Pontifícia Universidade Católica
do Rio de Janeiro

Vanessa Rodrigues Coelho Leite

Uma análise da classificação de litologias utilizando SVM, MLP e métodos Ensemble

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC–Rio

Orientador: Prof. Dr. Marcelo Gattass

Rio de Janeiro
Julho de 2012

Pontifícia Universidade Católica
do Rio de Janeiro

Vanessa Rodrigues Coelho Leite

Uma análise da classificação de litologias utilizando SVM, MLP e métodos Ensemble

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC–Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Dr. Marcelo Gattass
Orientador
Departamento de Informática — PUC–Rio

Prof. Dr. Aristófanes Corrêa Silva
UFMA

Dr. Pedro Mário Cruz e Silva
TecGraf/PUC–Rio

Prof. Dr. Waldemar Celes
Departamento de Informática — PUC–Rio

Aura Conci
UFF

José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico —
PUC–Rio

Rio de Janeiro, 13 de Julho de 2012

Vanessa Rodrigues Coelho Leite

Graduou-se em Bacharel em Ciência da Computação pela Universidade Federal do Maranhão no ano de 2009.

Ficha Catalográfica

Leite, Vanessa Rodrigues Coelho

Uma análise da classificação de litologias utilizando SVM, MLP e métodos Ensemble / Vanessa Rodrigues Coelho Leite; orientador: Prof. Dr. Marcelo Gattass. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2012.

v., 79 f: il. ; 29,7 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Tese. 2. Litologias. 3. Métodos Ensemble. 4. Redes Neurais. 5. Máquinas Vetores de Suporte. 6. Perceptron Multicamadas. I. Gattass, Marcelo. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 000

Agradecimentos

Ao meu orientador, Marcelo Gattass, que acreditou neste trabalho e me deu esta oportunidade, a Aristófanes Silva e Pedro Mário pelos muitos conselhos e sugestões durante a realização deste trabalho, me dando força e suporte nesta caminhada.

Aos meus pais, Andréa Coelho e Orlando Leite, pela educação que recebi. A minha irmã, Brenda, por me fazer rir em cada momento sofrido. Ao meu namorado, Roberto Gerson, por sempre me dar apoio, e me ajudar na organização das minhas ideias. A todos os meus tios e tias, por compreenderem a minha ausência e me darem apoio.

Aos amigos do V3O2 que me ajudaram a encontrar definições, e por terem me ajudado na conversão dos dados. E à equipe Telemídia, que me fez entender que NTPPM (Nascemos, 'Tamos Prontos Pra Morrer).

Aos amigos e primos, pelo apoio e por não insistirem em me convidar pra sair.

A todos que confiaram que eu podia ir além. Muito obrigada.

Resumo

Leite, Vanessa Rodrigues Coelho; Gattass, Marcelo. **Uma análise da classificação de litologias utilizando SVM, MLP e métodos Ensemble**. Rio de Janeiro, 2012. 79p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A classificação de litologias é uma tarefa importante na caracterização de reservatórios de petróleo. Um de seus principais objetivos é dar suporte ao planejamento e às atividades de perfuração de poços. Dessa forma, quanto mais rápidos e eficazes sejam os algoritmos de classificação, mais confiável será as decisões tomadas pelos geólogos e geofísicos. Esta dissertação analisa os métodos *ensemble* aplicados à classificação automática de litologias. Para isso, foi realizada uma comparação entre classificadores individuais (*Support Vector Machine* e *Multilayer Perceptron*) e estes mesmos classificadores com métodos *Ensemble* (*Bagging* e *Adaboost*). Assim, concluímos com uma avaliação comparativa entre as técnicas, bem como apresentamos o *trade-off* em utilizar métodos *Ensemble* em substituição aos classificadores individuais.

Palavras–chave

Litologias. Métodos Ensemble. Redes Neurais. Máquinas Vetores de Suporte. Perceptron Multicamadas.

Abstract

Leite, Vanessa Rodrigues Coelho; Gattass, Marcelo. **An analysis of lithology classification using SVM, MLP and Ensemble methods**. Rio de Janeiro, 2012. 79p. MsC Thesis — Department of Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Lithology classification is an important task in oil reservoir characterization, one of its major purposes is to support well planning and drilling activities. Therefore, faster and more effective classification algorithms will increase the speed and reliability of decisions made by geologists and geophysicists. This work analises ensemble methods applied to automatic lithology classification. For this, we performed a comparison between single classifiers (Support Vector Machine and Multilayer Perceptron) and these classifiers with ensemble methods (Bagging and Boost). Thus, we conclude with a comparative evaluation of techniques and present the trade-off in using Ensemble methods to replace single classifiers.

Keywords

Lithology. Ensemble Methods. Neural Network. Support Vector Machines. Multilayer Perceptron.

Sumário

Lista de figuras

Lista de tabelas

A ciência não estuda ferramentas. Ela estuda como nós as utilizamos, e o que descobrimos com elas.

Edsger Dijkstra.

1
Introdução

Em nosso planeta, debaixo do solo, vegetação ou água, podemos encontrar materiais sólidos, compostos de um ou mais minerais. Esses materiais agregados recebem o nome de rocha, e constituem parte essencial da crosta terrestre. O estudo da rocha nos permite identificar sua composição, estrutura, propriedades físicas etc. A esse estudo é dado o nome de litologia, e é por meio dele que é possível identificar possíveis reservatórios de petróleo. A Figura 1.1 mostra a relação espacial entre rochas e reservatórios de petróleo. Entretanto, o termo litologia também se refere

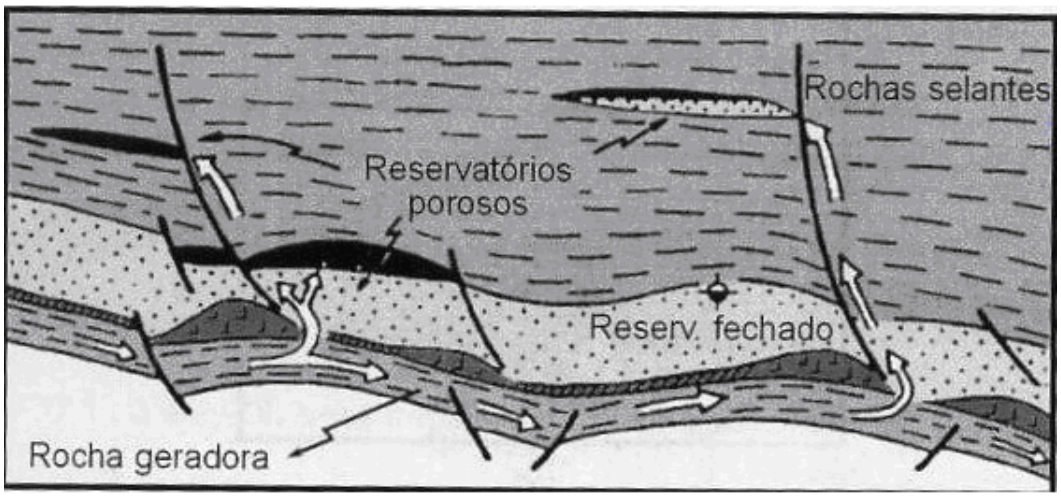

Figura 1.1: Relação espacial entre rochas e reservatórios. Fonte: (Thomas, 2004)

ao tipo de rocha, e é com esse sentido que utilizamos neste trabalho. Atualmente, a detecção da litologia é feita na área de um poço perfurado. Sendo comumente realizada de duas formas (Thomas, 2004):

- Análise de Perfis Elétricos: perfis de um poço representam a variação dos valores de uma ou mais propriedades elétricas da rocha ao longo de seu eixo (comprimento do poço). Com base nos gráficos destes perfis, um geólogo faz uma análise manual para identificar a litologia.
- Testemunhagem: processo de coleta de amostras do material da rocha ao longo do eixo do poço. Estas amostras, geralmente coletadas em cilindros chamados testemunhos, permitem uma avaliação direta da rocha. O erro

mais relevante neste caso é a posição do testemunho no eixo, que pode ser afetada pela compressão das camadas acima. Serve para calibrar processos de inferência. Seu principal problema é ser muito caro para ser executado em todos os poços.

A obtenção dos dados de perfis é feita de uma forma mais simples e mais barata, entretanto, a análise feita acaba dependendo da experiência do geólogo. No Capítulo Conceitos Básicos serão detalhadas as informações sobre perfis elétricos e litologias.

1.1 Motivação

Conforme discutido anteriormente, a identificação das litologias por meio de perfis é feita de forma manual, o que torna essa atividade subjetiva, dependente da experiência do geólogo, sujeita a erros, e principalmente, cansativa. A Figura 1.2 mostra um exemplo de perfis reais. Mais ainda, a quantidade de informações envolvidas na análise de vários perfis elétricos em conjunto torna inviável ao geólogo realizar a classificação, uma vez que não dispõe de ferramentas para auxiliá-lo. Dessa forma, encontrar classificadores capazes de ter bons resultados na identificação de litologias, é importante para tornar este processo mais rápido, menos cansativo e menos propenso a erros.

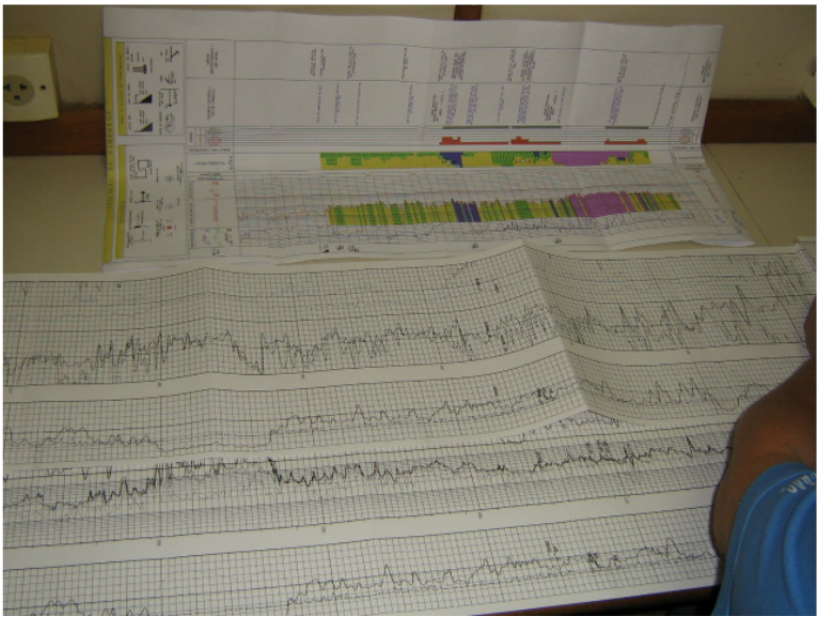

Figura 1.2: Perfis reais para análise. Fonte: (Antunes, 2010)

1.2 Objetivos

O objetivo deste trabalho é avaliar a utilização de métodos de aprendizado de máquina como *Support Vector Machines* (SVM) e *Multilayer Perceptron* (MLP), individualmente e em conjunto com métodos *ensemble*.

1.3 Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma: o Capítulo 2 apresenta os conceitos básicos para auxiliar o entendimento deste trabalho, o Capítulo 3 apresenta os trabalhos relacionados. No Capítulo 4 são apresentados os experimentos e resultados, bem como uma discussão geral, e por fim, no Capítulo 5, apresenta-se a conclusão sobre o trabalho apresentado e propostas de trabalho futuro.

2
Conceitos Básicos

Os conceitos utilizados neste trabalho são apresentados neste capítulo e estão divididos entre a caracterização de reservatórios e aprendizado de máquina.

2.1
Caracterização de reservatórios

A caracterização de reservatórios é uma tarefa muito complexa devido a estrutura heterogênea dos reservatórios, na qual mudanças muito grande de propriedades são observadas em uma mesma área. Essas mudanças são ocasionadas pela natureza da rocha, idades geológicas e deposição. A caracterização de reservatório é muito importante para a industria do petróleo, por permitir trilhar o melhor caminho para a perfuração dos poços. Como dito anteriormente, essa caracterização pode ser dada de uma forma direta (contato direto visual ou tátil com as amostras do reservatório) como a testemunhagem, ou de uma forma indireta como por exemplo, através de perfis elétricos de poços.

2.1.1
Perfilagem de Poços

As rochas são diferenciadas e caracterizadas a partir de suas propriedades físicas como, por exemplo, resistividade, densidade, radioatividade, acústica, etc. Essas características são medidas ao longo de um poço através da perfilagem. A perfilagem é um processo que consiste no deslocamento contínuo de um sensor de perfilagem (sonda) dentro do poço, capaz de medir características e propriedades físicas da rocha.

É através da perfilagem que são obtidos os parâmetros necessários para a identificação dos tipos de rochas e dos fluidos existentes nas mesmas. Dessa forma, é possível identificar intervalos portadores de hidrocarbonetos, bem como quantificar os volumes descobertos (Antunes, 2010).

Cada característica extraída representa um perfil elétrico[1] do poço. Dessa forma, o perfil elétrico é equivalente a uma função que relaciona a profundidade

[1] Os dados obtidos são chamados de maneira geral de perfis elétricos, independente do processo físico pelo qual são obtidos (Duarte, 2003).

medida com um valor real, resultante da medição de uma determinada característica. A Figura 2.1 mostra um exemplo de perfil elétrico e sua litologia correspondente. Esses perfis são armazenados em arquivos no formato LAS (*Log ASCII Standard*),

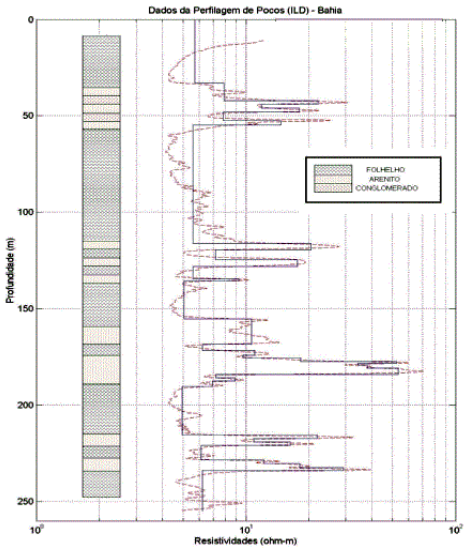

Figura 2.1: Perfil elétrico de resistividade e litologia correspondente. Fonte: (Carrasco *et al*, 2003)

que possui um cabeçalho com informações relacionadas ao poço e aos perfis medidos, e por colunas numéricas, sendo que a primeira representa a profundidade do poço, e as demais, os valores das medições. A Figura 2.2 mostra um exemplo de arquivo LAS.

```
~Version Information Block
 VERS.                 2.00:    CWLS LOG ASCII STANDARD - VERSION 2.000000
 WRAP.                   NO:    One Line Per Depth Step
#
~Well Information Block
#MNEM.UNIT                      Data                          Information
#----------  ------------------------------------------  ---------------
 STRT.FT                                        100.0000:
 STOP.FT                                       5300.0000:
 STEP.FT                                          0.5000:
 NULL.                                         -999.2500:
#
~Curve Information Block
#MNEM.UNIT       API CODE      Curve Description
#----------    ------------   ------------------
 DEPT.FT                   :   Depth in Feet
 GR   .GAPI                :   Gamma Ray
 RHOB.G/C3                 :   Bulk Density
#
~A  DEPTH         GR        RHOB
   100.000     59.449      2.705
   100.500     59.449      2.705
   101.000     61.024      2.703
   101.500     61.024      2.697
   102.000     61.811      2.670
   102.500     61.811      2.668
   103.000     61.811      2.668
   103.500     61.922      2.671
   104.000     59.356      2.664
   104.500     57.615      2.664
   105.000     57.021      2.664
   105.500     57.056      2.663
   106.000     57.807      2.662
   106.500     59.438      2.642
   107.000     60.948      2.640
   107.500     61.006      2.640
   108.000     61.036      2.644
   108.500     61.001      2.649
   109.000     61.176      2.648
   109.500     61.956      2.652
   110.000     60.861      2.639
   110.500     59.249      2.630
   111.000     59.501      2.627
   111.500     59.741      2.625
   112.000     63.713      2.624
```

Figura 2.2: Exemplo de arquivo LAS.

2.1.2
Tipos de Perfis Elétricos

Para fazer uma melhor análise da rocha e identificar um possível intervalo de hidrocarbonetos, vários tipos de perfis são utilizados. Os perfis mais comuns são apresentados na Tabela 2.1 juntamente com o modo de obtenção e suas principais utilizações.

2.1.3
Litologia

Segundo (Duarte, 2003), litologia é a descrição sistemática de uma rocha em termos de composição mineralógica, granulação, textura e cor. As rochas existentes na Terra são divididas em três grupos:

– ígneas: rochas formadas pela solidificação e consolidação do magma;
– sedimentares: é o tipo mais abundante de rocha do planeta, são formadas através do acúmulo de detritos, sedimentos, matéria orgânica etc;
– metamórficas: são formadas pela deformação de outras rochas devido à variação de temperatura e pressão, por exemplo.

As rochas sedimentares são as mais interessantes para a indústria de petróleo, uma vez que, devido ao seu processo de formação, possuem concentração de hidrocarbonetos, podendo conter bacias petrolíferas. A Figura 2.3 apresenta uma relação com as principais litologias encontradas nas bacias brasileiras, e sua legenda correspondente.

Figura 2.3: Representação litológica. Fonte: (Milani et al, 2007)

Tabela 2.1: Tipos de Perfis, medições e utilizações mais comuns. Fonte: (Thomas, 2004)

Nome do Perfil	Medição	Utilização
Potencial Espontâneo (SP)	Diferença de potencial entre dois eletrodos, um na superfície e um dentro do poço	Detectar camadas permoporosas, calcular a argilosidade das rochas e auxiliar na correlação de informações com poços vizinhos
Raios Gama (GR)	Radioatividade total da formação geológica	Identificação de litologias, minerais radioativos e cálculo do volume de argilas ou argilosidade
Neutrônico (NPHI)	Em perfis antigos representa os raios gama de captura após excitação artificial, em perfis mais novos representa a quantidade de nêutrons epitermais/termais da rocha até o bombardeio	Estimativa de porosidade, litologia e detecção de hidrocarbonetos leves ou gás
Sônico (DT)	Diferença nos tempos de trânsito de uma onda mecânica através das rochas	Estimativa de porosidade, correlação poço a poço, estimativa do grau de compactação das rochas, constantes elásticas, detecção de fraturas e apoio à sísmica
Densidade (RHOB)	Raios gama refletidos pelos elétrons orbitais dos elementos componentes das rochas	Densidade das camadas, porosidade e identificação de zonas de gás
Delta-Rô (DRHO)	Presença de lama ou buracos que causem desvios na curva de densidade	Monitora a quantidade de correção que está sendo adicionada à curva RHOB pelo circuito de compensação.
Caliper	Diâmetro do poço	Cálculo do volume de cimento para tampões ou cimentação do poço, controle de qualidade de perfis e indicações das condições do poço

2.2 Aprendizado de Máquina

Aprendizado de máquina é um subcampo da inteligencia artificial, que trata de técnicas que permitem ao computador aprender, ou seja, melhorar seu desempenho em dada tarefa a partir da experiência. Em geral essas tarefas envolvem reconhecimento, diagnóstico, planejamento, controle de robôs, previsões, etc. O aprendizado é feito através do reconhecimento de padrões automaticamente a partir de exemplos e observações, sendo assim, similar ao aprendizado humano, ou seja, capaz de criar generalizações. O aprendizado de máquina é dividido em três principais abordagens (Nilsson, 1998, Haykin, 1998):

- supervisionado: é fornecido um conjunto de entrada e saída da qual será identificado um padrão/comportamento;
- não supervisionado: é fornecido apenas o conjunto de entrada, procurando identificar os agrupamentos naturais existentes no conjunto;
- por reforço: há uma interação com o ambiente, recebendo bônus em caso de acerto, ou ônus em caso de erro.

Neste trabalho a abordagem utilizada é o aprendizado supervisionado. No aprendizado supervisionado existe a figura de um professor externo, o qual apresenta o conhecimento ao ambiente por tuplas de exemplos na forma: (entrada, saída desejada). Dessa forma é necessário um conjunto de dados de treinamento que foram classificados previamente para que o sistema possa aprender a classificar estes mesmos dados e também novas entradas não conhecidas. Para que seja possível a classificação de novos dados, assumimos que haja uma regra geral, ou seja, uma função que faça o mapeamento da entrada para a saída. Assim, tendo um grupo de amostras da forma (x_i, y_i), onde x_i representa uma característica e y_i é a classe, um classificador deve ser capaz de predizer a classe de novas amostras. Esse processo de indução através do qual um classificador aprende é denominado treinamento. Durante a fase de treinamento procuramos encontrar uma hipótese h, que melhor aproxime os valores da entrada com a saída, dessa forma, h será uma boa sugestão da função f que procuramos aprender - principalmente se a base do treinamento for suficientemente larga. As classes representam o objeto de interesse sobre o qual se deseja fazer previsões. Neste trabalho, considera-se o caso em que os rótulos assumem valores discretos $(1, 2, 3, ..., n)$, assim tem-se um problema de classificação. Caso as classes possuíssem valores contínuos, teria-se um problema de regressão. A Figura 2.4 demonstra o processo de indução do classificador de forma simplificada.

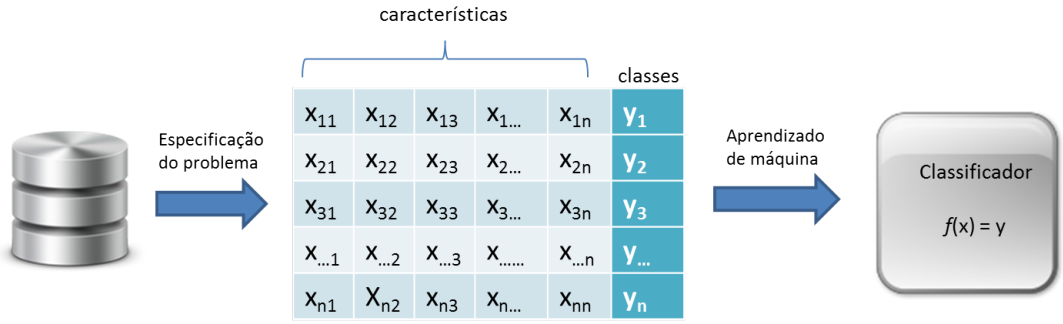

Figura 2.4: Modelo da indução de um classificador na técnica de aprendizado supervisionado: seja uma base de dados com x amostras, na qual cada amostra é formada por um conjunto de m características e está associada a uma classe, ou seja, $x_i = \{m_1, m_2, ..., m_m\} \rightarrow y$.

2.2.1 *Support Vector Machine* - SVM

O SVM é uma técnica de aprendizado de máquina que se baseia em aprendizado estatístico que foi proposta por Vapnik em 1995 (Cortes and Vapnik, 1995). Desde então vem recebendo constante atenção, principalmente pelo fato de que a aplicação dessa técnica tem provido resultados melhores que a aplicação de redes neurais. Consiste resumidamente em encontrar um hiperplano ótimo capaz de separar conjuntos, isto é, que possua a maior margem de diferenciação entre eles possível. SVM originalmente foi desenvolvido como um método de separação linear (Cortes and Vapnik, 1995), mas é possível extendê-lo de forma a separar classes que são não lineares, a ideia principal é transportar os dados para um espaço de dimensão maior no qual eles possam ser separados linearmente.

Quando tentamos reconhecer padrões (realizar classificação), estamos tentando estimar uma função $f : \mathbb{R}^n \rightarrow \{\pm 1\}$ usando dados de treinamento, ou seja, padrões x_i de n dimensões (características) e uma classe y_i, conforme mostra a Equação 2-1

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^n \times \pm 1 \tag{2-1}$$

de tal modo que f irá classificar corretamente novas amostras (x, y) (ou seja, $f(x) = y$ para cada par (x, y) que tenha sido gerado com a mesma distribuição de probabilidade $P(x, y)$ dos dados de treinamento. Se não colocarmos nenhuma restrição sobre a classe das funções a partir da qual escolhemos nossa estimativa f, mesmo que ela resolva muito bem para os dados de treinamento, pode ser que ela não generalize bem para exemplos desconhecidos. SVMs são uma implementação aproximada do método de minimização do risco estrutural, que vem da teoria do aprendizado estatístico (Vapnik, 1995). Este princípio de indução é baseado no fato

de que a taxa de erro do aprendizado de máquina nos dados de teste (isto é, a taxa de erro da generalização) é limitada pela soma da taxa de erro no treinamento com um termo que depende da dimensão Vapnik-Chervonenkis (VC) (Haykin, 1998).

Hiperplanos de separação

Para simplificar, consideremos classes que sejam linearmente separáveis, como por exemplo, padrões em que as classes y_i só possam assumir os valores $+1$ e -1. A equação de uma superfície de decisão, na forma de um hiperplano que separará as duas classes é dada por:

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0, \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R} \tag{2-2}$$

onde $\mathbf{x}$ é um elemento de entrada (característica), $\mathbf{w}$ é um vetor de peso ajustável e b é um limiar. Dessa forma, é possível escrever:

$$\begin{cases} (\mathbf{w} \cdot \mathbf{x}) + b \geq 0, \text{ para } y_i = 1 \\ (\mathbf{w} \cdot \mathbf{x}) + b < 0, \text{ para } y_i = -1 \end{cases} \tag{2-3}$$

Para um dado vetor $\mathbf{w}$ e um limiar b a separação entre o hiperplano definido pela Equação 2-2 e o ponto mais próximo dos dados é chamada de margem de separação

Dizemos que um hiperplano é ótimo quando ele possui a margem máxima de separação entre duas classes, conforme vemos na Figura 2.5.

Podemos observar que a margem de separação, medida perpendicularmente ao hiperplano, é igual a $2/\|\mathbf{w}\|$. E que, portanto, a menor distância entre o hiperplano separador e os dados de treinamento é igual a $1/\|\mathbf{w}\|$. Dessa forma, maximizar a margem de separação significa minimizar (Scholkopf and Smola, 2001):

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{2-4}$$

$$\text{Sujeito a: } y_i \cdot ((\mathbf{w} \cdot x_i) + b) \geq 1 \tag{2-5}$$

Essas restrições determinam que não haverá dados de treinamento entre as margens de separação das classes, definindo assim um **SVM com margens rígidas**. Esse hiperplano ótimo pode ser construído através da resolução de problema de otimização quadrática com restrições cuja solução $\mathbf{w}$ tem uma expansão $\mathbf{w} = \sum_i v_i \mathbf{xi}$ em termos de um subconjunto de padrões de treinamento que se encontram na margem. Estes padrões de treinamento da margem são chamados vetores de suporte (SV) e trazem todas as informações importantes para o problema de classificação. A minimização da Equação 2-4 pode ser resolvida através da introdução de uma equação Lagrangiana, gerando o seguinte problema de otimização (Scholkopf and Smola, 2001):

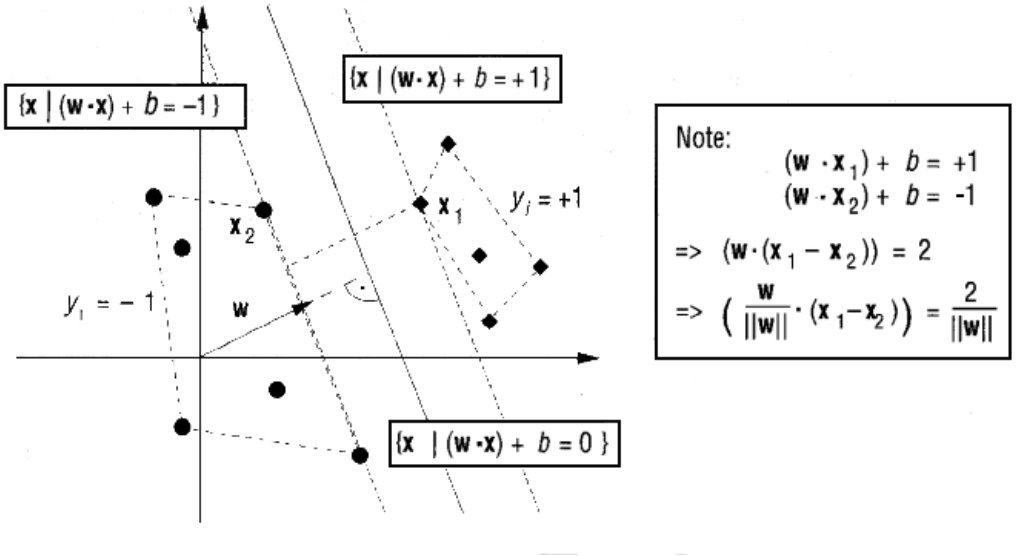

Figura 2.5: Exemplo de problema de classificação: separar as bolas dos losangos. O hiperplano ótimo é ortogonal à menor linha que conecta o *convex hull* das duas classes e a divide ao meio. Existe um vetor de peso $\mathbf{w}$ e um limiar b de tal modo que $y_i \cdot ((\mathbf{w} \cdot x_i) + b) > 0$. Redimensionando $\mathbf{w}$ e b de tal forma que os pontos mais próximos ao hiperplano satisfaçam $|(\mathbf{w} \cdot \mathbf{w}) + b| = 1$, nós obtemos uma forma $(\mathbf{w}, b)$ do hiperplano com $y_i \cdot ((\mathbf{w} \cdot x_i) + b) \geq 1$. Fonte:(Hearst *et al*, 1998)

$$\underset{\alpha}{\text{Maximizar}} \qquad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x_i} \cdot \mathbf{x_j}) \tag{2-6}$$

$$\text{Sujeito a:} \begin{cases} \alpha_i \geq 0, \forall i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{2-7}$$

Note que este problema é formulado utilizando-se apenas os dados de treinamento e suas classes. A solução deste problema gera um classificador $f(x)$, apresentado na Equação 2-8 que representa o hiperplano que separa os dados com a maior margem possível, onde sgn representa a função sinal e α_i é um multiplicador de Lagrange (Hearst *et al*, 1998).

$$f(x) = sgn\left(\sum_{x_i \in SV} y_i \alpha_i x_i \cdot x + b\right) \tag{2-8}$$

A característica mais importante desse algoritmo é que, tanto o problema de otimização quadrática, Equação 2-6, quanto a função de decisão final, Equação 2-8, dependem somente do produto escalar entre os padrões, e é isto que permite a generalização para casos não lineares.

SVM com Margens Suaves Algumas vezes os dados podem conter ruídos e *outliers*, o que faz com que a restrição da Equação 2-5 não possa ser totalmente

satisfeita. Essa violação da restrição pode acontecer de três formas, conforme mostra a Figura 2.6:

– A amostra (x_i, y_i) está dentro da margem de separação, mas do lado correto do hiperplano

– A amostra (x_i, y_i) está dentro da margem de separação, mas do lado errado do hiperplano

– A amostra (x_i, y_i) está fora da margem de separação e do lado errado do hiperplano

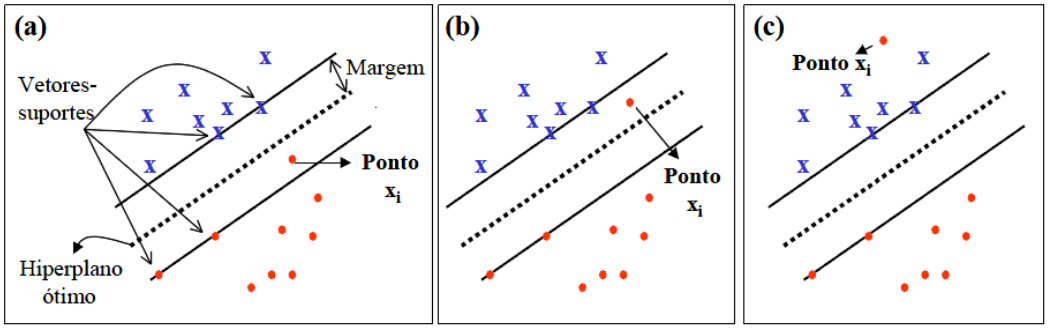

Figura 2.6: Formas de violação da restrição da Equação 2-5 de um SVM com Margens Rígidas. A violação dessa restrição gera um SVM de Margens Suaves que minimize a probabilidade do erro de classificação. Fonte:(Semolini, 2002)

Dessa forma, faz-se necessário acrescentar variáveis de folga, ξ_i, para que, mesmo que não seja possível encontrar um hiperplano que separe todas as amostras se tenha um hiperplano que minimize a probabilidade do erro de classificação junto às amostras de treinamento.

$$y_i \cdot ((\mathbf{w} \cdot x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1, \cdots, n \tag{2-9}$$

Nota-se então que essa suavização permite alguns erros de classificação. Esse erro é indicado por um valor de $\xi_i > 1$. Para considerar a variável de folga, o problema de minimização original passa a ser

$$\underset{w,b,\xi}{\text{Minimizar}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{n} \xi_i\right) \tag{2-10}$$

A constante C que aparece na Equação 2-10 representa um termo de regularização, que impõe um peso aos erros no conjunto de treinamento. Quanto maior essa constante, maior é o "custo" de um erro. Assim como o problema de minimização anterior, este também é um problema de otimização quadrático com restrições lineares, e também pode ser resolvido através da introdução de uma equação Lagrangiana, gerando um novo problema de otimização, como mostra a Equação 2-11:

$$\underset{\alpha}{\text{Maximizar}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x_i} \cdot \mathbf{x_j}) \tag{2-11}$$

$$\text{Sujeito a:} \begin{cases} 0 \leq \alpha_i \leq C, \forall i = 1, \cdots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{2-12}$$

Espaço de características e *kernels*

Encontrar um hiperplano ótimo para dados que são linearmente separáveis é uma tarefa simples, entretanto, há muitos casos em que não é possível dividir os dados de forma satisfatória por um hiperplano. Na Figura 2.7 é observado um exemplo no qual a separação entre os conjuntos só é satisfatoriamente realizada através do uso de uma fronteira curva.

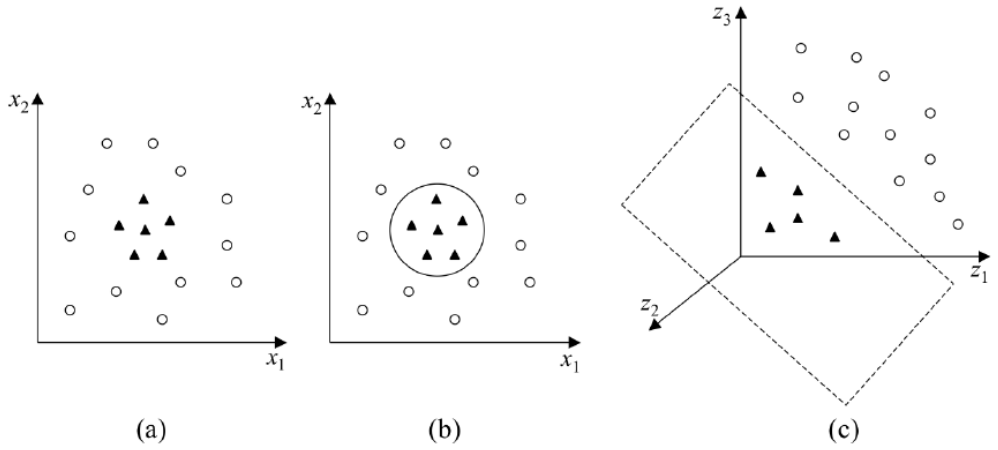

Figura 2.7: (a) Conjunto de dados não linear; (b) Fronteira não linear no espaço de entradas; (c) Fronteira linear no espaço de características. Fonte:(Lorena and Carvalho, 2007)

Para lidar com problemas não lineares a ideia básica do SVM é mapear o conjunto de treinamento do espaço original, para um outro espaço F, de dimensão maior, chamado espaço de características, através de um mapeamento não linear: $\phi : \mathbb{R}^n \rightarrow F$ e utilizar o algoritmo de separação linear em F. Esse mapeamento não linear associado a um espaço de características com dimensão suficientemente alta, assegura, pelo teorema de Cover (Haykin, 1998) que no espaço F os dados serão linearmente separáveis. Dessa forma, para aplicar SVM sobre problemas não linearmente separáveis, faz-se necessário mapear os dados para um espaço de dimensão maior utilizando ϕ (no qual eles são linearmente separáveis) e aplicar SVM nesse espaço. Entretanto como F pode ter uma dimensão muito alta, computar ϕ pode ser extremamente custoso ou até mesmo inviável, mas como a única

informação necessária do mapeamento é o produto escalar entre os dados no espaço de características, isso pode ser resolvido com a utilização de funções *kernels*.

Um *Kernel* K é uma função que recebe dois pontos x_i e x_j do espaço de entrada e computa o produto escalar desses dados no espaço de características (Herbrich, 2001). Assim:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{2-13}$$

É comum utilizar funções *kernels* sem conhecer o mapeamento ϕ, que é gerado implicitamente. Dessa forma, a vantagem da utilização dos *kernels* é a simplicidade de seu cálculo e a capacidade de representar espaços abstratos. Para garantir a convergência do problema de otimização, utiliza-se funções *kernel* que obedeçam o teorema de Mercer (Herbrich, 2001): resumidamente, que gere matrizes positivas semi-definidas M, na qual cada elemento $M_{i,j}$ seja definido por $K(x_i, x_j)$ para todo $i, j = 1, \cdots, n$.

Alguns dos *Kernels* mais utilizados na prática são os Lineares, Polinomiais, os Gaussianos ou RBF (*Radial-Basis Function*) e os Sigmoidais. Na Tabela 2.2 constam suas funções e seus parâmetros (que devem ser informados pelo usuário).

Tabela 2.2: Principais funções *kernels*

Nome do *kernel*	Função	Parametros
Linear	$x_i \cdot x_j$	-
Polinomial	$(\gamma(x_i \cdot x_j) + k)^d$	γ, k e d
Gaussiano	$exp(-\gamma\|x_i - x_j\|^2)$	γ
Sigmoidal	$\tanh(\gamma(x_i \cdot x_j) + k)$	γ e k

Neste trabalho utilizamos o SVM com a função *kernel* RBF.

***Kernel* RBF** O *kernel* RBF é dependente da distância euclidiana entre um vetor suporte e uma amostra do conjunto de dados. O vetor suporte será o centro do RBF e irá determinar a área de influência do vetor de suporte sobre a área de dados. O parâmetro γ desempenha um papel importante no *kernel* RBF, se for alto ele é capaz de reproduzir fronteiras de decisão irregulares, e se for baixo gera fronteiras suaves, evitando a reprodução de ruído das amostras, o que previne um *overfit* para os dados. Com alto valor de γ é possível conseguir melhores resultados com as amostras de treinamento, isso porque a função se adapta aos dados, o que possivelmente, para dados de testes não gerará bons resultados.

SVM multiclasses O algoritmo do SVM foi desenvolvido para problemas binários de classificação, entretanto, existem duas abordagens comumente

utilizadas para realizar a separação multiclasses: um contra um e, um contra todos. Seja um problema de classificação no qual existam N classes. Na abordagem um contra todos N classificadores serão construídos, um para cada classe, e uma nova amostra será considerada da classe correspondente ao classificador que gerar as melhores taxas de acerto. Já na abordagem um contra um são construídos $N(N-1)/2$ classificadores, uma nova amostra será classificada com o rótulo da classe que obtiver mais votos.

Neste trabalho foi utilizada a abordagem um contra um, implementada pela LibSVM (Chang *et al*, 2011). Essa abordagem possui uma implementação mais simples, e em geral, produz resultados mais precisos que a abordagem um contra todos, principalmente quando as amostras entre as classes possuem quantidades muito diferentes (Anthony *et. al.*, 2007, Milgram *et. al.*, 2006, Hsu and Lin, 2002)

2.2.2 *Multilayer Perceptron*

Rede Neural Artificial (RNA) é um paradigma de programação que procura simular a microestrutura do cérebro e são amplamente utilizadas em problemas de inteligência artificial como tarefas de reconhecimento de padrões, sinais, manipulação robótica, etc (Rojas, 1996). *Multilayer perceptron* (MLP) é uma RNA compostas por nós organizados em camadas: entrada, uma ou mais camadas escondidas e uma camada de saída (Haykin, 1998). Com exceção da camada de entrada, todas as outras camadas são constituídas por nós com capacidade computacional (neurônios). A Figura 2.8 mostra um modelo de rede MLP com duas camadas escondidas.

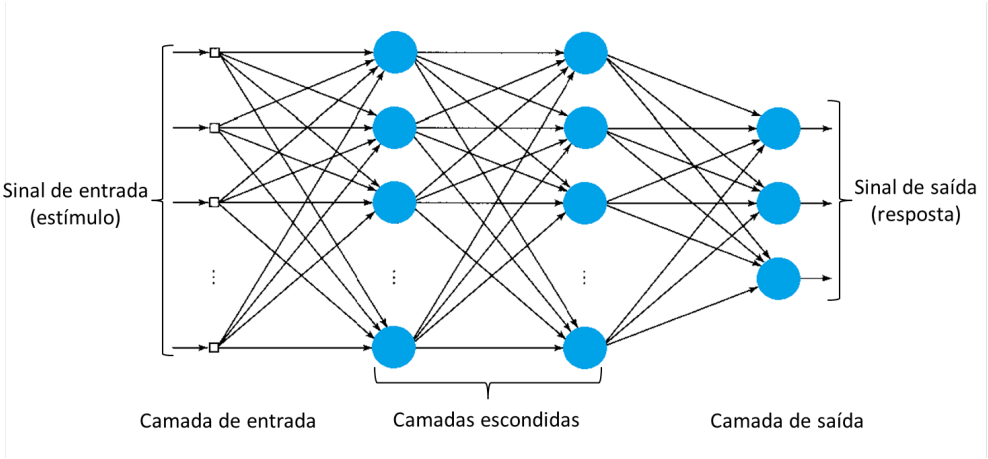

Figura 2.8: Arquitetura de uma rede neural MLP com duas camadas escondidas

MLP é uma rede progressiva, ou seja, cada nó de uma camada é conectado unicamente aos nós da camada seguinte, sem laços de realimentação. Dessa forma, o sinal da camada de entrada se propaga pela rede camada a camada,

progressivamente. Além disso, é completamente conectada, ou seja, cada nó de uma camada se conecta a todos os nós da camada seguinte. A conexão entre os nós são conhecidas como sinapses. A camada de entrada é a que trata a alimentação das características (recebe um sinal de entrada, ou seja, uma função de ativação para a geração dos sinais de entrada), e o número de nós dessa camada é determinado pela dimensionalidade do espaço de observação, as camadas escondidas são as que efetuam o processamento necessário para a classificação e a camada de saída representa as classes disponíveis para resultado mas também tem capacidade computacional, sendo o número de nós igual ao número de classes. A camada de saída gera um sinal de saída, que no caso da classificação de padrões é a resposta, e na fase de treinamento o sinal gerado representa uma função de erro, que será retropropagado camada a camada, de acordo com o algoritmo de treinamento supervisionado. O número de camadas escondidas é determinado de acordo com a complexidade do modelo. Redes com apenas uma camada escondida só representam problemas linearmente separáveis e são conhecidas como *single layer*. Cada sinapse possui um peso próprio que é determinado de acordo com a utilização de algoritmos de treino supervisionados. O principal algoritmo utilizado é o algoritmo de retropropagação do erro (*Backpropagation*).

A resolução de problemas não-lineares ocorre pela utilização de mais de uma camada escondida e também pela inserção de funções de ativação não-lineares de cada neurônio. A função de ativação, também chamada de função de transferência, é uma função matemática que, aplicada à combinação linear entre as variáveis de entrada e os pesos das sinapses, retorna ao seu valor de saída, ou seja, determina a relação entre entradas e saídas de cada neurônio da rede. Os nós que possuem capacidade computacional são responsáveis por dois tipos de operações:

- Sinal funcional na saída de cada neurônio (função de ativação) e cálculo dos pesos sinápticos
- Estimativa do vetor gradiente, ou seja, gradiente da superfície de erro relacionado aos pesos conectados.

Função de ativação

Cada neurônio na rede possui uma função de ativação não-linear, entretanto, essa não-linearidade é suave, de tal forma que a função seja diferenciável em qualquer ponto. Uma forma comum, e que é utilizada neste trabalho, é a não-linearidade sigmoidal (Rojas, 1996), definida por:

$$\varphi_j(v_j(n)) = \frac{1}{1 + exp(-av_j(n))}, a > 0 \text{ e } -\infty < v_j(n) < \infty \qquad (2\text{-}14)$$

onde v_j é o potencial de ativação do neurônio j, ou seja, a soma ponderada de todas as entradas sinápticas. De acordo com esta não-linearidade, a amplitude de saída fica restrita ao intervalo $0 \leq y_j \leq 1$. Derivando a função de ativação com respeito a $v_j(n)$, tem-se:

$$\begin{aligned} \varphi\prime(v) = \frac{d}{dv}\varphi(v) &= \frac{d}{dv}\{\frac{1}{1+exp(-av)}\} \\ &= \frac{aexp(-av)}{[1+exp(-av)]^2} \\ &= a\varphi^2(v)(\frac{1}{\varphi(v)} - 1) \\ &= a\varphi(v)(1-\varphi(v)) \end{aligned} \tag{2-15}$$

e como $y_j(n) = \varphi(v_j(n))$,

$$\varphi\prime(v_j(n)) = \frac{d}{dv}\phi(v_j(n)) = ay_j(n)[1-y_j(n)] \tag{2-16}$$

Nesta equação, pode-se perceber que a derivada atinge valor máximo quando $y_j(n) = 0.5$ e o valor mínimo quando $y_j(n) = 0$ ou $y_j(n) = 1$.

Algoritmo *Backpropagation*

O algoritmo *Backpropagation* é baseado em uma heurística de aprendizado por correção de erro, ou seja, o erro é retropropagado da camada de saída até a camada de entrada[2]. Esse algoritmo é composto por dois passos (Rojas, 1996): um passo direto e um passo reverso, como mostra a Figura 2.9.

De modo geral, no passo direto um padrão é apresentado à camada de entrada da rede. A atividade resultante segue através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No passo reverso, a saída obtida é comparada à saída desejada. Se estas não forem iguais, o erro é calculado e é propagado a partir da camada de saída até a camada de entrada. E nesse processo os pesos das sinapses das unidades da camada de saída e das camadas intermediárias são recalculados à media que o erro é retropropagado, de forma que a resposta obtida pelo MLP se aproxime da resposta desejada.

Considere o sinal de erro definido na Equação 2-17, onde n é o n-ésimo vetor de treinamento apresentado à rede e j é o neurônio na camada de saída que está gerando o sinal. Define-se o valor instantâneo do erro quadrático para o neurônio j como $\frac{1}{2}e_j^2(n)$. Dessa forma, a soma instantânea dos erros quadráticos para todos os neurônios na camada de saída (que é a única camada visível, permitindo o cálculo

[2]Esse caminho reverso - da saída para a entrada - só ocorre na fase de treinamento, que é quando se usa o algoritmo de retropropagação do erro, não retirando assim a característica de uma rede progressiva

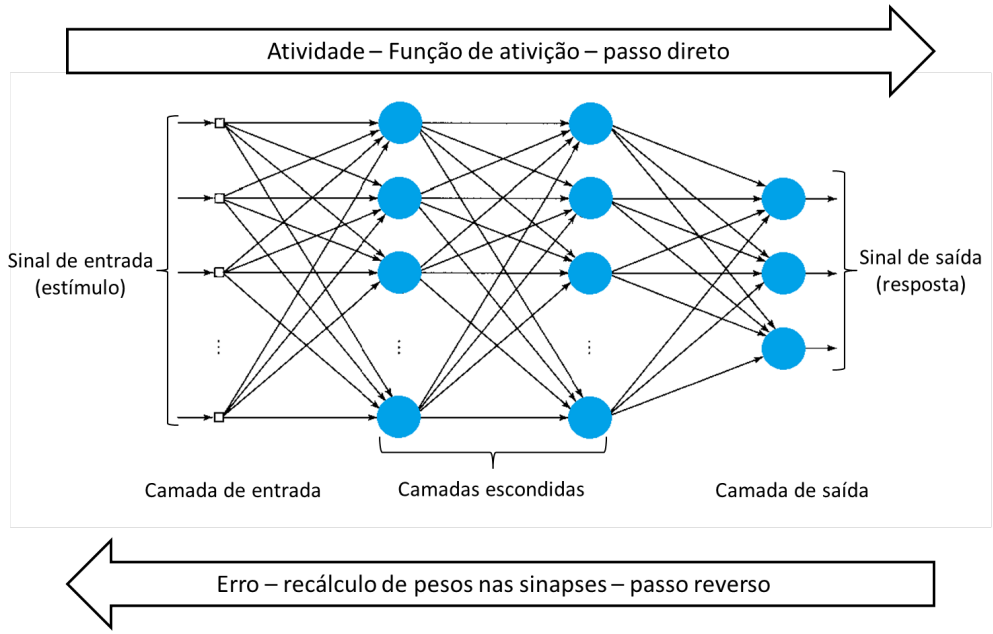

Figura 2.9: Sentido dos dois passos do algoritmo *Backpropagation*

direto da soma dos erros) é definido por $\mathcal{E}$, como mostra a Equação 2-18, onde C representa todos os nós da camada de saida.

$$e_j(n) = d_j(n) - y_j(n) \tag{2-17}$$

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2-18}$$

Seja N o número total de vetores de treinamento (padrões), o erro médio quadrático (MSE) é obtido somando $\mathcal{E}(n)$ sobre todo n e então normalizando com relação ao tamanho do conjunto. O MSE é denotado por $\mathcal{E}_{av}$ e é dado por:

$$\mathcal{E}_{av} = \frac{1}{N-1}\sum_{n=0}^{N-1} \mathcal{E}(n) \tag{2-19}$$

$\mathcal{E}_{av}$ representa a Função de Custo do processo de minimização do erro de aprendizado. Para conseguir minimizar $\mathcal{E}_{av}$ os pesos sinápticos são atualizados a cada entrada de um novo vetor de treinamento até o fim de uma época. Uma época é o intervalo correspondente à apresentação de todos os N vetores de treinamento à camada de entrada do MLP. A correção aplicada aos pesos sinápticos tem como base a direção contrária do gradiente local da superfície de erro $\mathcal{E}(w)$ relativo ao peso sináptico w. Dessa forma, se em uma variação do peso sináptico há um movimento ascendente na superfície $\mathcal{E}(w)$, então significa que esta variação deve ser aplicada com o sinal invertido, uma vez que houve um aumento do erro. Caso a variação gere um movimento descendente, então significa que esta variação deve ser aplicada com o sinal positivo já que houve uma diminuição do erro. Este método de correção é conhecido como Regra Delta (Rumelhart *et. al.*, 1986) e é definida pela

expressão $\Delta w(n) = -\eta \nabla J(w(n))$, onde $\nabla J(w(n)) = \dfrac{\partial J(w(n))}{\partial w(n)} = \dfrac{\partial \{\frac{1}{2} e^2(n)\}}{\partial w(n)}$ é o gradiente local da superfície de erro gerada pela função de custo $J = J(w(n)) = \dfrac{1}{2} e^2(n)$ a ser minimizada no instante n. A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$, ou seja, o peso sináptico que conecta a saída do neurônio i à entrada do neurônio j na iteração n, ditada pela Regra Delta é definida na Equação 2-20:

$$\Delta w_{ji}(n) = w_{ji}(n+1) - w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \tag{2-20}$$

onde η é a constante que determina a razão de aprendizado do algoritmo *backpropagation*. No algoritmo *backpropagation*, o aprendizado de um MLP estabelece, através da Regra Delta, que a correção dos pesos das sinapses é dada por:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{2-21}$$

onde $\Delta w_{ji}(n)$ é a correção aplicada à i-ésima sinapse do neurônio j, $y_i(n)$ é o sinal de entrada no i-ésimo nó de entrada do neurônio j e $\delta_j(n)$ é o gradiente local do neurônio j, que é definido por:

$$\delta_j(n) = \begin{cases} \varphi\prime_j(v_j(n)) e_j(n), & \text{neurônio } j \text{ é de saída} \\ \varphi\prime_j(v_j(n)) \displaystyle\sum_k \delta_k(n) w_{kj}(n), & \text{neurônio } j \text{ é escondido} \end{cases} \tag{2-22}$$

O gradiente local $\delta_j(n)$ para um neurônio de saída j é igual ao produto do correspondente sinal de erro $e_j(n)$ pela derivada $\varphi\prime_j(v_j(n))$ da função de ativação associada. Neste caso, o principal fator envolvido para o ajuste de pesos é o sinal de erro. Entretanto, quando o neurônio j está em uma camada escondida ele não é diretamente acessado, mas ainda assim é responsável pelo erro resultante na camada de saída. A dificuldade, entretanto, é saber como penalizar ou recompensar os pesos sinápticos, já que não há um sinal de erro específico para eles. A solução encontrada é computar o sinal de erro recursivamente para o neurônio escondido j retro-propagando os sinais de erros de todos os neurônios à direita do neurônio j. Dessa forma, para neurônios na camada escondida, o fator $\varphi\prime_j(v_j(n))$ depende somente da função de ativação. Os fatores envolvidos no somatório sobre k depende de $\delta_k(n)$ que representa os sinais de erros $e_k(n)$ recursivamente retro-propagados e de $w_{kj}(n)$ que representa os pesos sinápticos dos neurônios à direita de j - os mesmos dos quais os sinais de erros serão retro-propagados.

Os dois passos do algoritmo *backpropagation* estão descritos abaixo:

Passo direto Nessa etapa, os pesos sinápticos permanecem inalterados e os sinais são propagados da entrada para a saída, neurônio por neurônio. O sinal que resulta na camada de saída do neurônio j é dado por:

$$y_j(n) = \varphi(v_j(n)) \tag{2-23}$$

onde $v_j(n)$ é o potencial de ativação do neurônio j, que é definido por:

$$v_j(n) = \sum_{i=0}^{m} wji(n)y_i(n) \tag{2-24}$$

sendo m o total de entradas aplicadas ao neurônio j, $w_{ji}(n)$ é o peso sináptico que liga a saída do neurônio i à entrada do neurônio j e $y_i(n)$ é o sinal de saída do neurônio i (que é equivalente ao sinal de entrada do neurônio j). Se o neurônio j está na primeira camada escondida, então o índice i refere-se ao i-ésimo nó da camada de entrada, e dessa forma $y_i(n) = x_i(n)$, onde $x_i(n)$ é o i-ésimo componente do vetor de entrada do neurônio j. Mas, se o neurônio j estiver na camada de saída, o índice j refere-se ao j-ésimo nó da camada de saída, e dessa forma $y_j(n) = o_j(n)$, onde $o_j(n)$ é o j-ésimo componente do vetor de saída. Esta saída é comparada com a resposta desejada $d_j(n)$ sendo obtido o sinal de erro $e_j(n)$ para o j-ésimo neurônio da camada de saída.

Passo reverso O passo reverso começa na camada de saída e propaga os sinais de erro na direção contrária da rede, chegando até a camada de entrada. A cada camada passada vai, recursivamente, computando os gradientes locais para cada neurônio, e assim, recalculando os pesos das sinapses. Para um neurônio na camada de saída, o gradiente local é simplesmente o sinal de erro daquele neurônio multiplicado pela primeira derivada de sua não-linearidade, como mostra a Equação 2-22. A partir desse gradiente, aplica-se a Equação 2-21 para recalcular os pesos das sinapses que alimentam a camada de saída. Assim, obtido os gradientes locais para os neurônios da camada de saída, aplica-se novamente a Equação 2-22, só que dessa vez, o neurônio encontra-se em uma camada escondida, calculando então o gradiente local de todos os neurônios na camada à esquerda. Com esses gradientes, aplica-se novamente a Equação 2-21 para recalcular os pesos das sinapses que alimentam esta camada. E, até chegar na camada de entrada, esse procedimento é realizado recursivamente, fazendo as correções necessárias em todas as sinapses. É possível perceber que a mudança em um peso sináptico é proporcional a derivada da função de ativação, assim, para a função escolhida (Equação 2-14), os pesos sinápticos sofrem a maior alteração quando os sinais assumem valores no meio de seu intervalo de variação. Essa característica contribui para a estabilidade do algoritmo de aprendizagem.

Razão de Aprendizado e Fator de Momento

O algoritmo *backpropagation* possibilita uma aproximação da trajetória de movimento sobre a superfície de erro no espaço dos pesos sinápticos. Essa aproximação em cada ponto da superfície, segue a direção da descida mais ingrime. Quanto menor for a razão de aprendizado, η, menores serão as correções aplicadas aos pesos entre uma iteração e outra, e mais suave será a trajetória no espaço de pesos. Entretanto, esse resultado é obtido com uma lenta convergência do algoritmo, até que se alcance um valor de erro pequeno o suficiente pra ser aceitável. Mas, se aumentarmos o fator de aprendizado, a convergência do algoritmo será acelerada, mas as correções aplicadas aos pesos sinápticos podem ser muito grandes, deixando o algoritmo instável. De forma a manter o algoritmo estável, e acelerar a convergência do algoritmo, acrescenta-se à Regra Delta um fator de momento, dessa forma a Equação 2-21 é modificada para:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{2-25}$$

onde a constante α é a constante de Momento com valores entre 0 e 1. Seu efeito é aumentar a velocidade da trajetória no espaço de pesos na direção da descida mais ingrime. Utilizando esta nova equação, se a correção aplicada em determinado peso sináptico mantiver o mesmo sinal algébrico em várias iterações consecutivas, a correção é acelerada pelo fator de momento, já que o mínimo global deve estar longe ainda. Entretanto, se a correção aplicada troca o sinal algébrico em várias iterações consecutivas, a correção do peso é freada pela redução do valor absoluto médio do fator de momento acrescentado já que, um mínimo global deve estar próximo, e uma alta velocidade poderia desestabilizar o algoritmo.

2.2.3 Métodos *Ensemble*

Ensemble é uma palavra francesa que significa junto, comitê. A ideia geral é representar um conjunto de soluções combinadas para um único problema. Assim, *ensemble* é um conjunto de classificadores que tomam decisões individuais, mas que são combinados para classificar novas amostras. Para que classificadores *ensembles* tenham mais acurácia que os classificadores individuais, uma condição necessária e suficiente é que os classificadores individuais tenham acurácia e sejam diversos (Hansen and Salamon, 1990). Um classificador com acurácia é aquele que tem uma taxa de erro melhor que a utilização de uma adivinhação randômica para novos valores de entrada. E dois classificadores são diversos quando eles cometem erros diferentes para uma nova entrada (Dietterich, 2000).

A Figura 2.10 representa o modelo de um Método *Ensemble*. Basicamente, vários classificadores individuais são instanciados e recebem um conjunto de

características como entrada. Cada classificador realiza suas predições para os dados recebidos e gera uma regra de aprendizado. A partir de então, um método de combinação desses resultados é aplicado e um novo classificador é formado.

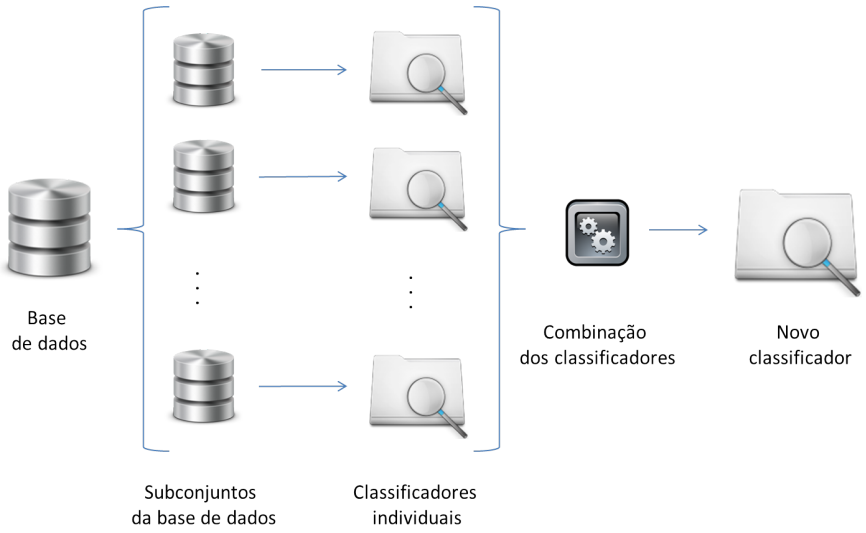

Figura 2.10: Modelo conceitual do aprendizado *ensemble*: vários conjuntos de dados são criados, para cada conjunto um classificador é treinado e então eles são combinados gerando um novo classificador.

Ao combinar soluções de classificadores independentes temos uma melhoria no resultado, uma vez que a combinação de estimativas diminui a variância e, o erro médio esperado de um conjunto de modelos é melhor que o erro médio esperado de cada um dos modelos. Métodos *ensemble* são indicados para serem utilizados em classificadores fracos[3], isto é, classificadores que realizam predições ligeiramente melhores que adivinhação randômica.

Construindo um classificador *ensemble*

Existem muitos métodos para construir classificadores *ensembles*, as mais comuns são: enumeração de hipóteses, manipulação dos exemplos de treinamento, manipulação das características de entrada, manipulação dos objetivos finais e injeção de aleatoriedade (Dietterich, 2000). Em nosso trabalho, utilizamos a abordagem de manipulação dos exemplos de treinamento por ser uma das mais comuns (Optiz and Maclin, 1999, Zhou *et. al.*, 2002) e também por ser útil para tratar grandes volumes de dados, particionando os dados em sub-conjuntos e treinando classificadores com diferentes partições e então combinar as saídas.

Manipulação dos exemplos de treinamento Nesta abordagem os exemplos de treinamento são manipulados para gerar múltiplas hipóteses. O algoritmo de

[3]O termo classificador fraco se refere a uma medida de desempenho, em geral precisão, que atinge taxas em torno de 50%

aprendizagem é executado várias vezes, cada vez com um subconjunto diferente dos exemplos de treinamento. Esta técnica funciona especialmente bem para algoritmos instáveis, ou seja, cuja classificação pode sofrer grandes alterações em razão de pequenas alterações nos dados de treinamento.

Os classificadores *ensemble* predizem as classes por meio de uma combinação dos votos dos classificadores individuais. Várias abordagens são conhecidas para combinar os classificadores: média, média ponderada, votação simples, integrais *fuzzy*, etc (Rokach, 2005). Neste trabalho é utilizada votação simples, por ser o método mais comum de combinação para classificação (Zhou, 2009).

Combinando classificadores com votação simples Neste tipo de abordagem, cada classificador individual faz uma predição e a opção mais votada é escolhida como resposta. Esse tipo de combinação apesar de ser bem simples produz bons resultados. Seja um *ensemble* com 5 classificadores, para que uma amostra seja classificada incorretamente é necessário que três dos cinco classificadores façam uma escolha errada. Se a probabilidade de erro de cada classificador individual for de 0.2, e se os classificadores cometem erros independentes, a probabilidade de erro do *ensemble* será de 0.008. É fato que na prática os erros não serão totalmente independentes, uma vez que os classificadores serão gerados com a mesma base de treinamento. Mas, ainda assim, como são usadas para cada classificador um subconjunto da base de treinamento, os classificadores serão pelo menos um pouco diferentes, reduzindo a correlação dos erros.

Os métodos usados neste trabalho são exemplos de algoritmos com manipulação dos exemplos de treinamento: *Adaboost* (Freund and Schapire, 1996) e *Bagging* (Breiman, 1996).

Adaboost

Adaboost é um algoritmo pertencente a uma categoria chamada *Boost*, que representa uma família de métodos, cujo objetivo principal é produzir um conjunto de classificadores. A definição do conjunto de treinamento aplicado a um classificador é dependente dos resultados dos classificadores anteriores, dessa forma, amostras que são incorretamente classificadas por um dado classificador do conjunto será mais importante para os classificadores futuros. O método *Adaboost* é um exemplo de *ensemble* no qual os subconjuntos da base de dados são iguais à própria base de dados. A diferença entre as bases são os pesos adicionados para cada amostra. Além disso, *Adaboost* é baseado no fato

de que o desempenho de classificadores simples é melhorado quando eles são combinados iterativamente, ou seja, os classificadores são construídos em sequência e cada classificador subsequente irá prestar mais atenção nas amostras que foram classificadas erroneamente pelos classificadores prévios. A Figura 2.11 representa o modelo de funcionamento dos métodos de *adaboost*.

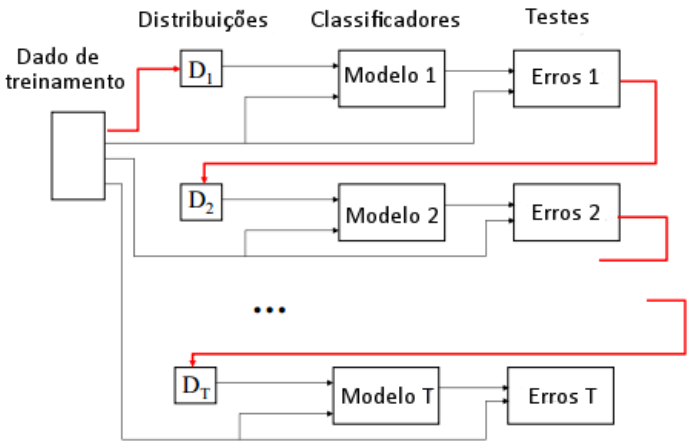

Figura 2.11: Modelo de funcionamento dos métodos *adaboost*. Fonte: (Hauskrecht, 2004)

Dessa forma, dado a base de dados, é considerado que cada amostra possui o mesmo peso inicial. O classificador então é construído para aquela base e ao final, para cada amostra classificada erroneamente, é realizado um recalculado de seu peso. Após a atualização do peso das amostras classificadas erroneamente o restante das amostras é atualizado de forma que a soma de todos os pesos seja igual a 1. Novamente o classificador é construído e o processo se repete até um número de vezes previamente definido. O Algoritmo 1 mostra o pseudo-código usado no *Adaboost*.

Como podemos perceber, cada amostra classificada erroneamente recebe uma atenção maior dos classificadores seguintes, dessa forma, se a base de dados contiver ruídos é possível que o classificador final gerado não tenha boa predição para novas amostras devido a um super ajuste aos dados (*overfit*). Além disso, uma vez que a mesma base de dados é utilizada por diversas vezes (apenas alterando a importância de algumas amostras), a base de dados utilizada pelo *adaboost* não tem a necessidade de ser larga (Sewell, 2008).

Bagging - Bootstrap Aggregating

O método *Bagging* se baseia no fato de que grande parte dos erros de classificação são devidos a uma escolha muito específica da base de treinamento, dessa forma, a ideia básica é gerar vários conjuntos de treinamento a partir do conjunto original que serão utilizados por classificadores individuais

Algoritmo 1 *Adaboost*

Entrada:

- Sequência de m exemplos $\{(x_i, y_i), \cdots, (x_m, y_m)\}$ com classes $y_i \in Y = \{1, \cdots, k\}$
- Algoritmo de aprendizagem fraca: **Classificador**
- Inteiro T indicando o número de iterações do algoritmo

Inicializar: $D_1(i) = 1/m$ para todo i

para $t = 1, 2, \cdots, T$

1. Construa o **Classificador** passando a distribuição D_t
2. Pegue a hipótese gerada pelo classificador: $h_t : X \to Y$
3. Calcule o erro da hipótese: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
4. Defina $\beta_t = \dfrac{\epsilon_t}{(1 - \epsilon_t)}$
5. Atualize a distribuição D_t:

se $h_t(x_i) = y_i$ **então**

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \beta_t$$

senão

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t}$$

fim se

onde Z_t é uma constante de normalização, para que o somatório de todos os pesos seja sempre igual a 1.

fim para

Saída: A hipótese final: $h_{final}(x) = \underset{y \in Y}{\operatorname{argmax}} \sum_{t:h_t(x_i = y)} \log \frac{1}{\beta}$

concorrentemente. Cada novo conjunto pode conter uma mesma amostra repetida, como pode não conter algumas amostras do conjunto original. Assim, para cada conjunto criado será gerada uma nova função capaz de classificar aquele conjunto e um método de combinação será aplicado para gerar um novo classificador. Geralmente o número de amostras do subconjunto gerado é o mesmo do conjunto original, mas é uma amostragem com reposição, ou seja, para um conjunto de treinamento com M amostras, para gerar os subconjunto é sorteado um número de forma aleatória entre 1 e M, M vezes, e o elemento do conjunto original na posição sorteada é adicionado ao subconjunto que está sendo gerado.

O Algoritmo 2 mostra o pseudo-código do método *bagging*. Assim, dada a base de dados BD de tamanho M, vários conjuntos de treinamento BD_t são construídos. A partir daí é criado um classificador C_t para cada BD_t e a hipótese h_t é adicionada ao *ensemble*. Então para classificar uma nova amostra, cada classificador C_t retorna a sua predição que conta como um voto. O Classificador final C_f conta todos os votos e determina a resposta como a classe com a maioria dos votos.

Algoritmo 2 *Bagging*

Entrada:

- Sequência de m exemplos $\{(x_i, y_i), \cdots, (x_m, y_m)\}$ com classes $y_i \in Y = \{1, \cdots, k\}$
- Algoritmo de aprendizagem: **Classificador**
- Inteiro T indicando o número de iterações do algoritmo

para $t = 1, 2, \cdots, T$

1. Construa o subconjunto BD_t da base de dados através de uma amostragem com reposição

2. Pegue a hipótese gerada pelo classificador para a base gerada: $h_t : X \to Y$ e adicione-a ao *ensemble*

fim para

Saída: A hipótese final: $h_{final}(x) = \frac{1}{M} \sum_t f_t(x)$

O algoritmo de *bagging* tem se mostrado bastante efetivo para algoritmos de aprendizado instáveis (Breiman, 1996), ou seja, aqueles em que pequenas diferenças no treinamento geram grandes diferenças nas classificações.

2.2.4 Medidas de Avaliação

Validação Cruzada

Existem várias abordagens de validação cruzada (Refaeilzadeh *et al*, 2009) como: método *holdout*, método k-*fold* e método *leave-one-out*. Neste trabalho foi utilizado o método k-*fold*, com n igual a 10. O valor escolhido para o número de *folds* baseia-se no fato de este ser o valor considerado, em consenso na comunidade de mineração de dados, como um bom valor para o número de partições (Refaeilzadeh *et al*, 2009).

Método k-*fold* Neste método o conjunto de dados é particionado em k partes iguais (ou o mais aproximadamente possível). Em seguida, k iterações de treinamento e validação são realizadas, sendo que a cada iteração uma partição é deixada de fora do treinamento, sendo usada apenas na validação. A Figura 2.12 apresenta como funciona a validação cruzada com 3 partições.

Matriz de confusão

A matriz de confusão oferece uma medida efetiva de avaliação mostrando o número de classificações corretas *versus* o número de classificações preditas para cada classe. Para um conjunto de n classes, a matriz de confusão seria construída como mostra a Tabela 2.3. Cada célula da matriz indica o número de

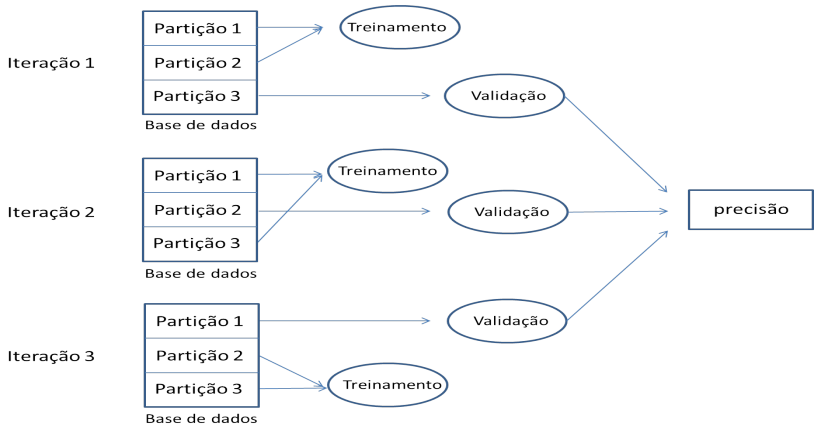

Figura 2.12: Simulação de validação cruzada com 3 partições

amostras que pertencem atualmente à classe C_i, mas que deveria estar na classe C_j. A Equação 2-26 mostra o cálculo para contagem de elementos pertencentes a cada célula da matriz de confusão, onde T representa a base de dados, e o operador $\|$ expressão $\|$ retorna 1, se a expressão for verdadeira e zero caso contrário (Baranauskas and Monard, 2000).

$$M(C_i, C_j) = \sum_{\forall (x,y) \in T : y = C_I} \|h(x) = C_j\| \tag{2-26}$$

Tabela 2.3: Modelo matriz de confusão para n classes

Nome da classe	Predição C_1	Predição C_2	$\cdots$	Predição C_n
Classe C_1	$M(C_1, C_1)$	$M(C_1, C_2)$	$\cdots$	$M(C_1, C_n)$
Classe C_2	$M(C_2, C_1)$	$M(C_2, C_2)$	$\cdots$	$M(C_2, C_n)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Classe C_n	$M(C_n, C_1)$	$M(C_n, C_2)$	$\cdots$	$M(C_n, C_n)$

Dessa forma, as predições corretas em cada classe se apresentam na diagonal principal da matriz. Os demais valores representam amostras classificadas erroneamente. Logo, um classificador ideal seria aquele que apresenta todos os valores, fora da diagonal principal, iguais a zero. Dado um processo de classificação com regras, quatro situações podem ocorrer:

- Uma instância satisfaz todas as regras e é classificada corretamente
- Uma instância satisfaz todas as regras mas é classificada incorretamente
- Uma instância não satisfaz as regras mas é classificada corretamente
- Uma instância não satisfaz as regras e é classificada incorretamente

Dessa matriz algumas medidas de avaliação dos classificadores são retiradas, como taxa de erro, precisão etc. Para simplificar, consideremos um problema

de classificação binário (positivos *versus* negativos). Com um problema de classificação binário as duas possibilidades de classificar incorretamente são chamadas de falso positivo (FP) e falso negativo (FN). A Tabela 2.4 representa a matriz de confusão para um problema binário, onde TP representa as classes verdadeiramente positivas, TN as classes verdadeiramente negativas. A soma das quatro possibilidades representa o total das instâncias.

Tabela 2.4: Modelo matriz de confusão para um problema binário

Nome da classe	Predição Positivos	Predição Negativos
Positivos	TP	FP
Negativos	FN	TN

As medidas utilizadas neste trabalho são: *recall*, *precision* e *f-measure*.

Recall

Recall é também conhecida como sensitividade ou taxa de verdadeiros positivos e representa a parte de amostras positivas entre todas as que foram consideradas positivas. Uma alta taxa de *recall* significa que o classificador acertou a maioria dos resultados da classe. A Equação 2-27 mostra como é calculada a medida de *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2-27}$$

Precision

A medida *precision* é também conhecida como valor predito positivo e representa a parte de amostras que realmente são positivas entre todas as amostras que deveriam ser classificadas como positivas. Uma alta taxa dessa medida significa que entre todos os valores que o classificador considerou como positivo, a maioria realmente era positivo. A Equação 2-28 mostra como é calculada a medida *precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2-28}$$

F-measure

F-measure é também conhecida como F_1-*score* e representa uma medida de acurácia. Essa medida leva em consideração *precision* e o *recall* fazendo uma média harmônica entre esses valores, e é dada pela Equação 2-29:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision + recall} \tag{2-29}$$

O parâmetro β representa qual a prioridade entre *precision* e *recall*. Assim, quanto maior for o valor de β, maior será o peso dado a taxa de *recall*. Aqui, utilizaremos o valor de $\beta = 1$, portanto, nossa medida equivale a:

$$F_\beta = 2 \cdot \frac{precision \cdot \textit{recall}}{precision + \textit{recall}} \tag{2-30}$$

3
Trabalhos Relacionados

Várias abordagens para identificar litologias automaticamente têm sido discutidas em muitos trabalhos na literatura.

Em (Flexa *et al*, 2004), utilizou-se análise de discriminante (uma técnica da estatística multivariada), tentando encontrar a associação entre as variações das propriedades físicas (perfis elétricos) e os tipos litológicos. Cinco perfis elétricos foram utilizados como entrada de características: Raios Gama (GR), Sônico (DT), Porosidade Neutrônica (PHIN), Densidade (RHOB) e Resistividade (ILD). A análise discriminante adotada tratou apenas com distribuições bivariadas, separando selantes (folhelhos) de reservatórios (arenitos/carbonatos) e em uma segunda análise separou os arenitos dos carbonatos. Para realizar os experimentos foram utilizadas informações de três poços do Campo de Namorado (Bacia de Campos, Brasil) e dois poço do Lago do Maracaibo (Venezuela). Sendo que, dos três poços do Campo de Namorado, um foi utilizado para recolhimento da análise estatística e dois para testes, enquanto no Lago do Maracaibo um poço foi utilizado para recolhimento da análise e o outro para teste. Os testes apresentaram resultados coerentes com a interpretação dos perfis, gerada por testemunhos. Não é informada a taxa de acerto desta análise, nem quaisquer outras medidas, de modo que não é possível realizar um estudo comparativo.

Em (Al-Anazi *et al*, 2010), foram utilizados *Linear Discriminant Analysis* (LDA), SVM e *Probabilistic Neural Network* (PNN) de modo a fazer um estudo comparativo entre os classificadores na classificação de litologias; oito perfis elétricos foram encontrados, comuns a todo os poços utilizados: Caliper, Raios Gama, Foto Elétrico, Neutrônico, Sônico, Densidade, Superficialidade e Microresistividade. Somados a esses perfis também foi adicionada a informação de profundidade. Um pré-processamento para filtragem dos dados, através de extração de características baseadas em um algoritmo de *ranking* de lógica *fuzzy* foi realizado, para reduzir a quantidade de perfis utilizados como entrada, assim, reduziu-se a entrada para quatro perfis: Raios Gama, Caliper, Neutronico e Foto Elétrico. Os três classificadores obtiveram taxas semelhantes, com o SVM obtendo melhores resultados (as menores taxas de classificação errada). Três tipos litológicos foram reconhecidos (*clean sand*, *shaly sand* e *sandy shale*). Os testes

foram realizados com informações de dois poços localizados em um reservatório de arenito heterogêneo do Oriente Médio. O primeiro poço consistia de 164 amostras de *clean sand*, 89 de *shaly sand* e 152 de *sandy shale*, já o segundo poço possuia 122, 232 e 106 amostras, respectivamente. Os melhores resultados são alcançados pelo SVM que têm taxas de classificação errada em torno de 5% e 10%.

Em (Hsieh *et al*, 2005) é feita a identificação de litologias em aquíferos utilizando análise baseada em lógica *fuzzy*. Para construir o sistema, quatro perfis elétricos são utilizados: Raios Gama, Sônico, Potencial Espontâneo e Fase de Indução (comumente utilizados em aquíferos), a identificação se baseia em um sistema de inferência composto de regras que determinam a probabilidade de um conjunto de entradas (perfis elétricos) pertencer a uma dada litologia. Para realizar essa inferência o sistema se compõe de quatro fases: fuzificação, regras *fuzzy* ("se-então"), sistema de inferência *fuzzy*, e por fim, desfuzificação. Durante a fuzificação os perfis são convertidos para cinco termos linguísticos (muito baixo, baixo, médio, alto e muito alto), a partir de então é montada uma base de dados com regras do tipo "se-então" para identificação da litologia. Nesse trabalho foram construídas 12 regras, que obedecem a seguinte estrutura: Se Perfil1 é A, Perfil2 é B, ..., PerfilN é C, então Litologia é D. Onde A, B, e C representam os termos linguísticos criados na fuzzificação, e D a resposta do sistema. Essas regras são construídas baseadas no conhecimento do geólogo para identificar as litologias. Com a base de dados montada, passa-se para a segunda fase (utilização do sistema de inferência) a fim de converter as regras em uma resposta litológica, para tanto é utilizado o sistema Madani (MATLAB, 2001), logo após aplica-se a desfuzzificação, revelando assim a litologia correspondente à área. Os testes foram realizados com as informações de dois poços de um aquífero na área Shui-Lin, em Taiwan contendo cinco tipos litológicos (*silt*, *clay*, *fine sand*, *medium sand* e *coarse sand*. Os dois poços possuiam 50 conjuntos de dados (formados pelos tipos litológicos e a litologia correspondente) que foram divididos em conjunto de treinamento (80%) e testes (20%). Os resultados se mostraram bastante promissores chegando a uma taxa de acerto de 90%.

Em (Cunha, 2002) é utilizada uma rede neural MLP para fazer a identificação de litologias. Para isso, informações dos perfis elétricos são associadas com dados da testemunhagem e dados de inclinação das trajetórias dos poços, a partir das quais são retiradas regras lógicas através de mineração de dados. Nesse trabalho foram utilizados dados do Campo de Namorado, da Bacia de Campos, foram nove poços selecionados, gerando um total de 28 litologias, dos quais cinco foram utilizados para treinamento e quatro para testes. Os perfis utilizados e comuns aos nove poços são Raios Gama, Potencial Espontâneo, Sônico, Neutrônico e Densidade. Várias fases de tratamento de padrões problemáticos são realizadas a fim de ter uma boa

taxa de acerto nos testes. Dessa forma das 28 litologias encontradas o trabalho identifica sete: arenito grosso amalgamado (com 133 amostras), arenito médio laminado (com 25 amostras), arenito médio maciço gradado (com 491 amostras), arenito médio cimentado (com 75 amostras), siltito argiloso estratificado (com 143 amostras), interlaminado siltito argiloso e marga (com 198 amostras) e marga bioturbada (com 345 amostras). Os resultados mostraram uma taxa de acerto na identificação de litologia de aproximadamente 80%.

Já em (Santos *et al*, 2002) e (Santos *et al*, 2003) a classificação de litologias é feita utilizando os métodos ensembles *Driven Pattern Replication* (DPR) e ARC-X4, com uma rede neural MLP. São utilizados no experimento os dados de uma bacia brasileira, com 3330 amostras dividas em oito tipos litológicos: *sand*, *clay*, *sandstone*, *shale+limestone*, *sand+limestone*, *shale*, *marl* e *siltstone*. Para a classificação foram utilizadas as informações de quatro perfis elétricos (*gamma ray*, *sonic*, *density*, *resistivity*) e mais a informação de profundidade. No primeiro trabalho são analisados dados não estratificados, ou seja, dados em que algumas litologias contém muito mais amostras que outras, que representa a distribuição real das amostras, e os resultados obtidos corroboram a melhoria provida por métodos ensembles, chegando a obter um crescimento na taxa média de acerto de 31%, chegando a aproximadamente 85% quando utilizando o método ensemble DPR, em comparação à utilização do MLP individualmente. Já no segundo trabalho, são utilizados dados estratificados que continuam corroborando a melhoria gerada por métodos ensemble, apesar do crescimento ter sido menor que o anterior (7%), e a técnica ensemble que melhor proveu resultados ter sido ARC-X4, alcançando aproximadamente 89%.

A Tabela 3.1 apresenta um resumo dos trabalhos relacionados.

Tabela 3.1: Resumo dos trabalhos relacionados

Trabalho	Técnica Utilizada	Quantidade de poços / amostras	Perfis de Entrada	Litologias Identificadas	Resultados apresentados
(Flexa *et al*, 2004)	Análise Discriminante	3 poços, sem definição de amostras	Raios Gama, Sônico, Porosidade Neutrônica, Densidade, Resistividade e Profundidade	Selantes (folhelho) e Reservatórios (arenito / carbonato)	———
(Al-Anazi *et al*, 2010)	LDA SVM PNN	2 poços, 900 amostras	Caliper, Raios Gama, Foto elétrico e Neutrônico	clean sand shaly sand	87% 90% 69%
(Hsieh *et al*, 2005)	Análise Lógica Fuzzy	2 poços, 50 amostras	Raios Gama, Potencial Espontâneo, Sônico e Fase de Indução	silt clay fine sand medium sand coarse sand	90%
(Cunha, 2002)	MLP	9 poços, 1410 amostras	Raios Gama, Potencial Espontâneo, Sônico, Neutrônico e Densidade	4 tipos de arenito 2 tipos de siltito 2 tipos de marga	85%
(Santos *et al*, 2002)	Ensemble MLP	3330 amostras	Raios Gama, Sônico, Densidade e Resistividade	sand clay sandstone shale+limestone sand+limestone shale marl siltstone	80%
(Santos *et al*, 2003)	Ensemble MLP	3330 amostras	Raios Gama, Sônico, Densidade, Resistividade e Caliper	Reservatório Selante Fonte	88%

4
Experimentos e Resultados

4.1
Aquisição de dados

Para a realização deste trabalho foram utilizadas as informações dos poços da base pública do Mar do Norte[1]. Os poços utilizados pertencem ao bloco F, que possui 108 poços, todos com informações de latitude e longitude. Os poços foram mapeados em suas localizações, conforme mostra a Figura 4.1, a fim de fazer uma seleção espacialmente bem distribuída, uma vez que poços em uma mesma área tendem a ter características semelhantes, o que não geraria uma base diversificada. Os poços selecionados encontram-se em vermelho.

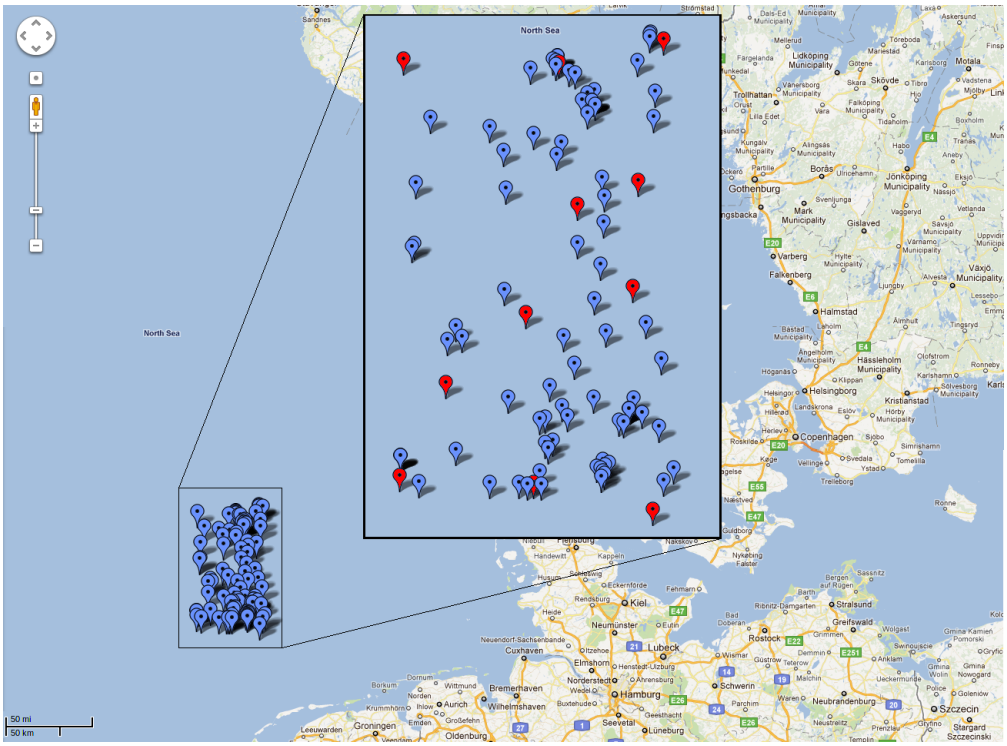

Figura 4.1: Mapeamento dos poços do conjunto F do Mar do Norte no Google Maps

Dos 108 poços, 11 foram selecionados para aquisição de amostras dos perfis elétricos e das litologias. Os 11 poços selecionados constam na Tabela 4.1. As

[1]Esta base está disponível em: http://www.nlog.nl/nlog/listAllWellLocations

informações necessárias (perfis elétricos e litologias) constavam separadamente, os perfis em um arquivo texto e as litologias em um arquivo de imagem. A conversão da informação de litologia para o arquivo texto foi realizada de forma manual. A Figura 4.2 demonstra um trecho de um arquivo do qual foi extraída a informação de litologia.

Tabela 4.1: Nome e variação de profundidade dos poços selecionados para os experimentos

Poço	Intervalo de profundidade (m)
F01-01	1594 - 1971.4
F03-03	2535 - 3909.8
F03-07	1722 - 3097.7
F06-03	1543.4 - 1903
F08-01	916.3 - 1876.9
F11-02	1734.9 - 1859.9
F12-01	1401.4 - 2076.8
F13-01	1351.4 - 1853.6
F16-05	4153.5 - 4549.9
F17-03	1790.8 - 2185.6
F18-11	3926.6 - 4457.7

Cada amostra possui a informação de profundidade do poço e de cinco perfis elétricos: raios gama (GR), neutrônico (NPHI), sônico (DT), densidade (RHOB) e delta-rhô (DRHO). Os dados já convertidos estão disponibilizados para utilização por outros trabalhos.[2]

A base completa de dados possui 53.181 amostras divididas em 17 tipos litológicos, que estão listados na Tabela 4.2 separados por formação geológica. Apesar de constarem como classes diferentes, os tipos litológicos encontrados possuem em sua formação características semelhantes. A Tabela 4.3, resume as principais características das litologias encontradas.

[2]Os arquivos estão disponíveis no excel (obedecendo o padrão .LAS) e no formato .arff utilizado pelo Weka, e podem ser encontrados em: http://www.inf.puc-rio.br/∼vleite/litologias_perfisEletricos.zip

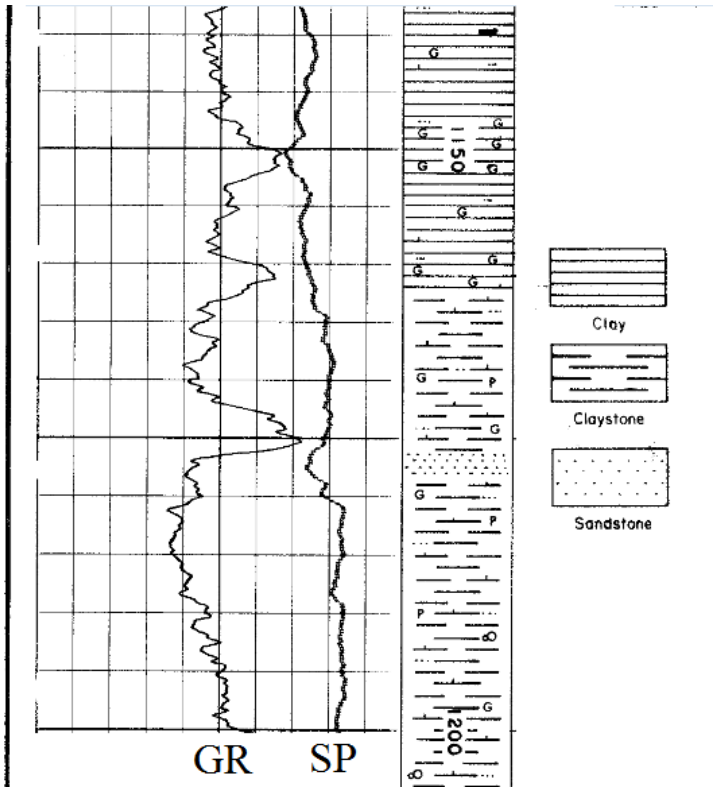

Figura 4.2: Exemplo de um trecho do arquivo do qual foi extraída a informação de litologia

Tabela 4.2: Tipos de Litologias encontradas nos poços selecionados por formação litológica

Formação geológica	Litologias pertencentes
Rocha Sedimentar	Mudstone (lamito) Claystone (argilito) Dolomitic (dolomito) Limestone (calcário) Sandstone (arenito) Shale/Claystone (folhelho/argilito) Argilaceous (argila e xisto) Calcareous (calcário e carbonato) Chert (silica e quartzo) Shale (xisto) Tuff (tufo)
Sal	Anhydrite (anidrita)
Calcário marinho poroso	Chalk
Sedimento	Silt (silte) Sand (areia) Clay (argila) Marl (marga)

Tabela 4.3: Características das litologias encontradas nos poços selecionados

Nome da litologia	Característica
Mudstone (lamito)	Uma pedra de argila semelhante ao xisto mas com a laminação menos desenvolvida
Anhydrite (anidrita)	Mineral de sulfato de cálcio anidro, CaSO4, ocorrendo principalmente em camadas de depósitos de gesso
Claystone (argilito)	Uma granulação fina de rocha sedimentar consistindo principalmente de argila compactada e endurecida. É semelhante ao xisto, mas sem laminações
Dolomitic (dolomito)	Rocha sedimentar consistindo essencialmente de magnésio e calcário
Limestone (calcário)	Rocha sedimentar consistindo principalmente de carbonato de cálcio, depositado com os restos calcários de animais marinhos ou quimicamente precipitado a partir do mar
Chalk (giz)	Grão fino de rocha sedimentar constituída de carbonato de cálcio quase puro, com fragmentos de fósseis de organismos marinhos, geralmente sem um material de cimentação
Sandstone (arenito)	Quaisquer grupos de rochas sedimentares comuns constituídas por grãos de areia consolidadas com materiais como quartzo, hematite e minerais de argila
Shale/Claystone (folhelho/argilito)	Grão fino de rocha formado por compressão de camadas, tanto de rocha laminada (xisto) quando de rocha não laminada (argila)
Silt (silte)	Material sedimentar consistindo de partículas muito finas de tamanho intermediário entre areia e argila
Sand (areia)	Material sedimentar solto composto de rocha ou grãos minerais
Argilaceous (argila e xisto)	Depósito de rocha sedimentar composta por materiais muito finos, como argila, xisto etc.
Calcareous (calcário e carbonato)	Composto por cálcio ou carbonato de cálcio, em proporções semelhantes
Chert (silica e quartzo)	Uma forma de sílica microcristalina geralmente encontrada em faixas ou camadas de pedras em rocha sedimentar.
Shale (xisto)	Grão fino de rocha laminada sedimentar formado por compressão de camadas sucessivas de argila rica em sedimento
Tuff (tufo)	Rocha formada pela compressão de fragmentos de rochas pequenas ejetados de um vulcão
Clay (argila)	Um material sedimentar com grãos bem pequenos
Marl (marga)	Uma mistura de argilas, carbonatos de cálcio, magnésio e restos de conchas

Tabela 4.4: Tipos litológicos e quantidade de amostras classificadas

Tipo Litológico	Quantidade de amostras (Grupo 1)	Quantidade de amostras (Grupo 2)
mudstone	5679	795
anhydrite	830	677
claystone	7877	896
limestone	3713	201
chalk	5963	179
sandstone	2861	736
sand	407	407
argilaceous	1670	536
calcareous	795	795
chert	74	74
shale	1554	522
tuff	344	344
clay	13681	1165
shale/sandstone	2822	659
marl	904	407
sandy	3687	536
silty	320	320

4.2 Procedimentos Adotados

Neste trabalho realizamos dois tipos de experimentos:

- Experimento 1: Comparação entre classificadores individuais e métodos *ensemble*
- Experimento 2: Verificação da automatização do processo de classificação de litologias

Para ambos experimentos foi construída uma base de dados que consistia das informações de profundidade, perfis e litologias. Dessa forma, consideramos como características as informações de profundidade e perfis: Raios Gama (GR), Neutronico (NPHI), Sônico (DT), Densidade (RHOB) e Correção de Densidade (DRHO), enquanto a litologia foi definida como classe.

Para o **Experimento 1** treinamos e classificamos novas amostras com os seguintes classificadores:

- SVM
- MLP
- SVM + *Adaboost*
- MLP + *Adaboost*

– SVM + *Bagging*

– MLP + *Bagging*

Totalizando seis testes. Dessa forma, podemos comparar os resultados dos classificadores individualmente e em conjunto com as duas formas *ensemble* apresentadas. Os parâmetros de cada uma das classificações e maiores detalhes sobre os teste são apresentados na próxima seção.

Já no **Experimento 2** treinamos e classificamos amostras do seguinte modo:

– Treinamento com um grupo pequeno de poços e teste com um poço localizado entre os poços do treinamento. O poço de teste não entrou no treinamento.

O objetivo principal desse teste é simular a classificação em um ambiente real, no qual o número de litologias pertencente a um poço é desconhecido. Maiores detalhes sobre este experimentos são encontrados na Seção 4.4.

4.3 Experimento 1

Para o Experimento 1 dois grupos foram criados, ambos contendo amostras das 17 classes (litologias): Grupo 1, com 53.181 amostras; e Grupo 2, com 9.996 amostras. O Grupo 2 é um subconjunto do primeiro, criado para melhor avaliar o comportamento do *Adaboost*, uma vez que, por usar a mesma base de dados em várias iterações consecutivas, não há a necessidade de uma grande quantidade de amostras. As divisões das amostras por tipos litológicos constam na Tabela 4.4.

Os testes foram executados em máquina 64-bits, com 4gb de RAM, processador intel core 2 duo, 2.40Ghtz, utilizando o Weka (Hall *et. al.*, 2009), que contém uma coleção de algoritmos de aprendizagem de máquina sob a licença GPL (General Public License). O algoritmo do SVM utilizado pertence à LibSVM (Chang *et al*, 2011).

Os doze testes foram realizados com os seguintes tempos, em minutos, conforme a Tabela 4.5:

Tabela 4.5: Tempos obtidos na realização dos experimentos

Amostras		Simples	Adaboost	Bagging
Grupo 1 - 53.181	SVM	272	15148	2870
	MLP	89	598	707
Grupo 2 - 9.996	SVM	1	254	194
	MLP	12	84	138

É possível perceber que os métodos *ensemble* levaram mais tempo para gerar resultados que os classificadores individuais, o que é óbvio se pensarmos que cada

método *ensemble* irá gerar 10 classificadores em vez de apenas um. Podemos perceber também que, apesar do método *Adaboost* ser sequencial, ele levou mais tempo que o *Bagging* apenas quando utilizado o classificador SVM.

Os resultados dos doze testes são apresentados abaixo, cada classe classificada aparece na matriz de confusão associada a uma letra de acordo com a Tabela 4.6.

Tabela 4.6: Nomenclatura utilizada nas matrizes de confusão relacionadas às litologias

Letra associada	Nome da litologia
a	mudstone
b	anhydrite
c	claystone
d	limestone
e	chalk
f	sandstone
g	shale/claystone
h	sand
i	argilaceous
j	calcareous
k	chert
l	shale
m	tuff
n	clay
o	marl
p	sandy
q	silty

4.3.1 Parâmetros utilizados

SVM

Como dito anteriormente, utilizamos o *kernel* RBF neste trabalho. Este *kernel* foi escolhido devido à sua simplicidade, uma vez que o *kernel* linear não era capaz de separar as classes de forma a gerar bons resultados, conforme apresentados na Tabela 4.7, e os demais *kernels* possuem mais parâmetros a serem ajustados, criando uma barreira a mais na obtenção de bons resultados (Hsu *et al*, 2010), além disso, o *kernel* RBF possui uma complexidade menor convergindo mais rapidamente (Ben-Hur and Weston, 2010).

Existem dois parâmetros a serem informados quando se escolhe um *kernel* RBF: C e γ. E não se tem como saber previamente quais os melhores C e γ para um determinado problema, e por isso, alguma heurística de seleção de parâmetros

Tabela 4.7: Resultados iniciais do *kernel* linear - Grupo 1

	Precision	*Recall*	*F-measure*
Kernel Linear	62.3%	62.8%	61.4%

se faz necessária. O objetivo é encontrar C e γ tais que seja possível classificar com precisão dados desconhecidos, uma vez que pode não ser útil alcançar altas taxas para o treinamento (caso em que as classes são realmente conhecidas). Dessa forma, uma estratégia comum é utilizar validação cruzada para realizar o *grid search*.

Grid search representa uma busca exaustiva em um subconjunto do espaço de parâmetros. Vários valores de C e γ são testados e aqueles com a melhor acurácia é escolhido. Um bom método de encontrar bons parâmetros é testar sequências exponencialmente crescentes em C e γ, por exemplo: $C = 2^{-5}, 2^{-3}, \cdots, 2^{15}, \gamma = 2^{-15}, 2^{-13}, \cdots, 2^{3}$ (Hsu *et al*, 2010). O *grid search* pode ser utilizado em conjunto com a validação cruzada, nesta, o conjunto de dados é particionado aleatoriamente em dois subconjuntos, um de teste e outro de treinamento. A validação cruzada diminui a probabilidade dos conjuntos de teste e treinamento serem particionados de forma inadequada, o que pode ocorrer caso essa divisão seja feita manualmente.

Após realizar o *grid search* encontramos os seguintes parâmetros: $c = 8$ e $\gamma = 8192$ para o Grupo 1 e $c = 8$ e $\gamma = 512$ para o Grupo 2.

MLP

O número de nós na camada de entrada e saída é determinado pela dimensionalidade do espaço de características e das respostas desejadas, respectivamente. Dessa forma, é preciso determinar para uma rede MLP, o número de camadas escondidas, o número de nós nessa camada, a taxa de aprendizado e a taxa *momentum*.

Não há uma regra capaz de determinar o melhor número de camadas e nós para uma rede neural. A função da camada escondida é influir na relação entrada-saída da rede de uma forma ampla (Haykin, 1998). Uma RNA com uma camada escondida já é capaz de inferir características sobre o conjunto de dados do qual está tentando aprender. Além disso, quanto mais o erro médio durante o treinamento é passado para as camadas anteriores mais ele se torna impreciso, uma vez que a camada de saída é a única que conhece o erro cometido pela rede. A camada anterior à de saída recebe uma estimativa do erro, a próxima camada escondida recebe uma estimativa da estimativa e assim por diante. Vários testes empíricos foram realizados e como não houve melhorias significativas com o aumento do número de camadas escondidas, neste trabalho usamos uma única camada.

A definição do número de neurônios da camada escondida também não possui regras para a sua determinação (Mas and Flores, 2008). De forma geral, este número é definido empiricamente, entretanto, muitos nós na camada escondida pode levar a rede a "armazenar" os dados de treinamento em vez de extrair características. Ou seja, com muitos nós a rede consegue bons resultados na fase de treinamento, mas resultados ruins na fase de teste pois a rede se ajusta aos dados (*overfiting*). E, por outro lado, poucos nós na camada fazem com que a rede passe mais tempo tentando encontrar uma solução ótima, o que dificulta a convergência (*underfiting*). Neste trabalho foram usados 11 neurônios na camada escondida (uma média entre o número de atributos e o número de classes).

A taxa de aprendizado e *momentum* representam a velocidade com que a rede irá aprender e o quão estável ela será. Uma taxa muito baixa de aprendizado faz com que a rede demore muito tempo para aprender, entretanto, um valor alto faz com que a rede tenha muita oscilação no aprendizado, impedindo a sua convergência. As taxas utilizadas foram 0.3 e 0.2 respectivamente.

Métodos Ensemble

Tanto para o *Adaboost* quanto para o *Bagging* o parâmetro T (número de instâncias criadas) foi igual a 10. E em todos os testes os resultados foram gerados através de uma validação cruzada com 10 partições.

4.3.2 Grupo 1

Classificação com SVM

Ao realizar a classificação utilizando apenas o SVM os resultados obtidos foram:

- Instâncias corretamente classificadas: 90.21%
- Taxa *precision*: 90.3%
- Taxa *recall*: 90.2%
- Taxa *F-measure*: 90.1%

A matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.8. Podemos perceber que as classes a, b, d, e, g, h, i, n e o, tiveram os melhores resultados (menos valores fora das diagonais), com o número de instâncias corretamente classificadas maiores que 90% do total. E percebemos também que as classes com os piores resultados foram f, j, k, m, p e q, essas classes alcançaram número de instâncias corretamente classificadas menores que 80%. As classes com

o maior número de classificações erradas foram k e m, alcançando taxas de 51% e 54%.

Tabela 4.8: Matriz de confusão usando SVM simples - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	5548	0	12	98	4	0	0	0	0	0	0	2	0	14	1	0	0
b	1	827	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c	25	2	6742	5	8	67	2	0	5	6	0	43	13	779	39	138	3
d	64	2	7	3537	11	3	0	0	80	0	0	2	0	3	0	3	1
e	2	0	10	1	5901	0	0	0	29	0	12	0	0	0	8	0	0
f	11	0	62	1	0	2197	401	0	0	0	0	1	0	183	0	5	0
g	0	0	2	0	0	174	2646	0	0	0	0	0	0	0	0	0	0
h	0	0	4	0	0	1	0	397	0	0	0	0	0	4	0	1	0
i	1	0	6	0	24	0	0	0	1636	0	0	0	0	0	2	1	0
j	0	0	1	0	0	0	0	0	20	560	0	8	0	82	10	94	20
k	0	0	0	0	36	0	0	0	0	0	38	0	0	0	0	0	0
l	9	0	29	6	0	5	0	0	0	11	0	1312	0	88	0	87	7
m	0	0	16	0	0	0	0	0	0	0	0	0	189	139	0	0	0
n	18	0	157	3	0	66	0	3	0	34	0	51	23	12930	8	345	43
o	1	0	14	0	25	0	0	0	3	14	0	0	0	16	831	0	0
p	3	0	88	1	1	7	0	1	0	33	0	85	0	1001	0	2467	0
q	0	0	2	0	0	0	0	0	0	15	0	11	0	71	0	0	221

Classificação com SVM e *Adaboost*

Ao realizar a classificação utilizando o SVM juntamente com *Adaboost* os resultados obtidos foram:

- Instâncias corretamente classificadas: 90.78%
- Taxa *precision*: 90.8%
- Taxa *recall*: 90.8%
- Taxa *F-measure*: 90.7%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.9. Observando a matriz vemos que as classes com o maior número de instâncias corretamente classificadas foram a, b, d, e, g, h, i, n e o, cada uma dessas classes tiveram mais amostras corretamente classificadas quando comparada à utilização do classificador simples, entretanto, essa melhoria foi em torno de 0.2%. Já as classes com o maior número de classes incorretamente classificadas foram j, k, m, p e q, pois o método *Adaboost* trouxe melhores resultados para a classe f (de 76% para 81%). E as classes k e m que tinham 51% e 54% passaram a ter 73% e 74% respectivamente.

Tabela 4.9: Matriz de confusão usando SVM e Adaboost - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	5559	0	10	77	6	5	0	0	1	0	0	3	0	15	3	0	0
b	0	828	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c	18	2	7003	5	9	53	1	0	3	4	0	36	11	617	20	90	5
d	76	1	2	3537	14	4	0	0	66	0	0	8	0	2	0	2	1
e	1	0	13	4	5881	0	0	0	32	0	19	0	0	0	13	0	0
f	7	0	57	2	2	2331	294	0	0	0	0	1	0	166	0	1	0
g	0	0	1	0	0	262	2559	0	0	0	0	0	0	0	0	0	0
h	0	0	7	0	0	0	0	395	0	0	0	0	0	5	0	0	0
i	0	0	5	28	39	0	0	0	1592	3	0	0	0	0	3	0	0
j	0	0	5	0	0	0	0	0	10	604	0	13	0	63	10	76	14
k	0	0	0	0	20	0	0	0	0	0	54	0	0	0	0	0	0
l	10	0	40	4	0	4	0	0	0	12	0	1320	1	85	0	75	3
m	0	0	14	0	0	0	0	0	0	0	0	0	257	73	0	0	0
n	11	0	274	2	0	80	0	5	0	29	0	51	57	12614	6	518	34
o	3	0	20	0	27	0	0	0	3	11	0	0	0	20	819	1	0
p	2	0	99	2	0	3	0	1	0	46	0	52	0	783	0	2699	0
q	0	0	1	0	0	0	0	0	0	10	0	8	0	70	0	0	231

Classificação com SVM e *Bagging*

Ao realizar a classificação utilizando o SVM juntamente com *Bagging* os resultados obtidos foram:

- Instâncias corretamente classificadas: 90.25%
- Taxa *precision*: 90.3%
- Taxa *recall*: 90.3%
- Taxa *F-measure*: 90.1%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.10. Na matriz observamos que as classes com o maior número de instâncias corretamente classificadas são: a, b, d, e, g, h, i, n, o. Comparando com o SVM simples, praticamente todas as classes mantiveram os mesmos resultados, nas demais houve uma variação de aproximadamente 2% (para mais e para menos).

Classificação com MLP

Ao realizar a classificação utilizando o MLP os resultados obtidos foram:

- Instâncias corretamente classificadas: 72.76%
- Taxa *precision*: 71.8%
- Taxa *recall*: 72.8%
- Taxa *F-measure*: 71.1%

Tabela 4.10: Matriz de confusão usando SVM e Bagging - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	5546	0	14	94	8	2	0	0	0	0	0	1	0	13	1	0	0
b	0	829	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c	25	2	6767	5	9	67	2	0	5	4	0	43	13	766	36	130	3
d	74	2	4	3533	9	5	0	0	80	0	0	3	0	2	0	1	0
e	1	0	12	1	5899	0	0	0	31	0	12	0	0	0	7	0	0
f	9	0	60	1	1	2219	386	0	1	0	0	2	0	177	0	5	0
g	0	0	1	0	0	197	2624	0	0	0	0	0	0	0	0	0	0
h	0	0	4	0	0	1	0	397	0	0	0	0	0	4	0	1	0
i	0	0	6	0	27	0	0	0	1636	0	0	0	0	0	1	0	0
j	0	0	4	0	0	0	0	0	20	564	0	11	0	79	10	87	20
k	0	0	0	0	39	0	0	0	0	0	35	0	0	0	0	0	0
l	9	0	26	6	0	7	0	0	0	12	0	1324	0	87	0	79	4
m	0	0	12	0	0	0	0	0	0	0	0	0	208	124	0	0	0
n	17	0	167	3	0	65	0	3	0	31	0	54	26	12931	8	339	37
o	1	0	17	0	25	0	0	0	3	8	0	0	0	21	829	0	0
p	5	0	83	3	2	5	0	1	0	33	0	93	0	1022	0	2440	0
q	0	0	1	2	0	0	0	0	0	13	0	9	0	79	0	0	216

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.11. Podemos perceber que as classes b, e e g tiveram os melhores resultados (menos valores fora das diagonais), com o número instâncias corretamente classificadas maiores que 90% do total. Percebemos também que a maioria das classes tiveram resultados confusos, mas as classes com o menor número de instâncias corretamente classificadas são f, j, k, l, m, o, p e q com taxas menores que 50%.

Classificação com MLP e *Adaboost*

Ao realizar a classificação utilizando o MLP juntamente com *Adaboost* os resultados obtidos foram:

- Instâncias corretamente classificadas: 72.76%
- Taxa *precision*: 71.8%
- Taxa *recall*: 72.8%
- Taxa *F-measure*: 71.1%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.12. Podemos perceber que os resultados se mantiveram os mesmos do classificador individualmente.

Tabela 4.11: Matriz de confusão usando MLP simples - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	4484	10	264	161	45	49	0	0	19	9	2	291	3	263	43	31	5
b	1	788	7	0	27	4	0	0	1	0	0	2	0	0	0	0	0
c	111	15	5232	2	41	65	18	5	10	24	4	185	9	1898	53	188	17
d	202	7	23	3074	229	7	0	0	91	5	0	30	0	29	3	3	10
e	49	8	22	90	5590	18	3	2	108	11	5	29	0	7	4	5	12
f	43	5	288	10	38	867	945	10	12	44	3	57	1	420	42	63	13
g	0	0	4	0	0	42	2771	2	0	0	0	0	0	1	0	2	0
h	5	0	24	0	5	15	1	213	0	7	0	4	0	63	2	59	9
i	28	0	4	1	140	2	0	0	1485	0	0	5	0	0	1	3	1
j	31	0	36	0	4	25	0	1	20	92	0	77	0	387	15	56	51
k	0	0	1	0	72	0	0	0	1	0	0	0	0	0	0	0	0
l	154	0	328	0	0	33	3	0	0	8	0	501	2	437	1	43	44
m	6	0	38	0	0	14	0	0	0	0	0	8	4	267	7	0	0
n	119	0	490	0	4	125	10	2	0	74	0	183	16	11953	8	628	69
o	57	0	133	0	116	43	0	0	4	24	0	22	0	74	415	15	1
p	153	0	154	7	5	52	2	1	1	25	0	302	1	1835	1	1132	16
q	4	0	2	0	0	2	0	0	0	8	0	33	0	172	0	4	95

Tabela 4.12: Matriz de confusão usando MLP e Adaboost - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	4484	10	264	161	45	49	0	0	19	9	2	291	3	263	43	31	5
b	1	788	7	0	27	4	0	0	1	0	0	2	0	0	0	0	0
c	111	15	5232	2	41	65	18	5	10	24	4	185	9	1898	53	188	17
d	202	7	23	3074	229	7	0	0	91	5	0	30	0	29	3	3	10
e	49	8	22	90	5590	18	3	2	108	11	5	29	0	7	4	5	12
f	43	5	288	10	38	867	945	10	12	44	3	57	1	420	42	63	13
g	0	0	4	0	0	42	2771	2	0	0	0	0	0	1	0	2	0
h	5	0	24	0	5	15	1	213	0	7	0	4	0	63	2	59	9
i	28	0	4	1	140	2	0	0	1485	0	0	5	0	0	1	3	1
j	31	0	36	0	4	25	0	1	20	92	0	77	0	387	15	56	51
k	0	0	1	0	72	0	0	0	1	0	0	0	0	0	0	0	0
l	154	0	328	0	0	33	3	0	0	8	0	501	2	437	1	43	44
m	6	0	38	0	0	14	0	0	0	0	0	8	4	267	7	0	0
n	119	0	490	0	4	125	10	2	0	74	0	183	16	11953	8	628	69
o	57	0	133	0	116	43	0	0	4	24	0	22	0	74	415	15	1
p	153	0	154	7	5	52	2	1	1	25	0	302	1	1835	1	1132	16
q	4	0	2	0	0	2	0	0	0	8	0	33	0	172	0	4	95

Classificação com MLP e *Bagging*

Ao realizar a classificação utilizando o MLP juntamente com *Adaboost* os resultados obtidos foram:

– Instâncias corretamente classificadas: 75.68%

– Taxa *precision*: 75.1%

– Taxa *recall*: 75.7%

– Taxa *F-measure*: 73.6%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.13. Ao adicionar o *Bagging* ao MLP notamos que aumentou o número de instâncias classificadas corretamente em cada classe com exceção da classe q que teve resultados piores. Entretanto, as classes f, h, j, k, l, m, o, p e q continuaram com resultados piores que 50%.

Tabela 4.13: Matriz de confusão usando MLP e Bagging - Grupo 1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	4732	0	264	191	43	48	0	0	13	3	0	126	0	218	25	16	0
b	1	811	0	0	15	0	0	0	3	0	0	0	0	0	0	0	0
c	85	16	5498	6	36	41	15	3	6	19	2	83	6	1847	43	167	4
d	118	11	18	3182	221	6	0	0	84	0	0	31	0	41	0	0	1
e	16	6	13	103	5709	2	0	0	109	0	0	0	0	0	5	0	0
f	33	0	308	1	23	995	970	2	8	22	0	39	0	390	5	60	5
g	0	0	2	0	0	19	2801	0	0	0	0	0	0	0	0	0	0
h	4	0	16	0	0	14	0	219	2	3	0	2	0	75	0	64	8
i	37	0	2	3	25	0	0	0	1603	0	0	0	0	0	0	0	0
j	29	0	44	0	3	4	0	0	20	100	0	72	0	447	18	14	44
k	0	0	0	0	74	0	0	0	0	0	0	0	0	0	0	0	0
l	126	0	333	0	1	21	2	0	0	1	0	518	0	505	0	31	16
m	0	0	28	0	0	12	0	0	0	0	0	6	7	291	0	0	0
n	69	0	267	0	3	90	1	0	0	31	0	124	2	12543	7	506	38
o	24	1	178	0	124	6	0	0	4	60	0	1	0	74	430	2	0
p	102	0	74	4	7	44	3	0	1	0	0	362	0	2043	0	1046	1
q	0	0	0	0	0	0	0	0	0	0	0	19	0	247	0	0	54

Comparação dos Resultados para o Grupo 1

Ao observarmos os resultados para o Grupo 1, notamos que o SVM sozinho gerou bons resultados atingindo taxas maiores que 90% e que, quando o *Adaboost* foi utilizado conjuntamente, mesmo o SVM não sendo um classificador fraco, houve uma ligeira melhoria na performance (em torno de 0.6%). E quando utilizamos o *Bagging* com o SVM houve uma melhoria quase imperceptível de 0.05%. Entretanto, quando usamos o MLP simples o resultado obtidos foi de aproximadamente 70%,e ao adicionar o *Adaboost* o que notamos foi que houve nenhuma melhoria, e ao utilizar o *Bagging* percebemos o maior porcentagem de melhoria obtida: quase 3%. De maneira específica, podemos perceber que as classes k e m (*chert* e *tuff*) foram as mais difíceis de classificar (em quaisquer um dos

métodos utilizados), isso pode ser resultado da pequena quantidade de amostras dessas classes. A Tabela 4.21 mostra o resumo das classificações para o Grupo 1.

Tabela 4.14: Comparação dos resultados dos classificadores para o Grupo 1

	Simples			*Adaboost*			*Bagging*		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
SVM	90.3%	90.2%	90.1%	90.8%	90.8%	90.7%	90.3%	90.3%	90.1%
MLP	71.8%	72.8%	71.1%	71.8%	72.8%	71.1%	75.1%	75.7%	73.6%

4.3.3 Grupo 2

Classificação com SVM

Ao realizar a classificação utilizando o SVM os resultados obtidos foram:

- Instâncias corretamente classificadas: 93.39%
- Taxa *precision*: 93.5%
- Taxa *recall*: 93.4%
- Taxa *F-measure*: 93.3%

A matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.15, nela podemos notar que poucas instâncias foram classificadas incorretamente, sendo que as classes com piores desempenho foram d e q, com a d sendo ligeiramente melhor que 50%. Por outro lado as classes b, e, g e k tiveram 100% das amostras corretamente classificadas.

Classificação com SVM e *Adaboost*

Ao realizar a classificação utilizando o SVM juntamente com o *Adaboost* os resultados obtidos foram:

- Instâncias corretamente classificadas: 94.78%
- Taxa *precision*: 94.8%
- Taxa *recall*: 94.8%
- Taxa *F-measure*: 94.8%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.16. Ao adicionarmos o *Adaboost* percebemos uma melhora de desempenho no classificador. As classes que o SVM individualmente conseguiu acertar 100% das amostras mantiveram-se com a mesma taxa de acerto, entretanto as classes d e q passaram de 53% e 76% para 70% e 86% respectivamente.

Tabela 4.15: Matriz de confusão usando SVM simples - Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	777	0	0	7	0	4	0	0	0	0	2	0	0	5	0	0	0
b	0	677	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	2	0	863	1	0	2	0	1	0	6	0	4	3	13	1	0	0
d	11	1	0	108	0	0	0	0	80	0	0	0	0	0	0	1	0
e	0	0	0	0	179	0	0	0	0	0	0	0	0	0	0	0	0
f	10	0	0	0	0	709	1	1	0	0	0	1	0	14	0	0	0
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	403	0	0	0	0	0	4	0	0	0
i	1	0	0	0	0	0	0	0	533	2	0	0	0	0	0	0	0
j	0	0	6	0	0	0	0	0	2	691	0	7	0	11	17	38	23
k	0	0	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0
l	0	0	0	0	0	2	0	0	0	3	0	434	1	18	0	54	10
m	0	0	4	0	0	0	0	0	0	0	0	0	324	16	0	0	0
n	0	0	17	0	0	14	0	6	0	16	0	6	6	1075	2	11	12
o	0	0	0	0	0	0	0	0	0	31	0	0	2	1	870	0	0
p	0	0	0	0	0	0	0	0	0	20	0	40	0	10	0	716	0
q	0	0	0	0	0	0	0	0	0	19	0	9	0	48	0	0	244

Tabela 4.16: Matriz de confusão usando SVM e Adaboost - Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	780	0	3	6	0	4	0	0	0	0	1	0	0	1	0	0	0
b	0	677	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	4	0	864	2	0	4	0	1	0	1	0	0	3	15	1	1	0
d	8	0	0	141	0	1	0	0	50	0	0	0	0	0	0	1	0
e	0	0	0	0	179	0	0	0	0	0	0	0	0	0	0	0	0
f	5	0	4	0	0	715	1	0	0	0	0	1	0	10	0	0	0
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	1	403	0	0	0	0	0	3	0	0	0
i	2	0	0	25	0	0	0	0	508	1	0	0	0	0	0	0	0
j	0	0	1	0	0	0	0	0	0	713	0	11	0	10	21	24	15
k	0	0	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0
l	0	0	2	0	0	2	0	0	0	5	0	468	0	11	0	26	8
m	0	0	4	0	0	0	0	0	0	0	0	0	328	12	0	0	0
n	0	0	14	0	0	8	0	5	0	5	0	12	8	1081	2	8	22
o	0	0	0	0	0	1	0	0	0	17	0	0	2	1	883	0	0
p	0	0	0	0	0	0	0	0	0	26	0	24	0	11	0	725	0
q	0	0	0	0	0	0	0	0	0	15	0	5	0	23	0	0	277

Classificação com SVM e *Bagging*

Ao realizar a classificação utilizando o SVM juntamente com o *Bagging* os resultados obtidos foram:

– Instâncias corretamente classificadas: 93.33%

– Taxa *precision*: 93.4%

– Taxa *recall*: 93.3%

– Taxa *F-measure*: 93.2%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.17. Com relação ao classificador SVM individualmente, o método *Bagging* trouxe uma ligeria piora nos resultados (0.06%), assim, a classe b que teve 100% de acerto agora teve uma amostra classificada erroneamente.

Tabela 4.17: Matriz de confusão usando SVM e Bagging - Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	780	0	1	6	0	3	0	0	0	0	2	0	0	3	0	0	0
b	0	676	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c	3	0	864	2	0	2	0	1	0	7	0	3	3	10	1	0	0
d	12	1	0	107	0	0	0	0	80	0	0	0	0	0	0	1	0
e	0	0	0	0	179	0	0	0	0	0	0	0	0	0	0	0	0
f	8	0	1	0	0	710	1	1	0	0	0	1	0	13	0	0	1
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	1	402	0	0	0	0	0	4	0	0	0
i	1	0	0	0	0	0	0	0	534	1	0	0	0	0	0	0	0
j	0	0	9	0	0	0	0	0	2	690	0	6	0	11	14	38	25
k	0	0	0	0	0	0	0	0	0	0	74	0	0	0	0	0	0
l	0	0	0	0	0	2	0	0	0	3	0	437	1	16	0	54	9
m	0	0	3	0	0	0	0	0	0	0	0	0	323	18	0	0	0
n	1	0	21	0	0	16	0	6	0	17	0	7	6	1067	2	12	10
o	0	0	0	0	0	0	0	0	1	29	0	0	2	1	871	0	0
p	1	0	0	0	0	1	0	0	0	21	0	42	1	13	0	707	0
q	0	0	0	0	0	0	0	0	0	18	0	8	0	44	0	0	250

Classificação com MLP

Ao realizar a classificação utilizando o MLP os resultados obtidos foram:

– Instâncias corretamente classificadas: 69.26%

– Taxa *precision*: 70.1%

– Taxa *recall*: 69.3%

– Taxa *F-measure*: 69.2%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.18. Ao observar a matriz notamos que muitas classes tiveram amostras classificadas fora da diagonal principal, sendo que as classes d e j são as classes

com mais instâncias incorretamente classificadas alcançando taxas de 39% e 42% respectivamente. Entretanto a classe g teve todas as suas amostras corretamente classificadas.

Tabela 4.18: Matriz de confusão usando MLP simples - Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	605	0	68	10	1	6	0	2	2	19	11	0	6	13	36	10	6
b	1	659	0	13	1	0	0	1	1	0	1	0	0	0	0	0	0
c	32	2	579	16	0	32	0	15	16	29	1	24	32	53	15	49	1
d	24	9	1	79	1	0	0	1	80	4	0	2	0	0	0	0	0
e	0	1	2	2	164	0	0	0	3	0	6	0	0	1	0	0	0
f	17	3	15	8	7	469	11	49	2	43	0	22	10	53	4	0	23
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	2	0	14	0	0	3	3	349	0	14	0	12	0	8	0	0	2
i	3	0	0	4	0	0	0	0	529	0	0	0	0	0	0	0	0
j	24	0	87	7	2	22	4	12	20	335	1	49	2	100	4	44	82
k	0	0	0	2	0	2	0	0	19	0	51	0	0	0	0	0	0
l	33	0	30	8	0	13	0	6	4	20	0	266	3	54	1	65	19
m	4	0	42	0	0	0	0	0	0	1	0	5	274	18	0	0	0
n	35	0	99	6	1	22	9	33	1	53	0	74	38	687	6	13	88
o	26	3	9	33	30	12	0	17	9	127	0	13	13	15	584	2	11
p	33	0	26	4	0	15	1	5	2	44	0	104	13	50	1	462	26
q	8	0	0	0	0	1	0	1	0	20	0	41	2	73	0	1	173

Classificação com MLP e *Adaboost*

Ao realizar a classificação utilizando o MLP juntamente com o *Adaboost* os resultados obtidos foram:

- Instâncias corretamente classificadas: 77.73%
- Taxa *precision*: 77.7%
- Taxa *recall*: 77.7%
- Taxa *F-measure*: 77.5%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.19. Quando adicionamos o método *Adaboost* ao MLP observamos uma melhoria bastante aparente (mais de 7%) e, nessa matriz percebemos que apenas a classe d teve mais da metade das instâncias incorretamente classificadas, mas, ainda assim, em menor porcentagem que o MLP individualmente: 45%.

Tabela 4.19: Matriz de confusão usando MLP e Adaboost - Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	632	0	65	8	1	3	0	2	2	12	10	1	1	17	33	6	2
b	0	663	0	12	0	0	0	0	1	0	1	0	0	0	0	0	0
c	35	3	705	6	0	2	0	10	2	20	0	12	21	39	2	39	0
d	26	2	3	90	0	0	0	0	80	0	0	0	0	0	0	0	0
e	0	1	2	2	162	0	0	0	3	0	6	0	0	1	2	0	0
f	13	1	17	4	2	583	10	26	0	22	0	11	5	30	3	1	8
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	3	0	17	1	0	3	2	366	0	12	0	1	0	1	0	0	1
i	3	0	0	5	0	0	0	0	527	0	1	0	0	0	0	0	0
j	18	0	50	3	0	7	5	6	20	417	0	34	1	65	37	66	66
k	0	0	0	0	0	0	0	0	15	0	59	0	0	0	0	0	0
l	31	0	20	6	0	11	0	5	0	17	0	299	1	35	1	81	15
m	4	0	31	0	0	0	0	0	0	0	0	0	281	28	0	0	0
n	32	0	71	1	0	38	9	15	0	41	0	59	36	785	6	15	57
o	9	0	8	12	0	2	0	0	2	53	0	0	14	3	793	2	6
p	31	0	20	2	0	7	0	4	1	18	0	88	13	33	0	562	7
q	6	0	0	0	0	0	0	1	0	20	0	32	2	69	0	3	187

Classificação com MLP e *Bagging*

Ao realizar a classificação utilizando o MLP juntamente com o *Bagging* os resultados obtidos foram:

- Instâncias corretamente classificadas: 73.54%
- Taxa *precision*: 74.8%
- Taxa *recall*: 73.5%
- Taxa *F-measure*: 73.5%

E a matriz de confusão com as instâncias classificadas por classe encontra-se na Tabela 4.20. O método *Bagging* proveu melhorias na classificação do MLP, e notamos na matriz que a classe g manteve-se com o acerto em todas as amostras, além disso, a maioria das classes tiveram melhorias, com exceção apenas das classes i e o. Sendo que, a classe i continuou com uma boa taxa de classificação (aproximadamente 98%), entretanto a classe o que teve no classificador individual uma taxa de 64% passou para 59%.

Tabela 4.20: Matriz de confusão usando MLP e Bagging- Grupo 2

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a	625	0	55	6	2	2	0	0	0	10	15	0	13	12	46	6	3
b	0	663	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0
c	41	3	714	10	0	7	0	4	0	11	0	0	31	24	1	50	0
d	29	2	2	87	0	0	0	0	80	0	0	0	0	0	0	1	0
e	0	0	0	0	165	0	0	0	9	0	5	0	0	0	0	0	0
f	12	2	12	5	9	516	11	54	0	47	0	15	4	29	9	1	10
g	0	0	0	0	0	0	659	0	0	0	0	0	0	0	0	0	0
h	1	0	19	6	0	0	1	374	0	2	0	3	0	1	0	0	0
i	2	0	0	5	0	0	0	0	527	0	2	0	0	0	0	0	0
j	18	0	81	0	0	10	2	8	20	364	0	45	0	73	5	72	97
k	0	0	0	1	0	0	0	0	22	0	51	0	0	0	0	0	0
l	17	0	29	3	0	2	0	1	0	10	0	336	0	13	0	109	2
m	0	0	26	0	0	0	0	0	0	0	0	0	311	7	0	0	0
n	30	0	91	1	3	14	9	16	0	30	0	120	67	731	1	7	45
o	4	1	2	64	23	13	0	2	8	217	0	5	17	2	539	2	5
p	14	0	15	0	0	2	2	4	0	20	0	160	12	27	0	510	20
q	3	0	0	0	0	0	0	1	0	8	0	54	0	74	0	0	180

Comparação dos Resultados para o Grupo 2

Ao observarmos os resultados para o Grupo 2, notamos que o SVM sozinho gerou novamente resultados tão bons quanto no Grupo1, atingindo taxas acima de 90% e que, quando o *Adaboost* foi utilizado conjuntamente houve uma ligeira melhoria na performance (em torno de 1%). E quando utilizamos o *Bagging* com o SVM houve um decréscimo na performance de mínimos 0.03%. Observamos ainda que quando usamos o MLP simples o resultado obtido foi de menos de 70%, entretanto a utilização do *Adaboost* trouxe uma melhoria significativa (em torno de 7%), apresentando-se como o melhor resultado obtido para o MLP, e ao utilizar o *Bagging* percebemos uma boa melhoria: um pouco maior que 4%. A Tabela 4.21 mostra o resumo das classificações para o Grupo 2.

Tabela 4.21: Comparação dos resultados dos classificadores para o Grupo 2

	Simples			*Adaboost*			*Bagging*		
	Precision	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*	*Precision*	*Recall*	*F-measure*
SVM	93.5%	93.4%	93.3%	94.8%	94.8%	94.8%	93.4%	93.3%	93.2%
MLP	70.1%	69.3%	69.2%	77.7%	77.7%	77.5%	74.8%	73.5%	73.5%

4.3.4 Características dos Resultados

É interessante notar como a variação na quantidade dos dados de treinamento (Grupo 1 e Grupo 2) causa uma grande diferença nos resultados. Notamos que o classificador SVM se manteve sempre melhor que o MLP, e que as melhorias providas por métodos *ensembles* para um classificador não considerado fraco são de pouca expressividade. Além do mais, devido às características do método *Adaboost* percebemos que ele se sobressaiu quando a base de dados era menor (Grupo 2), contrariamente ao *Bagging* que para uma base de dados pequena quando utilizado com o SVM teve resultados piores que o classificador individualmente.

Outra observação interessante a ser notada é que a grande concentração de erro dos classificadores, perceptível através das matrizes de confusão é que, as classes que mais se misturam (ou seja a taxa de predição errada entre elas é maior) são: *chert, shale, tuff, clay, marl, sandy* e *silty*. Uma explicação para isso é o fato de que essas litologias representam sedimentos, ou seja, são decomposição de outras rochas, tendo dessa forma características semelhantes.

4.4 Experimento 2

Uma vez que o SVM teve melhores resultados, em relação ao MLP e métodos *ensemble*, os testes deste experimento foram realizados apenas com o SVM.

Para verificar a capacidade de automatização do processo de identificação de litologias, dois testes foram realizados. No primeiro teste, o poço do qual se deseja descobrir a litologia está entre poços com litologias conhecidas. Já no segundo teste, o poço não possui poços ao redor com litologias conhecidas, situando-se na borda do conjunto.

4.4.1 Poço de teste central

Para o primeiro teste seis poços foram selecionados, sendo cinco para o treinamento (F01-01, F03-03, F06-03, F11-02 e F12-01) e um para teste (F08-01), conforme mostra a Figura 4.3.

Os cinco poços possuem amostras de quinze litologias num total de 15.652 amostras. A Tabela 4.22 apresenta as litologias e as quantidades de amostras de cada litologia encontradas nos poços de treinamento.

Após treinar o SVM com os dados dos cinco poços e realizar a classificação, obteve-se uma taxa de acerto de aproximadamente 73%. Uma vez que os poços encontram-se em uma mesma área da bacia, imagina-se que eles possuam características semelhantes, e assim, que seja possível encontrar a litologia em

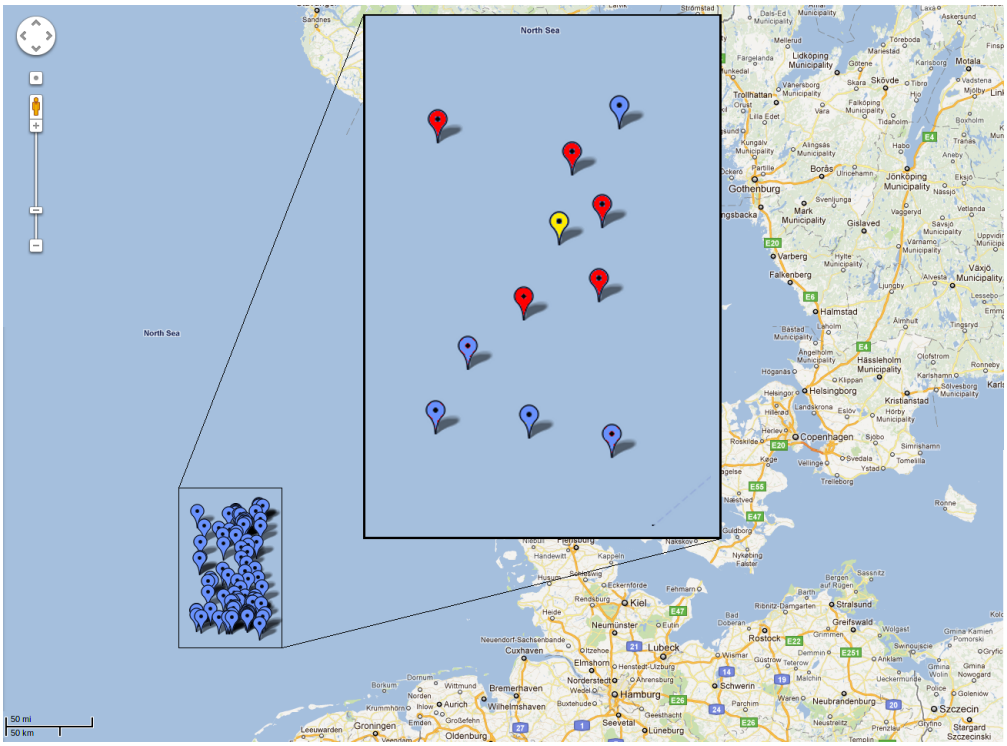

Figura 4.3: Mapeamento dos poços utilizados para treinamento (vermelhos) e teste (amarelo). Os pontos azuis representam a posição dos demais poços que foram utilizados neste trabalho.

Tabela 4.22: Quantidade de amostras por litologias nos poços de treinamento

	Litologia	Quantidade de amostras
1	mudstone	3775
2	limestone	3580
3	claystone	717
4	chalk	1913
5	sandstone	770
6	sand	19
7	argilaceous	1670
8	calcareous	118
9	chert	74
10	shale	588
11	tuff	75
12	clay	1648
13	marl	275
14	sandy	340
15	silty	90

poços sem essa informação. Enquanto os cinco poços utilizados no treinamento cobriam quinze tipos litológicos, o poço de teste continha amostras de cinco tipos litológicos, em um total e 5.083 amostras, distribuídos conforme a Tabela 4.23.

Tabela 4.23: Quantidade de amostras por litologias no poço de teste

	Litologia	Quantidade de amostras
1	sandstone	1308
2	shale	444
3	tuff	100
4	clay	2741
5	marl	490

Ao analisarmos os resultados da classificação por classe, conforme mostra a Tabela 4.24, percebemos que, apesar dos poços de treinamento englobarem as classes existentes no poço de teste, a quantidade de indivíduos de cada classe pode não ser suficiente, como por exemplo no caso da litologias tuff que obteve uma taxa de acerto de 0.4%.

Tabela 4.24: Percentual de acerto para as cinco classes conhecidas pelo classificador

	Litologia	Taxa de acerto
1	sandstone	59.93%
2	shale	85.81%
3	tuff	0.4%
4	clay	80.88%
5	marl	75.10%

Assim, para tentar melhorar o resultado provido pelo classificador, adicionamos mais indivíduos das litologias utilizadas no treinamento, conforme mostra a Tabela 4.25. Conhecer os poços ao redor do poço de interesse, auxilia o treinamento, uma vez que as litologias que não constam nesses poços não precisam ser incluídas no treinamento, diminuindo assim o tempo necessário para treinar o classificador, bem como diminuindo o número de classes que o classificador precisa aprender. O resultado da inclusão de mais amostras no treinamento fez com que o resultado melhorasse consideravelmente, alcançando 93.7%, o detalhamento do resultado por classe pode ser visto na Tabela 4.26.

Tabela 4.25: Quantidade de amostras por litologias nos poços de treinamento

	Litologia	Quantidade de amostras
1	mudstone	5679
2	limestone	3713
3	claystone	4070
4	chalk	5963
5	sandstone	1553
6	sand	407
7	argilaceous	1670
8	calcareous	795
9	chert	74
10	shale	1110
11	tuff	244
12	clay	6327
13	marl	414
14	sandy	3687
15	silty	320

Tabela 4.26: Percentual de acerto para as cinco classes conhecidas pelo classificador

	Litologia	Taxa de acerto
1	sandstone	84.2%
2	shale	90.1%
3	tuff	98.2%
4	clay	98.3%
5	marl	95.7%

4.4.2 Poço de teste na borda

Para esse teste foram selecionados quatro poços (F11-02, F13-01, F16-05 e F18-11) para treinamento e um quinto poço (F17-03) para teste, conforme mostra a Figura 4.4.

Os quatro poços possuem amostras de sete classes (litologias) num total de 3.919 amostras. A Tabela 4.27 apresenta as litologias e as quantidades de amostrasem cada litologia.

Após treinar o SVM com os dados dos quatros poços e realizar a classificação, obteve-se uma taxa de acerto de aproximadamente 33%. Isso se deve ao fato de que o poço utilizado no teste possui litologias diferentes das informadas para o treinamento. Enquanto os quatro poços utilizados no treinamento possuem sete tipos litológicos, o poço de teste possui amostras de treze tipos litológicos distribuídos conforme a Tabela 4.28.

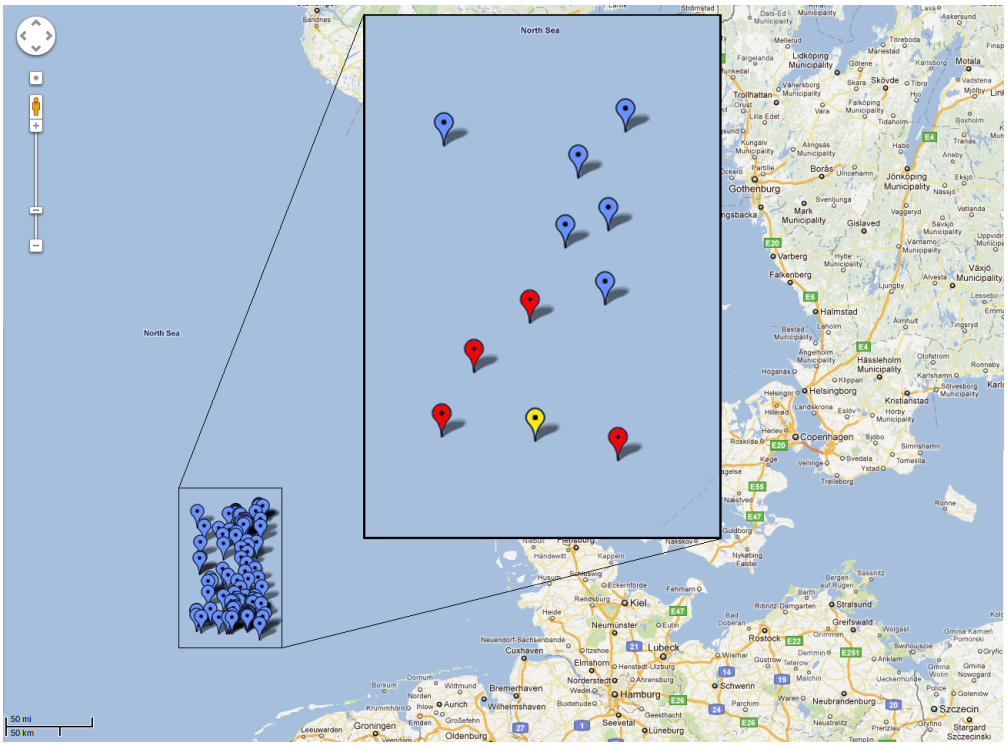

Figura 4.4: Mapeamento dos poços utilizados para treinamento (vermelhos) e teste (amarelo). Os pontos azuis representam a posição dos demais poços que foram utilizados neste trabalho.

Tabela 4.27: Quantidade de amostras por litologias nos poços de treinamento

	Litologia	Quantidade de amostras
1	limestone	50
2	claystone	2325
3	sandstone	990
4	sand	217
5	clay	187
6	sandy	75
7	tuff	75

Podemos notar que o classificador não teve informações suficientes para classificar seis tipos litológicos, o que representam 43.91% do total de amostras a serem classificadas. Considerando então apenas as classes que o classificador conhecia, a taxa de acerto calculada é de aproximadamente 58%. Mesmo considerando apenas as sete classes utilizadas no treinamento o resultado foi abaixo do esperado, possivelmente devido à baixa quantidade de amostras em algumas classes utilizadas no treinamento. A Tabela 4.29 mostra o percentual de acerto para as sete classes conhecidas pelo classificador.

Dessa forma, um segundo teste foi realizado ampliando o número de poços de treinamento para aumentar tanto o número de litologias quanto o número de

Tabela 4.28: Quantidade de amostras por litologias no poço de teste

	Litologia	Quantidade de amostras
1	coal	79
2	shale	571
3	clay	530
4	silty	230
5	calcareous	202
6	claystone	252
7	tuff	19
8	bituminous	101
9	sandstone	295
10	limestone	83
11	sand	18
12	sandy	513
13	siltstone	156

Tabela 4.29: Percentual de acerto para as classes conhecidas pelo classificador

	Litologia	Taxa de acerto
1	limestone	53%
2	claystone	100%
3	sandstone	100%
4	sand	83.33%
5	clay	14.23%
6	sandy	14.23%
7	tuff	63.15%

indivíduos para cada litologia. Assim, sete poços (F01-01, F03-07, F06-03, F12-01, F13-01, F16-05 e F18-11) foram selecionados para o treinamento e o manteve-se o mesmo poço (F17-03) para teste. Os poços selecionados e o poço de teste são mostrados na Figura 4.5.

Os sete poços continham amostras de dezessete litologias num total de 40.530 amostras. A Tabela 4.30 apresenta as litologias e as quantidades de amostras em cada litologia. Ao realizar a classificação do poço F17-03, a taxa de acerto passou a ser de 87.37%. A taxa de acerto por classe é apresentada na Tabela 4.31.

Podemos notar que o aumento na quantidade de amostras no treinamento melhorou o resultado da classificação consideravelmente, entretanto, três classes (bituminous, coal e siltstone) não tiveram amostras no treinamento e, portanto, não tiveram amostras classificadas no teste. Dessa forma, se levarmos em consideracão apenas as classes que foram treinadas, o percentual de taxa de acerto passa a ser 98.14%.

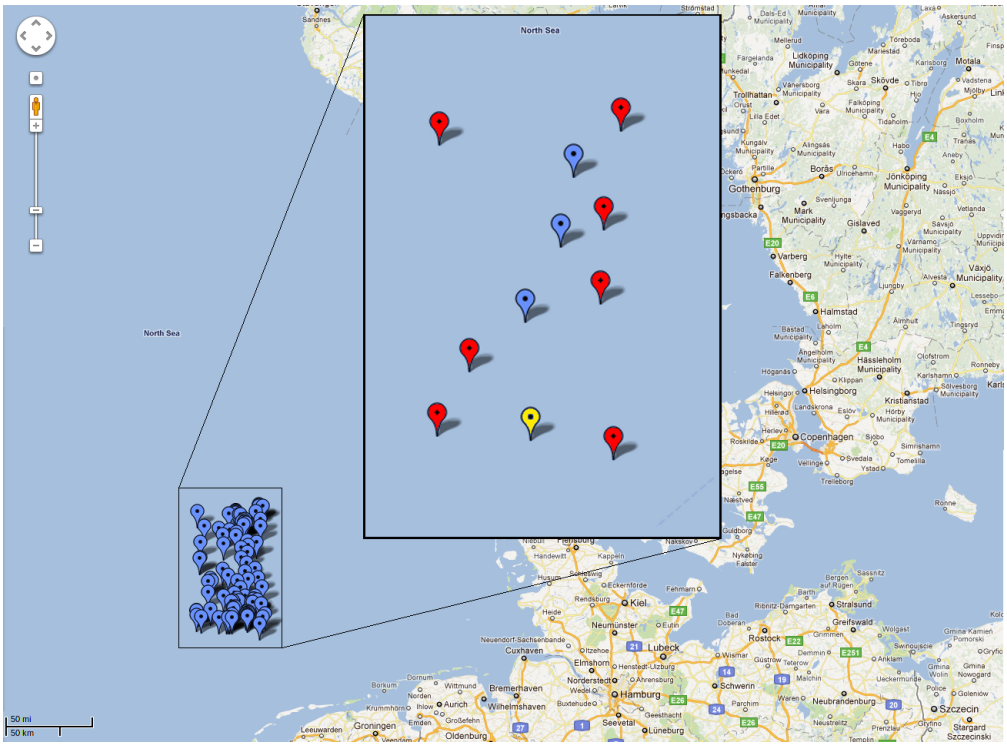

Figura 4.5: Mapeamento dos poços utilizados para treinamento (vermelhos) e teste (amarelo). Os pontos azuis representam a posição dos demais poços que foram utilizados neste trabalho.

Tabela 4.30: Quantidade de amostras por litologias nos poços de treinamento

	Litologia	Quantidade de amostras anterior	Quantidade de amostras atual
1	mudstone	-	5679
2	anhydrite	-	830
3	claystone	2325	3568
4	limestone	50	3630
5	chalk	-	5566
6	sandstone	990	1258
7	shale/claystone	-	2822
8	sand	217	389
9	argilaceous	-	1670
10	calcareous	-	593
11	chert	-	74
12	shale	-	539
13	tuff	75	150
14	clay	187	10223
15	marl	-	350
16	sandy	75	3099
17	silty	-	90

4.4.3

Tabela 4.31: Percentual de acerto para as classes conhecidas pelo classificador

	Litologia	Taxa de acerto anterior	Taxa de acerto atual
1	shale	-	100%
2	clay	14.23%	100%
3	silty	-	100%
4	calcareous	-	100%
5	claystone	100%	80.16%
6	tuff	63.15%	100%
7	sandstone	100%	100%
8	limestone	53%	100%
9	sandy	14.23%	100%
10	sand	83.33%	100%

Considerações

Em um caso real de classificação de litologias não temos como garantir a quantidade de litologias que serão encontradas em um determinado poço. Dessa forma, para automatizar o processo e ter um bom resultado se faz necessário possuir uma base de treinamento que contemple todas as classes conhecidas para uma dada bacia. Como resultado do Experimento 2, notamos que é possível identificar as litologias de um determinado poço, desde que tenhamos uma base com representação suficiente de cada classe para a identificação.

5
Conclusão e Trabalhos Futuros

Ao observar os resultados do Experimento 1 notamos primeiramente que SVM obteve resultados sempre melhores que MLP, inclusive se compararmos as medidas de desempenho do classificador individual SVM *versus* MLP com a utilização dos métodos *ensemble*. Essa variação demonstra que o SVM é muito melhor na tarefa de classificação, um dos fatores que podem levar a este resultado é que MLP implementa uma estratégia de aproximação global, enquanto o SVM utiliza-se de aproximação local, além de ter uma formulação de aprendizado estatístico. Entretanto,os métodos *ensemble* de *bagging* e *adaboost* provêem melhorias nos resultados da classificação de litologias quando da utilização da rede neural MLP, atingindo taxas entre 4 e 7%, o que pode justificar a escolha dos métodos *ensembles* apesar do maior gasto de recursos para a obtenção de resultados. Com base nas taxas apresentadas, podemos notar que a taxa de *recall* para o MLP foi melhor no grupo com mais informações para treinamento, entretanto quando utilizamos métodos *ensemble* houve uma melhoria nos resultados com exceção do Grupo 1 com *Adaboost*. Isso sugere que redes MLP não melhoram quando muitos dados com informações semelhantes são utilizados, entretanto no Grupo 2, as melhorias providas pelo *Adaboost* são expressivas.

O SVM demonstra-se um melhor classificador para identificação de litologias com grande variação litológica, conforme demonstram as matrizes de confusão apresentadas e a utilização de métodos *ensembles* proveu pequenas melhorias nas medidas de desempenho, e em alguns casos, obteve as mesmas taxas que o classificador individualmente, tornando a principal diferença entre eles o grande consumo de recursos, tornando a utilização de *ensembles* com SVM praticamente inviável.

Ao comparar os métodos ensembles notamos que o *Adaboost* teve melhores desempenhos que *Bagging*, podemos atribuir essa melhoria à maneira como eles são construídos. Uma vez que o *Adaboost* reajusta os pesos das amostras para a classificação, sempre dando mais importância para as amostras classificadas erroneamente, se tornando melhor em casos específicos, enquanto o *Bagging* cria subconjuntos da base de dados, podendo repetir alguns dados, mas também podendo não conter amostras de outros. Essas características fazem o *Adaboost* ser

muito sensível a ruídos, entretanto, este não é o caso da base de dados usada no treinamento.

De forma geral, podemos concluir que, neste contexto, utilizar SVM é melhor para a tarefa de classificação que MLP, e que existe um grande *trade-off* entre precisão e gasto de recursos quando utiliza-se métodos *ensembles*.

Além disso, com os resultados do Experimento 2 percebemos que a automatização do processo é possível, sendo condição necessária ter uma base com representação suficiente de cada litologia. O conhecimento das litologias dos poços ao redor do poço no qual deseja-se fazer a classificação, auxilia à medida que se pode focar em um número limitado de litologias, em vez de realizar o treinamento com todas as litologias disponíveis na bacia. Esta automatização por mais que não represente com exatidão o resultado final, serviria como um guia para o geólogo, que terá seu trabalho simplificado.

Podemos citar ainda como contribuição deste trabalho a criação de uma base pública com as informações de perfis e litologias, tanto no formato .TXT quanto no formato .ARFF (utilizado pelo *Weka*).

Como trabalho futuro, pretendemos analisar uma base de dados ruidosa, a fim de verificar o comportamento do método *Bagging*. Outra modificação interessante que deve melhorar os resultados do MLP é agregar ainda mais as litologias classificadas. Uma ideia a ser analisada é fazer um subprocesso de classificação, ou seja, realizar uma classificação em um nível "macro" e em seguida, para cada nível realizar uma nova classificação para melhor separar os resultados. Esta ideia está melhor especificada na Figura 5.1.

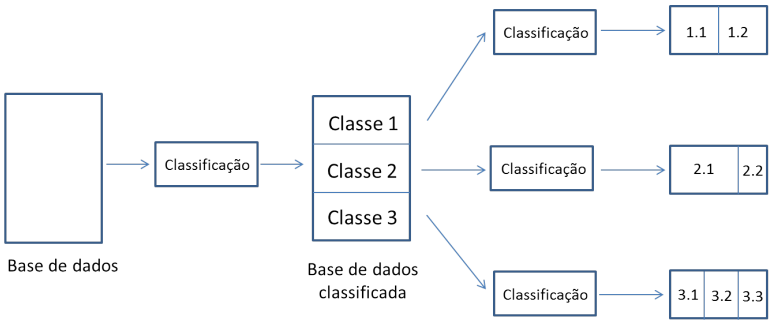

Figura 5.1: Modelo de Classificação em níveis

Além disso, a utilização de um *grid*, e a modificação dos algoritmos para serem paralelizados, permitiria uma obtenção de resultados mais rapidamente do que os demonstrados neste trabalho, permitindo assim um número maior de experimentos e testes.

Referências Bibliográficas

[Al-Anazi *et al*, 2010] AL-ANAZI, A.; IDGATES. **On the capability of Support Vector Machines to Classify Lithology from Well Logs**. Natural Resources Research, 19(2):125–139, 2010.

[Anthony *et. al.*, 2007] ANTHONY, G.; GREG, H. ; TSHILIDZI, M.. **Image classification using svms: One-against-one vs one-against-all**. Proccedings of the 28th Asian Conference on Remote Sensing, 2:801–806, 2007.

[Antunes, 2010] ANTUNES, P. T.. **Curso de Geologia do Petróleo para não Geólogos**, Agosto 2010.

[Baranauskas and Monard, 2000] BARANAUSKAS, J. A.; MONARD, M. C.. **Reviewing some machine learning concepts and methods**. Technical Report 102, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2000.

[Ben-Hur and Weston, 2010] BEN-HUR, A.; WESTON, J.. **A user's guide to support vector machines**, 2010.

[Breiman, 1996] BREIMAN, L.. **Bagging predictors**. In: Machine Learning, volume 24, p. 123–140, 1996.

[Carrasco *et al*, 2003] CARRASCO, A.; TOMASO, K. ; CARRASQUILLA, A.. **Inversão unidimensional de perfis geofísicos de indução na caracterização dos turbiditos da bacia de almada-ba**. Revista Brasileira de Geofísica, 21(3):259–267, 2003.

[Chang *et al*, 2011] CHANG; CHIH-CHUNG; LIN ; CHIH-JEN. **LIBSVM: A library for support vector machines**. ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software disponível em `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cortes and Vapnik, 1995] CORTES, C.; VAPNIK, V.. **Support-vector networks**. In: Machine Learning, p. 273–297, 1995.

[Cunha, 2002] CUNHA, E. S.. **Identificação de litofácies de poços de petróleo utilizando um método baseado em redes neurais artificiais**. Master's thesis, UFCG, Paraíba, Agosto 2002.

[Dietterich, 2000] DIETTERICH, T. G.. **Ensemble methods in machine learning**. In: Proceedings of the First International Workshop on Multiple Classifier Systems, MCS '00, p. 1–15, London, UK, UK, 2000. Springer-Verlag.

[Duarte, 2003] DUARTE, O. O.. **Dicionário enciclopédico Inglês-Português de geofísica e geologia**. Sociedade Brasileira de Geofísica, SBGf, Rio de Janeiro, 2 edition, 2003. ISBN: 85-88690-07-1.

[Flexa *et al*, 2004] FLEXA, R. T.; ANDRADE, A. ; CARRASQUILLA, A.. **Identificação de Litotipos nos perfis de poço do Campo de Namorado (Bacia de Campos, Brasil) e do Lago do Maracaibo (Venezuela) aplicando Estatística Multivariada**. Revista Brasileira de Geociências, 34(4):571–578, Dezembro 2004.

[Freund and Schapire, 1996] FREUND, Y.; SCHAPIRE, R. E.. **Experiments with a New Boosting Algorithm**. In: International Conference on Machine Learning, p. 148–156, 1996.

[Hall *et. al.*, 2009] HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P. ; WITTEN, I. H.. **The weka data mining software: an update**. SIGKDD Explorations Newsletter, 11(1):10–18, Nov. 2009.

[Hansen and Salamon, 1990] HANSEN, L. K.; SALAMON, P.. **Neural network ensembles**. IEEE Trans. Pattern Anal. Mach. Intell., 12(10):993–1001, Oct. 1990.

[Hauskrecht, 2004] HAUSKRECHT, M.. **Cs 2750 machine learning lecture 23: Ensemble methods. bagging and boosting**, 2004.
Available: http://www.cs.pitt.edu/ milos/courses/cs2750-Spring04/lectures/class23.pdf.

[Haykin, 1998] HAYKIN, S.. **Neural Networks: A Comprehensive Foundation (2nd Edition)**. Prentice Hall, 2 edition, jul 1998.

[Hearst *et al*, 1998] HEARST, M. A.; SCHÖLKOPF, B.; DUMAIS, S. T.; OSUNA, E. ; PLATT, J.. **Trends and controversies - support vector machines.** IEEE Intelligent Systems, 13(4):18–28, 1998.

[Herbrich, 2001] HERBRICH, R.. **Learning Kernel Classifiers: Theory and Algorithms**. MIT Press, Cambridge, MA, USA, 2001.

[Hsieh *et al*, 2005] HSIEH, B.-Z.; LEWIS, C. ; LIN, Z.-S.. **Lithology identification of aquifers from geophysical well logs and fuzzy logic analysis: Shui–Lin Area, Taiwan**. Computers & Geosciences, 31(3):263–275, 2005.

[Hsu and Lin, 2002] HSU, C.-W.; LIN, C.-J.. **A comparison of methods for multiclass support vector machines**. IEEE Transactions on Neural Networks, 13:415–425, 2002.

[Hsu *et al*, 2010] HSU, C.-W.; CHANG, C.-C. ; LIN, C.-J.. **A Practical Guide to Support Vector Classification**, 2010.

[Lorena and Carvalho, 2007] LORENA, A. C.; DE CARVALHO, A. C. P. L. F.. **Uma introdução às support vector machines**. Revista de Informática Teórica e Aplicada, 14(2):43–67, 2007.

[MATLAB, 2001] **Fuzzy logic toolbox user's guide**. The MathWorks, Inc., 2001.

[Mas and Flores, 2008] MAS, J. F.; FLORES, J. J.. **The application of artificial neural networks to the analysis of remotely sensed data**. International Journal of Remote Sensing, 29(3):617–663, 2008.

[Milgram *et. al.*, 2006] MILGRAM, J.; CHERIET, M. ; SABOURIN, R.. **"One Against One"or "One Against All": Which One is Better for Handwriting Recognition with SVMs?** In: Lorette, G., editor, 10th International Workshop on Frontiers in Handwriting Recognition, La Baule (France), Oct. 2006. Université de Rennes 1, Suvisoft.

[Milani et al, 2007] MILANI, E. J.; RANGEL, H. D.; , G. V. B.; STICA, J. M.; WINTER, W. R.; CAIXETA, J. M. ; DA CRUZ PESSOA NETO, O.. **Boletim de Geociências da Petrobrás**, volume 15. Maio/Nov 2007.

[Nilsson, 1998] NILSSON, N. J.. **Introduction to Machine Learning**. Stanford University, 1998.

[Optiz and Maclin, 1999] OPITZ, D.; MACLIN, R.. **Popular ensemble methods: An empirical study**. Journal of Artificial Intelligence Research, 11:169–198, 1999.

[Refaeilzadeh *et al*, 2009] REFAEILZADEH, P.; TANG, L. ; LIU, H.. **Cross-validation**. In: Encyclopedia of Database Systems, p. 532–538. 2009.

[Rojas, 1996] ROJAS, R.. **Neural Networks: A Systematic Introduction**. Springer, 1 edition, July 1996.

[Rokach, 2005] ROKACH, L.. **Ensemble methods for classifiers**. In: Maimon, O.; Rokach, L., editors, The Data Mining and Knowledge Discovery Handbook, p. 957–980. Springer, 2005.

[Rumelhart *et. al.*, 1986] RUMELHART, D. E.; HINTON, G. E. ; WILLIAMS, R. J.. **Learning representations by back-propagating errors**. Nature, 323(Oct):533–536+, 1986.

[Santos *et al*, 2002] SANTOS, R.; VELLASCO, M.; ARTOLA, F. ; DA FONTOURA, S.. **Lithology recognition by neural network ensembles**. In: Bittencourt, G.; Ramalho, G., editors, Advances in Artificial Intelligence, volume 2507 de **Lecture Notes in Computer Science**, p. 421–429. Springer Berlin / Heidelberg, 2002.

[Santos *et al*, 2003] SANTOS, R.; VELLASCO, M.; ARTOLA, F. ; DA FONTOURA, S.. **Neural net ensembles for lithology recognition**. In: Windeatt, T.; Roli, F., editors, Multiple Classifier Systems, volume 2709 de **Lecture Notes in Computer Science**, p. 161–161. Springer Berlin / Heidelberg, 2003.

[Scholkopf and Smola, 2001] SCHOLKOPF, B.; SMOLA, A. J.. **Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond**. MIT Press, Cambridge, MA, USA, 2001.

[Semolini, 2002] SEMOLINI, R.. **Support vector machines, inferência transdutiva e o problema de classificação**. Master's thesis, Universidade Estadual de Campinas, 2002.

[Sewell, 2008] SEWELL, M.. **Ensemble Learning**. Technical report, 2008.

[Thomas, 2004] THOMAS, J. E.. **Fundamentos de Engenharia de Petróleo**. Interciência, Rio de Janeiro, 2004.

[Vapnik, 1995] VAPNIK, V.. **The Nature of Statistical Learning Theory**. Springer-Verlag, New York, 1995.

[Zhou *et. al.*, 2002] ZHOU, Z.-H.; WU, J. ; TANG, W.. **Ensembling neural networks: many could be better than all**. Artificial Intelligence, 137(1-2):239–263, May 2002.

[Zhou, 2009] ZHOU, Z.-H.. **Ensemble**. Encyclopedia of Database System, p. 988–991, 2009.