aproximadamente 15s. Para a metodologia SVM, lembrando que as simulações foram realizadas com metade do tamanho das amostras originais, o tempo médio foi de aproximadamente 30 minutos.

Um fato importante a ser observado foi que, considerando os dados de entrada do modelo proposto, não houve convergência do algoritmo quando utilizamos as funções polinomiais como produtos internos kernel. Sendo assim, adotamos em nosso estudo somente as funções de base radial. Desta forma, o passo seguinte foi identificar o parâmetro de ajuste mais apropriado para a função kernel do modelo. Para tal, realizamos um processo similar ao realizado nas RNAs multicamada, onde a idéia era encontrar o número apropriado de neurônios na camada escondida em função do erro apurado em cada simulação.

Desta forma, utilizando os dados de entrada originais e considerando 10 combinações aleatórias de amostras de treino e teste, foram realizadas 10 simulações SVM. A cada simulação, variamos o valor do parâmetro σ e apuramos os erros médios e desvios das 10 combinações. O erros bem como os desvios, coletados através das matrizes de confusão, são apresentados na Figura 4.10.

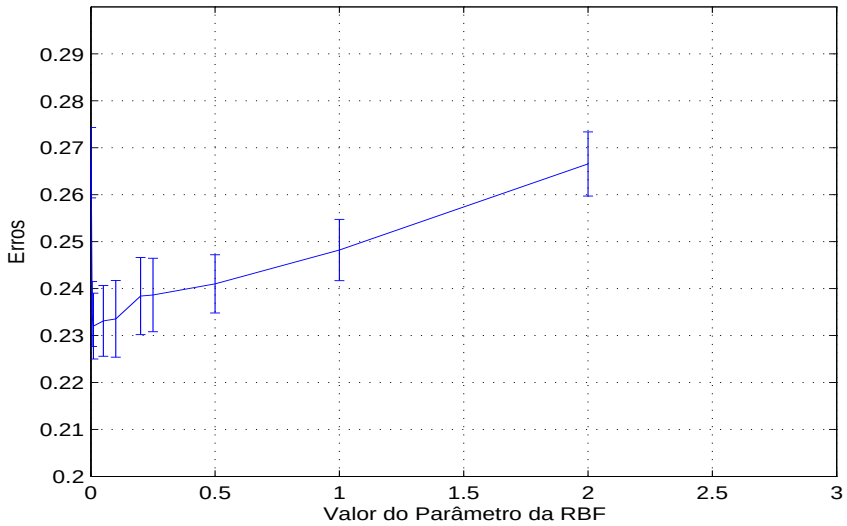


Figura 4.10: Evolução dos Erros de Classificação - SVM com Função de Base Radial (RBF).

Analisando a Figura 4.10, identificamos que para o nosso modelo proposto, os erros de classificação pioram conforme variamos o valor do parâmetro σ da RBF. Sendo assim, escolhemos trabalhar com uma RBF de parâmetro $\sigma = 0.01$.

Desta forma, partimos para o treinamento das SVMs, novamente variando as amostras de treino e teste, duas a duas. Foram realizadas 10 simulações independentes e o ponto de corte classificar as transações em boas ou ruins foi agora o valor 0.

Para avaliação da performance do modelo neural multicamada, utilizamos novamente os indicadores apurados através das matrizes de confusão bem como a estatística KS e ordenação das classes. A partir das 10 simulações de treinamento e teste, foram apurados os resultados permitindo obter um valor médio de 76,81% de taxa de acerto. A matriz de confusão referente à simulação cujo erro geral mais se aproximou do valor médio é apresentada na Figura 4.11.

		CLASSE VERDADEIRA		
		RUINS	BONS	TOTAL
CLASSE DO MODELO	RUINS	1.962	620	2.582
	BONS	538	1.880	2.418
	TOTAL	2.500	2.500	5.000

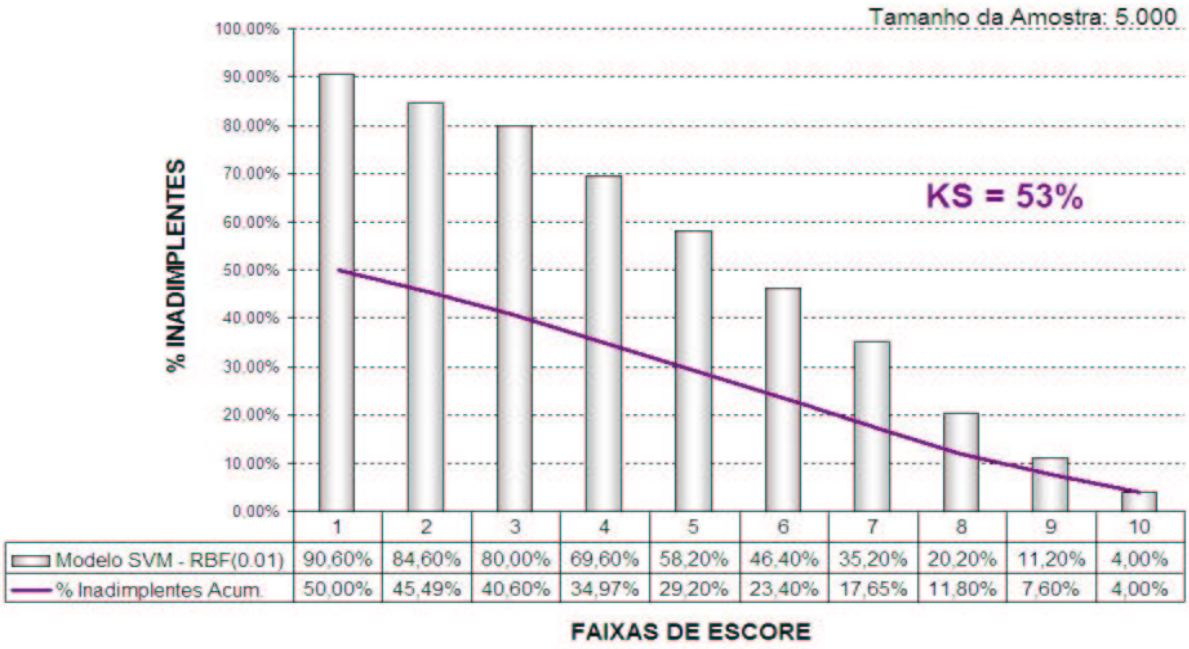
ERROS	
Erro I	0,2480
Erro II	0,2152
Erro Geral	0,2316

Figura 4.11: Matriz de Confusão - Modelo SVM com RBF(0.01).

A matriz de confusão da Figura 4.11 apresenta taxa de acerto de 76,84% para o modelo proposto. Comparando os erros tipo I e II, tivemos erro tipo I maior que o erro tipo II. É interessante lembrar que, este tipo de comportamento nos erros também ocorreu no processo de modelagem dos dados utilizando-se uma RNA multicamada com um neurônio na camada escondida que, conforme mencionado no início do capítulo, é o tipo de arquitetura de máquina que mais se aproxima de uma SVM.

Entretanto, vamos confirmar a consistência dos resultados através dos demais indicadores de performance.

A estatística KS bem como a curva de ordenação são apresentados na Figura 4.12.



	1	2	3	4	5	6	7	8	9	10
Modelo SVM - RBF(0.01)	90,60%	84,60%	80,00%	69,60%	58,20%	46,40%	35,20%	20,20%	11,20%	4,00%
% Inadimplentes Acum.	50,00%	45,49%	40,60%	34,97%	29,20%	23,40%	17,65%	11,80%	7,60%	4,00%

Figura 4.12: Desempenho do Modelo de Classificação SVM com RBF(0.01).

Os indicadores de ordenação e KS também apresentam resultados adequados para o modelo proposto. Através da taxa de acerto de 76,84% e estatística KS de 53% confirmamos desempenho similar aos modelos logístico e RNA camada única.

Capítulo 5

Conclusões

Neste trabalho estudamos a aplicação de modelos de *behavior scoring* em um problema de reconhecimento de padrões com 2 classes distintas. O modelo proposto baseia-se em classificar transações com cartão de crédito como boas ou ruins, tendo como base o risco de inadimplência. Foram utilizados como dados de entrada informações sobre o produto cartão de crédito, sobre as transações, informações comportamentais e cadastrais dos clientes.

A idéia foi comparar o desempenho de três técnicas diferentes de modelagem. Inicialmente foi desenvolvido um modelo utilizando a metodologia de regressão logística, considerada a mais conhecida. Em seguida, foram desenvolvidos novos modelos a partir das técnicas de aprendizagem de máquina: redes neurais artificiais, arquiteturas uma camada e multicamada, e máquina de vetores de suporte utilizando como produto interno kernel a função de base radial.

O desempenho de cada uma das técnicas foi avaliado em função do poder de discrimação obtido em cada modelo desenvolvido. A partir dos resultados, foram desenvolvidos indicadores de desempenho de maneira a mensurar o poder de separação das classes. Além dos indicadores já utilizados pelo mercado, como é o caso das matrizes de confusão e o teste Kolmogorov-Smirnov, também desenvolvemos um indicador que permite identificar graficamente se existe ordenação adequada dos dados segundo a probabilidade de inadimplência.

É apresentada na Figura 5.1 uma tabela com os indicadores de desempenho obtidos em cada uma das técnicas, permitindo realizar a comparação entre elas.

Metodologias	Regressão Logística (RL)	RNA Uma Camada (LP)	RNA Multicamada 1 Neurônio (MLP)	Máquina de Vetores de Suporte (SVM)
Erro Tipo I	0,2148	0,2132	0,2366	0,2480
Erro Tipo II	0,2470	0,2550	0,1942	0,2152
Erro Geral	0,2309	0,2341	0,2154	0,2316
Taxa de Acerto	76,91%	76,59%	78,46%	76,84%
KS	54,26%	53,54%	55,30%	53,20%

Ordenação	% Ruins (RL)	% Ruins (LP)	% Ruins (MLP)	% Ruins (SVM)
Faixa 1	90,19%	89,89%	89,47%	90,60%
Faixa 2	84,60%	85,10%	84,83%	84,60%
Faixa 3	79,00%	78,30%	80,00%	80,00%
Faixa 4	72,00%	71,80%	72,52%	69,60%
Faixa 5	59,90%	58,80%	61,29%	58,20%
Faixa 6	46,80%	47,50%	47,12%	46,40%
Faixa 7	32,60%	33,60%	32,06%	35,20%
Faixa 8	19,70%	19,60%	18,31%	20,20%
Faixa 9	9,80%	10,00%	8,52%	11,20%
Faixa 10	5,49%	5,49%	3,67%	4,00%

Figura 5.1: Quadro Comparativo - Indicadores de Desempenho.

Analisando os resultados obtidos em todas as técnicas, identificamos melhor desempenho no modelo neural multicamada. A arquitetura escolhida foi a *feed-foward* com um neuroônio na camada escondida sendo que foi adotada a função $\varphi(z) = \tanh(z)$ para a camada interna e a função $\varphi_0(z) = 1/(1 + e^{-z})$ para a camada externa. Indicadores como taxa de acerto de 78,46% e KS de 55% mostram desempenho superior em relação aos índices obtidos nos modelos baseados em regressão logística, rede neural com uma camada e SVM.

Entretanto, dada a nossa proposta inicial de desenvolvimento de um instrumento eficiente para classificação de padrões, consideramos que todas as metodologias apresentaram resultados consistentes. Desta maneira, entendemos que qualquer uma das técnicas de modelagem pode ser adotada para avaliação de risco de inadimplência, considerando-se o problema proposto. O que poderíamos sugerir ao usuário é que, na escolha do modelo apropriado para avaliação de riscos em transações com cartão de crédito, tome-se em consideração não somente a precisão do modelo mas principalmente a eficiência computacional.

Considerando os modelos desenvolvidos, observamos eficiência computacional somente nas técnicas de regressão logística e redes neurais. Na metodologia SVM, identificamos elevado tempo de processamento exigido, apesar de atingidas as convergências utilizando-se as funções de base radial. Outro aspecto crítico observado foi a não convergência do algoritmo quando utilizamos as funções polinomiais para mapeamento do espaço característico.

No entanto, a idéia do estudo não visou o questionamento dos algoritmos ou estratégias de implementação computacional utilizadas. O que se buscou foi uma análise crítica e pragmática dos algoritmos existentes, de maneira a viabilizar sua aplicação em novos problemas.

Desta forma, dado o fato de a metodologia SVM forneceu os resultados menos expressivos para o problema proposto, poderíamos apenas sugerir a realização de novas pesquisas visando refinarmos ainda mais os estudos sobre a metodologia SVM, assim como comprovar melhores resultados. Apresentamos algumas questões em aberto como possíveis extensões deste trabalho:

• Visando minimizar o elevado custo computacional utilizado pelas SVMs para resolver o problema quadrático de minimização, uma futura proposta seria utilizar as *Least Squares Support Vector Machines* (LS-SVMs). As LS-SVMs correspondem a modificações das SVMs cuja proposta é usar uma função objetivo de mínimos quadrados com restrições de igualdade. O treinamento é realizado resolvendo-se um sistema de equações lineares ao invés de programação quadrática. Muitos trabalhos já foram realizados utilizando esta metodologia. Resultados consistentes foram comprovados por Semolini[28] e Carvalho[29].

• Outra possibilidade em termos de menor esforço computacional, seria sugerir a utilização do algoritmo SVM^{light} proposto por Joachims[30]. Este algoritmo baseia-se na decomposição do problema de otimização em uma série de problemas menores, de maneira que cada pequeno problema possa ser resolvido de maneira mais rápida. Resultados interessantes podem ser identificados através do trabalho de Semolini[28].

• Visto que, em nossas simulações arbitrárias, conseguimos visualizar melhores resultados quando utilizamos as funções polinomiais como produtos internos, segue também como futura proposta a implementação de um algoritmo eficiente para resolução de problemas desta natureza.

• Entretanto, o fato mais importante a ser questionado ainda decorre do pragmatismo existente nos dias de hoje quando problemas desta natureza são resolvidos através de metodologias baseadas em aprendizado de máquina, como é o caso das redes neurais e das máquinas de vetores de suporte. Até então, quase não existem estudos viabilizando interpretar qual é a influência de cada atributo no modelo final.

Apêndice A

Resultados Regressão Logística

Para a implementação da metodologia, utilizamos o módulo *Enterprise Miner* do aplicativo computacional SAS. A estrutura do modelo é apresentada na Figura A.1.

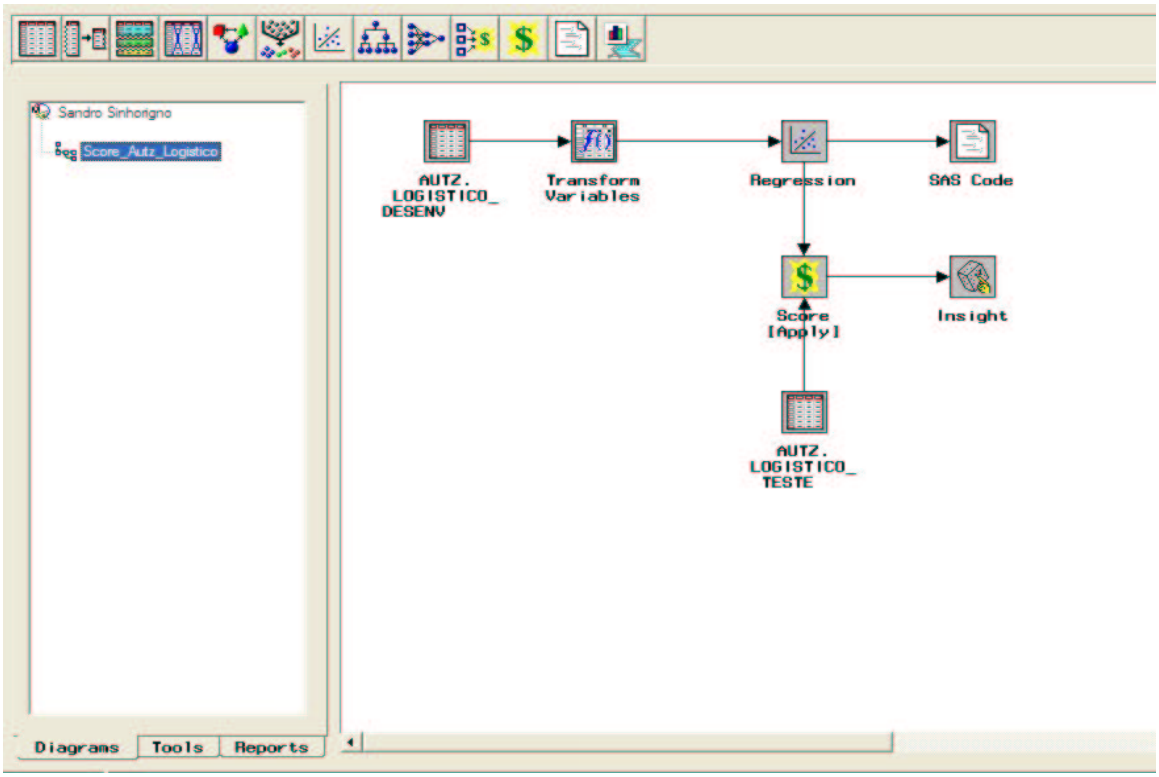


Figura A.1: Arquitetura do Modelo Logístico Implementado em SAS.

Na Figura A.2 é apresentada uma tabela geral com os resultados das matrizes de confusão em todas as simulações. Em seguida, são listadas todas as variáveis selecionadas pelo método *stepwise*.

Simulacoes	Desenvolv. 9 & Validacao 8	Desenvolv. 1 & Validacao 2	Desenvolv. 8 & Validacao 10	Desenvolv. 7 & Validacao 4	Desenvolv. 2 & Validacao 5
Erro Tipo I	0,2144	0,2206	0,2324	0,2128	0,2058
Erro Tipo II	0,2316	0,2454	0,2428	0,2536	0,2476
Erro Geral	0,2230	0,2330	0,2376	0,2332	0,2267
Taxa de Acerto	77,70%	76,70%	76,24%	76,68%	77,33%

Simulacoes	Desenvolv. 3 & Validacao 6	Desenvolv. 4 & Validacao 1	Desenvolv. 5 & Validacao 9	Desenvolv. 6 & Validacao 7	Desenvolv. 10 & Validacao 3	Valores Medios	Desvios Padrao
Erro Tipo I	0,2256	0,2214	0,2148	0,2252	0,2318	**0,2205**	0,0086
Erro Tipo II	0,2374	0,2416	0,2470	0,2446	0,2186	**0,2410**	0,0099
Erro Geral	0,2315	0,2315	0,2309	0,2349	0,2252	**0,2308**	0,0045
Taxa de Acerto	76,85%	76,85%	76,91%	76,51%	77,48%	**76,93%**	0,0045

Figura A.2: Indicadores de Desempenho - Regressão Logística.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-3.233	0.9867	10.74	0.0010
var_discr_01	1	0.2394	0.0560	18.29	<.0001
var_discr_02	1	2.509	0.9106	7.59	0.0059
var_discr_02	1	2.256	0.9097	6.15	0.0131
var_discr_02	1	2.100	0.9087	5.34	0.0208
var_discr_02	1	1.491	0.9071	2.70	0.1004
var_discr_03	1	3.014	0.4018	56.25	<.0001
var_discr_03	1	2.430	0.2379	104.37	<.0001
var_discr_03	1	1.397	0.2339	35.64	<.0001
var_discr_03	1	0.1847	0.2734	0.46	0.4994
var_discr_05	1	1.496	0.1385	116.59	<.0001
var_discr_05	1	0.5940	0.1370	18.79	<.0001
var_discr_05	1	0.2403	0.0550	19.08	<.0001
var_discr_06	1	0.7208	0.3250	4.92	0.0266
var_discr_07	1	2.275	1.126	4.09	0.0432
var_cont_01	1	0.0887	0.0149	35.62	<.0001
var_cont_02	1	0.2600	0.0138	352.91	<.0001
var_cont_03	1	-0.0205	0.000983	436.21	<.0001
var_cont_04	1	-0.00310	0.000302	105.71	<.0001
var_cont_05	1	0.000159	0.000027	34.98	<.0001

Figura A.3: Modelo Logístico 1: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-4.183	1.032	16.44	<.0001
var_discr_01	1	0.2631	0.0560	22.08	<.0001
var_discr_02	1	3.439	0.9668	12.65	0.0004
var_discr_02	1	3.223	0.9652	11.15	0.0008
var_discr_02	1	3.020	0.9631	9.83	0.0017
var_discr_02	1	2.583	0.9573	7.28	0.0070
var_discr_03	1	4.028	0.4783	70.94	<.0001
var_discr_03	1	3.269	0.3474	88.54	<.0001
var_discr_03	1	2.188	0.3447	40.28	<.0001
var_discr_03	1	1.013	0.3675	7.59	0.0059
var_discr_05	1	1.553	0.1385	125.83	<.0001
var_discr_05	1	0.4236	0.1305	10.53	0.0012
var_discr_05	1	0.2943	0.0547	28.92	<.0001
var_cont_01	1	0.1131	0.0152	55.17	<.0001
var_cont_02	1	0.2424	0.0137	310.93	<.0001
var_cont_03	1	-0.0190	0.000953	398.98	<.0001
var_cont_04	1	-0.00343	0.000296	134.64	<.0001
var_cont_05	1	0.000152	0.000024	38.82	<.0001

Figura A.4: Modelo Logístico 2: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-3.608	1.134	10.13	0.0015
var_discr_01	1	0.2402	0.0562	18.27	<.0001
var_discr_02	1	2.624	1.059	6.14	0.0132
var_discr_02	1	2.378	1.058	5.05	0.0246
var_discr_02	1	2.308	1.057	4.76	0.0291
var_discr_02	1	1.641	1.055	2.42	0.1198
var_discr_03	1	3.511	0.4713	55.48	<.0001
var_discr_03	1	3.514	0.3670	91.66	<.0001
var_discr_03	1	2.421	0.3644	44.15	<.0001
var_discr_03	1	1.225	0.3890	9.92	0.0016
var_discr_05	1	1.237	0.1369	81.67	<.0001
var_discr_05	1	0.3544	0.1369	6.70	0.0097
var_discr_05	1	0.1349	0.0583	5.36	0.0207
var_cont_01	1	0.1071	0.0162	43.92	<.0001
var_cont_02	1	0.2792	0.0144	374.07	<.0001
var_cont_03	1	-0.0206	0.000991	432.18	<.0001
var_cont_04	1	-0.00316	0.000295	114.65	<.0001
var_cont_05	1	0.000210	0.000028	58.49	<.0001
var_cont_06	1	-0.00029	0.000113	6.74	0.0095

Figura A.5: Modelo Logístico 3: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-2.952	0.8035	13.50	0.0002
var_discr_01	1	0.2161	0.0560	14.88	0.0001
var_discr_02	1	2.175	0.7142	9.27	0.0023
var_discr_02	1	1.891	0.7130	7.03	0.0080
var_discr_02	1	1.689	0.7120	5.63	0.0177
var_discr_02	1	1.166	0.7105	2.69	0.1008
var_discr_03	1	3.538	0.4392	64.90	<.0001
var_discr_03	1	3.264	0.3529	85.51	<.0001
var_discr_03	1	2.315	0.3503	43.66	<.0001
var_discr_03	1	0.9907	0.3773	6.90	0.0086
var_discr_05	1	1.370	0.1387	97.51	<.0001
var_discr_05	1	0.5275	0.1404	14.12	0.0002
var_discr_05	1	0.1664	0.0571	8.48	0.0036
var_cont_01	1	0.0813	0.0148	30.36	<.0001
var_cont_02	1	0.2875	0.0144	400.38	<.0001
var_cont_03	1	-0.0191	0.000965	391.49	<.0001
var_cont_04	1	-0.00334	0.000303	121.48	<.0001
var_cont_05	1	0.000226	0.000028	64.45	<.0001
var_cont_06	1	-0.00024	0.000101	5.41	0.0200
var_cont_07	1	0.000552	0.000167	10.95	0.0009

Figura A.6: Modelo Logístico 4: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-5.580	1.088	26.31	<.0001
var_discr_01	1	0.2726	0.0562	23.50	<.0001
var_discr_02	1	2.760	0.9456	8.52	0.0035
var_discr_02	1	2.458	0.9448	6.77	0.0093
var_discr_02	1	2.423	0.9440	6.59	0.0103
var_discr_02	1	1.824	0.9420	3.75	0.0528
var_discr_03	1	4.360	0.4973	76.88	<.0001
var_discr_03	1	3.475	0.3900	79.39	<.0001
var_discr_03	1	2.531	0.3877	42.61	<.0001
var_discr_03	1	0.8734	0.4208	4.31	0.0379
var_discr_05	1	1.091	0.1321	68.15	<.0001
var_discr_05	1	0.3932	0.1390	8.00	0.0047
var_discr_05	1	0.1958	0.0549	12.70	0.0004
var_discr_06	1	1.420	0.3522	16.26	<.0001
var_cont_01	1	0.0808	0.0143	31.91	<.0001
var_cont_02	1	0.2708	0.0142	363.03	<.0001
var_cont_03	1	-0.0206	0.000987	434.44	<.0001
var_cont_04	1	-0.00257	0.000274	88.01	<.0001
var_cont_05	1	0.000187	0.000028	44.94	<.0001

Figura A.7: Modelo Logístico 5: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-8.569	16.858	0.26	0.6112
var_discr_01	1	0.3148	0.0554	32.28	<.0001
var_discr_02	1	7.988	16.852	0.22	0.6355
var_discr_02	1	7.760	16.852	0.21	0.6452
var_discr_02	1	7.536	16.852	0.20	0.6547
var_discr_02	1	7.050	16.852	0.18	0.6757
var_discr_03	1	3.448	0.6233	30.61	<.0001
var_discr_03	1	2.850	0.2798	103.79	<.0001
var_discr_03	1	2.912	0.2774	110.13	<.0001
var_discr_03	1	1.820	0.2707	45.20	<.0001
var_discr_03	1	0.5980	0.3020	3.92	0.0477
var_discr_05	1	1.449	0.1454	99.34	<.0001
var_discr_05	1	0.5733	0.1372	17.47	<.0001
var_discr_05	1	0.1578	0.0577	7.48	0.0063
var_discr_06	1	0.6993	0.3447	4.12	0.0425
var_cont_01	1	0.0832	0.0137	36.80	<.0001
var_cont_02	1	0.1721	0.0122	198.09	<.0001
var_cont_03	1	-0.0199	0.000975	416.85	<.0001
var_cont_04	1	-0.00414	0.000332	155.29	<.0001
var_cont_05	1	0.000254	0.000029	78.18	<.0001
var_cont_06	1	-0.00039	0.000123	10.27	0.0014

Figura A.8: Modelo Logístico 6: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-5.297	1.044	25.74	<.0001
var_discr_01	1	0.2976	0.0559	28.36	<.0001
var_discr_02	1	2.548	0.8339	9.33	0.0023
var_discr_02	1	2.402	0.8322	8.33	0.0039
var_discr_02	1	2.247	0.8302	7.33	0.0068
var_discr_02	1	1.635	0.8261	3.92	0.0478
var_discr_03	1	4.159	0.5114	66.13	<.0001
var_discr_03	1	3.699	0.4188	78.01	<.0001
var_discr_03	1	2.783	0.4168	44.59	<.0001
var_discr_03	1	1.272	0.4424	8.26	0.0040
var_discr_04	1	0.7308	0.2998	5.94	0.0148
var_discr_05	1	1.178	0.1271	85.92	<.0001
var_discr_05	1	0.5300	0.1368	15.02	0.0001
var_discr_05	1	0.2402	0.0590	16.55	<.0001
var_discr_06	1	0.6683	0.3097	4.66	0.0309
var_cont_01	1	0.0922	0.0155	35.37	<.0001
var_cont_02	1	0.2456	0.0134	336.60	<.0001
var_cont_03	1	-0.0190	0.000951	398.62	<.0001
var_cont_04	1	-0.00338	0.000300	126.90	<.0001
var_cont_05	1	0.000222	0.000031	50.61	<.0001
var_cont_06	1	-0.00035	0.000137	6.71	0.0096

Figura A.9: Modelo Logístico 7: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-4.093	0.8659	22.34	<.0001
var_discr_01	1	0.1228	0.0555	4.89	0.0270
var_discr_02	1	2.013	0.6894	8.53	0.0035
var_discr_02	1	1.829	0.6872	7.09	0.0078
var_discr_02	1	1.592	0.6852	5.40	0.0202
var_discr_02	1	1.150	0.6806	2.85	0.0912
var_discr_03	1	4.360	0.5242	69.18	<.0001
var_discr_03	1	3.738	0.4230	78.11	<.0001
var_discr_03	1	2.814	0.4209	44.69	<.0001
var_discr_03	1	1.467	0.4441	10.91	0.0010
var_discr_05	1	1.379	0.1319	109.42	<.0001
var_discr_05	1	0.4467	0.1314	11.56	0.0007
var_discr_05	1	0.2910	0.0544	28.62	<.0001
var_discr_06	1	0.7182	0.3117	5.31	0.0212
var_cont_01	1	0.1027	0.0144	50.57	<.0001
var_cont_02	1	0.2582	0.0140	341.91	<.0001
var_cont_03	1	-0.0179	0.000940	363.03	<.0001
var_cont_04	1	-0.00322	0.000290	123.40	<.0001
var_cont_05	1	0.000129	0.000027	22.92	<.0001

Figura A.10: Modelo Logístico 8: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-2.017	0.7459	7.31	0.0069
var_discr_02	1	1.748	0.6633	6.94	0.0084
var_discr_02	1	1.525	0.6617	5.31	0.0212
var_discr_02	1	1.210	0.6608	3.35	0.0671
var_discr_02	1	0.6639	0.6585	1.02	0.3133
var_discr_03	1	3.432	0.4593	55.84	<.0001
var_discr_03	1	2.842	0.3050	86.83	<.0001
var_discr_03	1	1.835	0.3019	36.96	<.0001
var_discr_03	1	0.8739	0.3264	7.17	0.0074
var_discr_05	1	1.521	0.1392	119.31	<.0001
var_discr_05	1	0.6584	0.1388	22.51	<.0001
var_discr_05	1	0.2546	0.0547	21.65	<.0001
var_cont_01	1	0.0594	0.0132	20.16	<.0001
var_cont_02	1	0.2850	0.0143	397.25	<.0001
var_cont_03	1	-0.0200	0.000971	423.72	<.0001
var_cont_04	1	-0.00323	0.000302	114.58	<.0001
var_cont_05	1	0.000200	0.000028	51.04	<.0001

Figura A.11: Modelo Logístico 9: Variáveis e Parâmetros Estatísticos.

Parameter	DF	Etimate	Standard Error	Wald Chi - Square	Pr > Chi - Square
Intercept	1	-4.611	1.102	17.51	<.0001
var_discr_01	1	0.2696	0.0561	23.12	<.0001
var_discr_02	1	2.457	0.9000	7.45	0.0063
var_discr_02	1	2.154	0.8984	5.75	0.0165
var_discr_02	1	1.978	0.8969	4.87	0.0274
var_discr_02	1	1.483	0.8931	2.76	0.0968
var_discr_03	1	4.994	0.6820	53.62	<.0001
var_discr_03	1	4.596	0.6035	58.00	<.0001
var_discr_03	1	3.449	0.6019	32.83	<.0001
var_discr_03	1	2.419	0.6149	15.48	<.0001
var_discr_05	1	1.602	0.1374	135.87	<.0001
var_discr_05	1	0.7663	0.1363	31.59	<.0001
var_discr_05	1	0.3323	0.0550	36.54	<.0001
var_cont_01	1	0.0467	0.0137	11.70	0.0006
var_cont_02	1	0.2669	0.0144	343.87	<.0001
var_cont_03	1	-0.0197	0.000971	411.75	<.0001
var_cont_04	1	-0.00310	0.000291	113.71	<.0001
var_cont_05	1	0.000170	0.000029	34.72	<.0001

Figura A.12: Modelo Logístico 10: Variáveis e Parâmetros Estatísticos.

Apêndice B

Resultados Redes Neurais Camada Única

No *Netlab* as RNAs de camada única são identificadas como *Generalized Linear Models* (GLMs). Sendo assim, o função para criação da rede, utilizando uma GLM, pode ser utilizada como:

net = glm(qtde entradas, qtde saídas, 'função de saída)

Neste trabalho, utilizamos 14 variáveis de entrada, 1 variável resposta e a função logística como função de saída. Desta maneira, a rede foi apresentada como:

net =
type: 'glm'
nin: 14
nout: 1
nwts: 15
outfn: 'logistic'
w1: [14x1 double]
b1: 0.1782

Para treinamento da rede, a sintaxe utilizada foi:

options = foptions;
[net,options]=netopt(net, options, base de entradas, variável resposta, 'função de otimização');

O método de otimização escolhido foi o Gradiente Conjugado Escalado (scg). Para validação do modelo, utilizamos a rede treinada na base de dados de validação utilizando o comando:

glmfwd(net,base de validação);

A partir desta função, obtemos o vetor de probabilidades com 10.000 registros associado à variável resposta submetida no modelo. Foram apurados 10 vetores no processo de validação e calcudados os erros. Os resultados obtidos bem como a performance do modelo são apresentados através de matrizes de confusão. Na Figura B.1 é apresentada uma tabela geral com os resultados de todas as matrizes de confusão.

Simulacoes	Treino 9 & Teste 8	Treino 1 & Teste 2	Treino 8 & Teste 10	Treino 7 & Teste 4	Treino 2 & Teste 5
Erro Tipo I	0,2190	0,2230	0,2074	0,2218	0,2342
Erro Tipo II	0,2318	0,2542	0,2544	0,2540	0,2464
Erro Geral	0,2254	0,2386	0,2309	0,2379	0,2403
Taxa de Acerto	77,46%	76,14%	76,91%	76,21%	75,97%

Simulacoes	Treino 3 & Teste 6	Treino 4 & Teste 1	Treino 5 & Teste 9	Treino 6 & Teste 7	Treino 10 & Teste 3	Valores Medios	Desvios Padrao
Erro Tipo I	0,2092	0,2234	0,2132	0,2210	0,2348	**0,2207**	0,0092
Erro Tipo II	0,2576	0,2480	0,2550	0,2488	0,2242	**0,2474**	0,0110
Erro Geral	0,2334	0,2357	0,2341	0,2349	0,2295	**0,2341**	0,0045
Taxa de Acerto	76,66%	76,43%	76,59%	76,51%	77,05%	**76,59%**	0,0045

Figura B.1: Indicadores de Desempenho - Rede Neural Camada Única.

Apêndice C

Resultados Redes Neurais Multicamada

As RNAs multicamada são identificadas no Netlab como redes *Multilayer Perceptrons* (MLPs). A função para criação de uma rede MLP é apresentada abaixo:

net = mlp(qtde entradas, qtde unidades ocultas, qtde saídas, 'função de saída)

Neste experimento, utilizamos 14 variáveis de entrada, 1 unidade oculta, 1 variável resposta. Para a camada intermediária, o aplicativo utiliza a função tangente hiperbólica. Na camada de saída, novamente adotamos a função logística.

A rede é apresentada no *Netlab* como:

net =
type: 'mlp'
nin: 14
nhidden: 1
nout: 1
nwts: 17
outfn: 'logistic'
w1: [14x1 double]
b1: 0.0566
w2: -0.6519
b2: -1.5349

Para treinamento da rede:

options = foptions;
[net,options]=netopt(net, options, base de entradas, variável resposta, 'função de otimização');

Para validação do modelo, utilizamos a rede treinada novamente em nossa base de validação através agora do comando:

mlpfwd(net,base de validação);

A partir desta função, também obtemos o vetor de probabilidades com 10.000 registros associado à variável resposta e o processo de validação foi análogo ao adotado na rede de camada única. Os resultados associados às matrizes de confusão para a rede MLP são apresentados na Figura C.1.

Simulacoes	Treino 1 & Teste 2	Treino 3 & Teste 6	Treino 5 & Teste 9	Treino 7 & Teste 4	Treino 8 & Teste 10
Erro Tipo I	0,2298	0,2274	0,2366	0,2432	0,2428
Erro Tipo II	0,2318	0,2156	0,1942	0,2016	0,1784
Erro Geral	0,2308	0,2215	0,2154	0,2224	0,2106
Taxa de Acerto	76,92%	77,85%	78,46%	77,76%	78,94%

Simulacoes	Treino 2 & Teste 5	Treino 6 & Teste 7	Treino 9 & Teste 8	Treino 4 & Teste 1	Treino 10 & Teste 3	Valores Medios	Desvios Padrao
Erro Tipo I	0,2704	0,2570	0,2474	0,2512	0,2734	**0,2479**	0,0155
Erro Tipo II	0,1726	0,1710	0,1704	0,1684	0,1502	**0,1854**	0,0249
Erro Geral	0,2215	0,2140	0,2089	0,2098	0,2118	**0,2167**	0,0071
Taxa de Acerto	77,85%	78,60%	79,11%	79,02%	78,82%	**78,33%**	0,0071

Figura C.1: Indicadores de Desempenho - Rede Neural Multicamada.

Apêndice D

Resultados Máquinas de Vetores de Suporte

Os resultados do experimento foram obtidos através da implementação da técnica SVM utilizando uma *Matlab SVM Toolbox*. No entanto, se fez necessária a utilização de um compilador C++. Adotamos o *Borland C/C++ (free command line tools) version 5.5*.

Para inicialização do SVM, é necessário escolher o tipo de produto interno kernel assim como o algoritmo de otimização. Inicialmente são apresentados na Figura D.1 os resultados obtidos na fase de escolha do parâmetro da função RBF para o modelo SVM proposto.

Parametro de Ajuste (RBF)	Simulacoes	Treino 9 & Teste 3	Treino 5 & Teste 10	Treino 10 & Teste 4	Treino 7 & Teste 1	Treino 8 & Teste 5	Treino 4 & Teste 8	Treino 1 & Teste 2	Treino 2 & Teste 6	Treino 3 & Teste 7	Treino 6 & Teste 9	Erros Medios	Desvios Padrao
σ=0,001	Erro Tipo I	0,1528	0,1276	0,1424	0,1452	0,1576	0,1288	0,1148	0,1476	0,1456	0,1580	0,1420	0,0141
	Erro Tipo II	0,3956	0,4088	0,3616	0,3872	0,3500	0,4144	0,4484	0,3912	0,4056	0,3272	0,3890	0,0349
	Erro Geral	0,2742	0,2682	0,2620	0,2662	0,2538	0,2716	0,2716	0,2694	0,2756	0,2556	0,2668	0,0075
σ=0,005	Erro Tipo I	0,2204	0,1996	0,2132	0,2188	0,2228	0,2172	0,1640	0,2108	0,2136	0,2464	0,2127	0,0208
	Erro Tipo II	0,2616	0,2704	0,2244	0,2468	0,2404	0,2580	0,3008	0,2792	0,2560	0,2272	0,2565	0,0234
	Erro Geral	0,2410	0,2350	0,2188	0,2328	0,2316	0,2376	0,2324	0,2450	0,2348	0,2368	0,2346	0,0069
σ=0,01	Erro Tipo I	0,2396	0,2356	0,2180	0,2208	0,2380	0,2276	0,1812	0,2196	0,2356	0,2480	0,2264	0,0187
	Erro Tipo II	0,2364	0,2184	0,2432	0,2140	0,2356	0,2408	0,2744	0,2660	0,2316	0,2160	0,2376	0,0201
	Erro Geral	0,2380	0,2270	0,2306	0,2174	0,2368	0,2342	0,2278	0,2428	0,2336	0,2320	0,2320	0,0070
σ=0,05	Erro Tipo I	0,2500	0,2424	0,2372	0,2760	0,2564	0,2492	0,2072	0,2332	0,2624	0,2772	0,2491	0,0209
	Erro Tipo II	0,2272	0,2184	0,2024	0,2040	0,2180	0,2200	0,2324	0,2444	0,2080	0,1964	0,2171	0,0149
	Erro Geral	0,2386	0,2304	0,2198	0,2400	0,2372	0,2346	0,2198	0,2388	0,2352	0,2368	0,2331	0,0075
σ=0,1	Erro Tipo I	0,2732	0,2528	0,2432	0,2876	0,2664	0,2568	0,2264	0,2436	0,2704	0,2828	0,2603	0,0193
	Erro Tipo II	0,2076	0,2052	0,1904	0,1812	0,2136	0,2144	0,2196	0,2364	0,2024	0,1972	0,2068	0,0156
	Erro Geral	0,2404	0,2290	0,2168	0,2344	0,2400	0,2356	0,2230	0,2400	0,2364	0,2400	0,2336	0,0081
σ=0,2	Erro Tipo I	0,2864	0,2600	0,2608	0,2908	0,2672	0,2644	0,2552	0,2520	0,2708	0,2896	0,2697	0,0143
	Erro Tipo II	0,1984	0,2064	0,1828	0,1816	0,2240	0,2144	0,2072	0,2496	0,2104	0,1964	0,2071	0,0200
	Erro Geral	0,2424	0,2332	0,2218	0,2362	0,2456	0,2394	0,2312	0,2508	0,2406	0,2430	0,2384	0,0082
σ=0,25	Erro Tipo I	0,2896	0,2592	0,2604	0,2940	0,2684	0,2684	0,2596	0,2492	0,2696	0,2912	0,2710	0,0155
	Erro Tipo II	0,1996	0,2044	0,1860	0,1780	0,2224	0,2128	0,2036	0,2476	0,2140	0,1948	0,2063	0,0196
	Erro Geral	0,2446	0,2318	0,2232	0,2360	0,2454	0,2406	0,2316	0,2484	0,2418	0,2430	0,2386	0,0078
σ=0,5	Erro Tipo I	0,2924	0,2704	0,2772	0,3004	0,2712	0,2768	0,2756	0,2744	0,2796	0,3000	0,2818	0,0114
	Erro Tipo II	0,1932	0,1944	0,1848	0,1780	0,2148	0,2188	0,1948	0,2192	0,2120	0,1920	0,2002	0,0148
	Erro Geral	0,2428	0,2324	0,2310	0,2392	0,2430	0,2478	0,2352	0,2468	0,2458	0,2460	0,2410	0,0062
σ=1,0	Erro Tipo I	0,3180	0,2880	0,2996	0,3336	0,2984	0,2948	0,2944	0,3156	0,3064	0,3256	0,3074	0,0151
	Erro Tipo II	0,1856	0,1924	0,1712	0,1720	0,1960	0,2040	0,1936	0,1960	0,2012	0,1780	0,1890	0,0118
	Erro Geral	0,2518	0,2402	0,2354	0,2528	0,2472	0,2494	0,2440	0,2558	0,2538	0,2518	0,2482	0,0065
σ=2,0	Erro Tipo I	0,3744	0,3320	0,3448	0,3836	0,3388	0,3324	0,3536	0,3636	0,3584	0,3668	0,3548	0,0177
	Erro Tipo II	0,1680	0,1892	0,1680	0,1584	0,1816	0,1884	0,1824	0,1932	0,1828	0,1704	0,1782	0,0114
	Erro Geral	0,2712	0,2606	0,2564	0,2710	0,2602	0,2604	0,2680	0,2784	0,2706	0,2686	0,2665	0,0068

Figura D.1: Variação do Parâmetro σ da RBF para SVM.

Encontrado o parâmetro mais adequado para utilização no modelo, escolhemos o algoritmo de otimizção e definimos a máquina SVM através da sintaxe:

kernel = rbf(0.01);
C = 1.0;
tutor = smosvctutor;

O passo seguinte foi realizar o treinamento dos dados. A sintaxe para treinamento das SVMs é apresentada abaixo:

net_{aux} = train(svc, tutor, base de entradas, variável resposta, C, kernel);
net_{aux} = fixduplicates(net_{aux}, base de entradas, variável resposta);
net = strip(net_{aux});

Para identificar quais são os vetores de suporte:

sv = getsv(net);

A aplicação dos resultados de treinamento na amostra de teste, obtendo um vetor com os resultados de $\hat{f}(x)$ e outro com os resultados do classificador $\hat{G}(x) = \text{sign}(\hat{f}(x))$:

$output_{value}$ = fwd(net,base de teste);
$output_{sign}$ = sign(fwd(net,base de teste));

As matrizes de confusão foram obtidas a partir do classificador $\hat{G}(x) = \text{sign}(\hat{f}(x))$. Os erros associados são apresentados na Figura D.2.

Simulacoes	Treino 9 & Teste 3	Treino 5 & Teste 10	Treino 10 & Teste 4	Treino 7 & Teste 10	Treino 7 & Teste 1
Erro Tipo I	0,2396	0,2376	0,2196	0,2176	0,2276
Erro Tipo II	0,2356	0,2360	0,2660	0,2432	0,2404
Erro Geral	0,2376	0,2368	0,2428	0,2304	0,2340
Taxa de Acerto	76,24%	76,32%	75,72%	76,96%	76,60%

Simulacoes	Treino 8 & Teste 4	Treino 4 & Teste 5	Treino 5 & Teste 8	Treino 1 & Teste 3	Treino 1 & Teste 10	Valores Medios	Desvios Padrao
Erro Tipo I	0,2356	0,2212	0,1812	0,2480	0,2356	**0,2264**	0,0187
Erro Tipo II	0,2324	0,2140	0,2744	0,2152	0,2180	**0,2375**	0,0202
Erro Geral	0,2340	0,2176	0,2278	0,2316	0,2268	**0,2319**	0,0069
Taxa de Acerto	76,60%	78,24%	77,22%	76,84%	77,32%	**76,81%**	0,0069

Figura D.2: Indicadores de Desempenho - Máquina de Vetores de Suporte.

Bibliografia

[1] Thomas, L. C., Eldeman, D. B. and Crook, J. N., *Credit Scoring and Its Applicatons*, SIAM, Monograph on Mathematical Modeling and Computation, 2002.

[2] Pereira, S. J., *Gestão e Análise do Risco de Crédito*, 3 ed. São Paulo, Editora Atlas, 2002.

[3] Altman, E. I., *Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy*, Journal of Finance, 1968.

[4] Cox, D. R., *The Analysis of Binary Data*, Methuen, London, 1970.

[5] Hair, J. et al., *Multivariate Data Analysis*, 5 ed. New Jersey, Prentice Hall, 1998.

[6] Gujarati, D. N., *Econometria Básica*, 3 ed. São Paulo, Makron Books, 2000.

[7] SAS Institute Inc., *SAS Enterprise Miner Graphical User Interface*, http://www.sas.com/technologies/analytics/datamining/miner/, 2006.

[8] Paula, G. A., *Modelos de Regressão com Apoio Computacional*, Versão Preliminar, IME/USP, 2004.

[9] Kohavi, R. and Provost, F., *Special Issue on Applications of Machine Learning and the Knowledge Discovery Process*, Kluwer Academic Publishers, Boston, 1998.

[10] Mood, A. M., Graybill F. A. and Boes, D. C., *Introduction to the Theory of Statistics*, McGraw-Hill, 3rd. edition, 1974.

[11] Picinini R., Oliveira G. M. B. & Monteiro, *Mineração de Critério de Crédit Scoring Utilizando Algoritmos Genéticos*, VI Simpósio Brasileiro de Automação Inteligente, 2003.

[12] Friedman, J., Hastie, T. and Tibshirani, R., *The Elements of Statistical Learning*, Friedman, Hastie & Tibshirani, 2001.

[13] McCulloch, W. S. and Pitts W., *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics 5, 1943.

[14] Rosenblatt, F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan, Washington D.C., 1962.

[15] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., *Learning Representation by Back-Propagating Errors*, Nature v323, 1986.

[16] Haykin, S., *Neural Networks - A Comprehensive Foundation*, Prentice Hall, 2nd. edition, 1999.

[17] Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[18] Moller, M., *A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning*, Neural Networks 6, 1993.

[19] Netlab, *Matlab Neural Networks Toolbox*, http://www.ncrg.aston.ac.uk/netlab/, 2006.

[20] Vicente, R., *Redes Neurais para Inferência Estatística*, Notas de Aula Programa de Mestrado em Modelagem Matemática em Finanças, FEA/USP, 2002.

[21] Boser, B. E., Guyon, I. M. and Vapnik, V. N., *A Trainning Algorithm for Optimal Margin Classifiers*, Computational Learning Theory, 1992.

[22] Vapnik, V. N., *The Nature of Statistical Learning Theory*, Springer, 1995.

[23] Fletcher, R., *Practical Methods of Optimization*, Wiley, 2nd. edition, 1987.

[24] Murray, W., Gill, P. and Wright, M., *Practical Optimization*, Academic Press, 1981.

[25] Aizerman, M. A., Braverman, E. M. and Rozonoer, L. I. *Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning*, Automation and Remote Control vol 25, 1964a.

[26] Aizerman, M. A., Braverman, E. M. and Rozonoer, L. I. *The Probability Problem of Pattern Recognition Learning and The Method of Potential Functions*, Automation and Remote Control vol 25, 1964b.

[27] Matlab SVM Toolbox, *Beta Version of Support Vector Machine Toolbox*, http://theoval.sys.uea.ac.uk/svm/toolbox/, 2006.

[28] Semolini, R., *Support Vector Machines, Inferência Transdutiva e o Problema de Classificação*, Dissertação de Mestrado, FECC/UNICAMP, 2002.

[29] Carvalho, B. P. R. e Braga, A. P., *Estratégias Neurais para Treinamento de Least Square Support Vector Machines*, VIII Simpósio Brasileiro de Redes Neurais (SBRN 2004), São Luis - MA, 2004.

[30] Joachims, T., *Making Large-Scale SVM Learning Practical - Advances in Kernel Methods in Support Vector Learning*, C. Burges and A. Smola (ed.), MIT Press, 1999.

[31] Van Gestel, T., Baesens, B., Garcia, J. and Van Dijcke, P., *A Support Vector Machine Approach to Credit Scoring*, 2004.

[32] Johnson, R. A. and Wichern, D. W., *Applied Multivariate Statistical Analysis*, Prentice Hall, 4th. edition, 1998.

[33] Capuano, S. M., *Redes Neurais Aplicadas ao Reconhecimento e Classificação de Padrões em Séries Financeiras*, Dissertação de Mestrado, FEA & IME - USP, 2002.

[34] Nabney, I. T., *Netlab: Algorithms for Pattern recognition*, Springer, 2002.

[35] The MathWorks Inc., *Using Matlab*.