Universidade Federal de Pernambuco
Centro de Informática

Graduação em Engenharia da Computação

Análise comparativa de técnicas de seleção de protótipos

Dayvid Victor Rodrigues de Oliveira

Trabalho de Graduação

Recife
22 de novembro de 2011

Universidade Federal de Pernambuco
Centro de Informática

Dayvid Victor Rodrigues de Oliveira

Análise comparativa de técnicas de seleção de protótipos

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Prof. Dr. George Darmiton*

Recife
22 de novembro de 2011

Eu dedico este trabalho a João Rodrigues de Silva, meu avô.

Agradecimentos

Agradeço a Deus.

What can I give back to God, for the blessings You pour out on me?
—BONO (Boston, 2001)

Resumo

RESUMO

Palavras-chave: PORTUGUES

Abstract

ABSTRACT

Keywords: INGLES

Sumário

1 Introdução **1**
1.1 Motivação e Contextualização 1
1.1.1 Histórico 1
1.1.2 Seleção de Protótipos 2
1.1.3 Bases Desbalanceadas 4
1.2 Objetivo 4
1.3 Estrutura do Trabalho 4

2 Técnicas de Seleção de Protótipos **5**
2.1 ENN 5
2.2 CNN 7
2.3 Tomek Links 9
2.4 OSS 11
2.5 LVQ 13
2.5.1 LVQ 1 14
2.5.2 Optimized-learning-rate LVQ 14
2.5.3 LVQ 2.1 15
2.5.4 LVQ 3 16
2.6 SGP 1 17
2.6.1 Fator de Generalização 20
2.7 SGP 2 20

3 Análise em Bases Artificiais **24**
3.1 Bases Artificiais 24
3.2 Análise Individual 27
3.2.1 ENN 27
3.2.2 CNN 28
3.2.3 Tomek Links 31
3.2.4 OSS 32
3.2.5 LVQ 34
3.2.5.1 LVQ 1 34
3.2.5.2 LVQ 2.1 37
3.2.5.3 LVQ 3 39
3.2.6 SGP 1 40
3.2.7 SGP 2 42

Lista de Figuras

1.1 Exemplo do NPC 3

2.1 ENN aplicado com K=3 6
2.2 Exemplo da aplicação do CNN 8
2.3 Exemplo da aplicação do Tomek Links 10
2.4 Base Original antes da aplicação do OSS 12
2.5 Aplicação da primeira etapa do OSS, o CNN 12
2.6 Resultado final do OSS, após aplicação da segunda etapa, o Tomek Links 13
2.7 Etapa da divisão de grupos [dSPC08]. 18

3.1 Base Original com 10% de desbalanceamento e intersecção mínima 24
3.2 Base Original com 10% de desbalanceamento e intersecção máxima 25
3.3 Base Original com 5% de desbalanceamento e intersecção mínima 26
3.4 Base Original com 5% de desbalanceamento e intersecção máxima 26
3.5 ENN sobre sobre base com total sobreposição de classes e 10% de desbalanceamento 28
3.6 CNN sobre sobre base sem sobreposição de classes e 5% de desbalanceamento 30
3.7 CNN sobreposição III de classes e 5% de desbalanceamento 30
3.8 Tomek Links sobreposição III de classes e 10% de desbalanceamento 32
3.9 OSS com sobreposição III de classes e 5% de desbalanceamento 33
3.10 LVQ 1 com sobreposição de classes III e 5% de desbalanceamento 35
3.11 LVQ 1 com sobreposição de classes V e 5% de desbalanceamento 35
3.12 LVQ 1 com sobreposição de classes V e 10% de desbalanceamento 36
3.13 LVQ 2.1 sem sobreposição de classes e 10% de desbalanceamento 38
3.14 LVQ 1 sem sobreposição de classes e 10% de desbalanceamento 38
3.15 LVQ 3 sem sobreposição de classes e 10% de desbalanceamento 40
3.16 SGP 1 com total sobreposição de classes e 5% de desbalanceamento 41
3.17 SGP 2 com total sobreposição de classes e 5% de desbalanceamento 43

Lista de Tabelas

3.1 ENN com nível de desbalanceamento 5% 27
3.2 ENN com nível de desbalanceamento 10% 27
3.3 CNN com nível de desbalanceamento 5% 29
3.4 CNN com nível de desbalanceamento 10% 29
3.5 Tomek Links com nível de desbalanceamento 5% 31
3.6 Tomek Links com nível de desbalanceamento 10% 31
3.7 OSS com nível de desbalanceamento 5% 33
3.8 OSS com nível de desbalanceamento 10% 33
3.9 LVQ1 com nível de desbalanceamento 5% 36
3.10 LVQ 1 com nível de desbalanceamento 10% 36
3.11 LVQ 2.1 com nível de desbalanceamento 5% 37
3.12 LVQ 2.1 com nível de desbalanceamento 10% 37
3.13 LVQ 3 com nível de desbalanceamento 5% 39
3.14 LVQ 3 com nível de desbalanceamento 10% 39
3.15 SGP 1 com nível de desbalanceamento 5% 41
3.16 SGP 1 com nível de desbalanceamento 10% 41
3.17 SGP 2 com nível de desbalanceamento 5% 42
3.18 SGP 2 com nível de desbalanceamento 10% 43

CAPÍTULO 1

Introdução

Neste capítulo será dada uma introdução à seleção de protótipos. Para melhor entendimento deste assunto, serão apresentados contexto histórico, motivação do uso de seleção de protótipos, e os desafios atuais.

Logo depois da sessão de motivação e contextualização, seguirá um detalhamento deste trabalho, abordando objetivo e estrutura do mesmo.

1.1 Motivação e Contextualização

1.1.1 Histórico

No final dos anos 50, surgiram os primeiros trabalhos de aprendizagem de máquina. De uma forma geral, elas consistiam em dar ao computador a habilidade de reconhecer formas. A partir daí, surgiram diversos problemas onde a aprendizagem de máquina atuava.

Existem três problemas gerais que a aprendizagem de máquina tenta resolver. Um deles é o problema do agrupamento, que consiste em agrupar dados de acordo com suas características, de forma que seja possível extrair informação útil desdes agrupamentos. Um outro problema é a discriminação, que basicamente é achar uma forma de reconhecer um conceito, dado um conjunto de conceitos exemplos. O terceiro e último problema, é o da generalização, que é o problema de como reduzir uma regra, tornando-a mais abrangente e menos custosa.

Reconhecimento de padrões ataca principalmente o problema da discriminação, tendo por objetivo classificar padrões, discriminando-os entre duas ou mais classes. A classificação pode ser feita com padrões pertencentes a qualquer domínio, como reconhecimento de digitais, gestos, escrita, fala, entre outros.

Todo sistema de reconhecimento de padrões utiliza um classificador para discriminar os padrões de teste. O quanto um dado classificador é eficiente é medido pela taxa de acerto média, pela variância, e pela eficiência em termos de custo computacional. Um classificador de aprendizagem baseada em instâncias muito utilizado é o *K-Nearest Neighbor*, KNN [PI69].

O KNN é muito usado por ser um método de aprendizagem supervisionado simples, e por possuir uma taxa de acerto relativamente alta. O conceito básico consiste em: Dado um padrão x a ser classificado e um conjunto de padrões conhecidos T, obter as classes dos K elementos de T mais próximos de x. A classe que obtiver maior ocorrência, ou peso, será a classe de x. Pode-se dizer que o KNN utiliza uma abordagem *"Dize-me com quem andas, e direi quem és."*. O algoritmo esta descrito em Algorithm 1.

Algorithm 1 KNN

Require: K: um número
Require: T: conjunto de treinamento
Require: x: elemento para ser classificado
Require: L: uma lista

1. **for all** $t_i \in T$ **do**
2. $d_i = distance(t_i, x)$
3. adicione $(d_i, Classe(t_i))$ em L
4. **end for**
5. $Ordene(L)$ de acordo com as distâncias
6. obtenha os K primeiros elementos de L
7. **return** a classe de maior ocorrência, ou peso, entre os K

Conforme mostrado em Algorithm 1, o KNN é muito simples, porém, possui um custo alto, pois precisa visitar todos os elementos da base de dados para realizar uma classificação. Assim sendo, é preciso resolver o problema de agrupamento e generalização. Uma das abordagens utilizadas é a seleção de protótipos, detalhada na próxima subsessão.

1.1.2 Seleção de Protótipos

A estratégia do KNN, apesar de eficiente, possui algumas desvantagens. A primeira desvantagem é que o KNN é sensível á ruídos, para baixos valores de K. Outra desvantagem é que o KNN é custoso, pois precisa calcular a distância do padrão que se deseja classificar para cada um dos padrões da base de treinamento, com isso, o KNN torna-se lento em relação a outros classificadores.

Para resolver este problema, surgiu a idéia de utilizar um conjunto menor, gerado a partir da base de dados original (conjunto de treinamento), que represente bem todas as classes, este processo é chamado de seleção de protótipos. A escolha dos protótipos deve ser feita cuidadosamente, pois é necessário que estes elementos possuam uma boa representatividade de todo o conjunto de treinamento. É importante também, que os protótipos não sejam elementos ruidosos, pois isso compromete a taxa de acerto do classificador.

Com os protótipos gerados é possível utilizar o *Nearest Prototype Classification*, NPC, que é utilizar protótipos gerados como treinamento do KNN. Assim a base de dados é reduzida, diminuindo o espaço de armazenamento e o tempo de processamento. Na figura 1.1 percebe-se a redução obtida com o uso de seleção de protótipos.

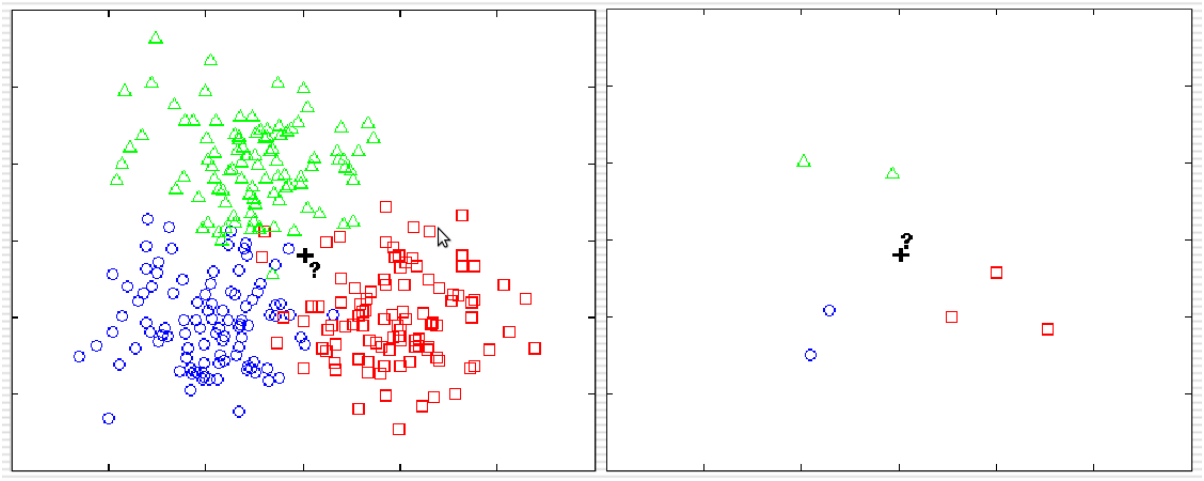

Figura 1.1 Exemplo do NPC

Além de possuir vantagens gerais como a diminuição do espaço de armazenamento e redução de esforço computacional para classificação, a seleção de protótipos pode ainda aumentar o desempenho do classificador. Esta melhora acontece com a eliminação de ruídos e outliers, pois os protótipos aumentam a capacidade de generalização do classificador, levando a maiores taxas de acerto.

Algumas técnicas de seleção de protótipos selecionam instâncias que pertecem ao conjunto de treinamento, ou seja, elas escolhem, dentre as instâncias utilizadas, aquelas que julgam ser mais apropriadas para serem protótipos. Técnicas que utilizam esta abordagem são chamadas de puramente seletivas. Exemplos de técnicas seletivas são *Edited Nearest Neighbor* [CPZ11], *Condensed Nearest Neighbor* [Har68], *Tomek Links* e *One-Sided Selection* [KM97].

Outras técnicas criam novos elementos durante o processo de redução, os protótipos são criados através de combinação entre as instâncias do conjunto de original e ajustes relizados por meio de treinamento supervisionado. Estas técnicas são chamadas de criativas, entre elas estão *Learning Vector Quantization 1, 2.1 e 3* [Koh88] e *Self-Generating Prototypes*[FHA07].

Técnicas de seleção de protótipos também podem ser classificadas como determinísticas ou não deterministicas. Técnicas determinísticas são aquelas que, dada uma base de dados, sempre será gera o mesmo conjunto de protótipos, independente da ordem em que as instâncias de treinamento são apresentadas. Técnicas não determinísticas são aquelas que dependem da ordem das instâncias de treinamento, ou dependem de instâncias pré-selecionadas para ajuste posterior.

Cada uma das técnicas de seleção de protótipos aprensentam características próprias, sendo necessário uma análise do quanto cada uma destas técnicas é apropriada para um dado tipo de base de dados. Algumas técnicas removem instâncias redundantes, outras, removem instâncias que estão na fronteira de classificação, e outras fazem uma combinação das duas abordagens. Detalhes de algumas destas técnicas serão mostrados no próximo capítulo.

1.1.3 Bases Desbalanceadas

Em várias situações do mundo real, os classificadores precisam ser treinados com bases de dados que possue muito mais instâncias de uma de uma classe do que das outras classes, tais bases de dados são chamadas de bases desbalanceadas. Quanto maior a diferença entre a quantidade de instâncias de cada classe, maior o nível de desbalanceamento da base.

Quando treinados com bases de dados desbalanceadas, classificadores sofrem uma redução da performace, e normalmente tendem a classificar mais padrões com as classes marjoritárias. Este é um problema grave, visto que, normalmente, a classificação de instâncias da classe minoritária é que são mais importantes (como exemplo, informações sobre doenças) [HKN07].

Da mesma forma que classificadores podem ser prejudicados por um desbalanceamento, técnicas de seleção de protótipos podem sofrer da mesma forma, selecionando muitas instâncias da classe marjoritária e poucas, ou nenhuma, da classe minoritária.

1.2 Objetivo

O objetivo deste trabalho é expor algumas técnicas de seleção de protótipos e avaliar seu desempenho em bases desbalanceadas. A avaliação de desempenho se refere a taxa de acerto utilizando bases de dados reais, e a disposição dos protótipos por meio de bases artificiais de diferentes níveis de desbalanceamento e sobreposição de classes.

Para que o trabalho seja mais objetivo, apenas os exemplos mais interessante de cada técnica de seleção de protótipos serão enfatizados, citando as maiores vantagens e desvantagens de cada uma delas.

No final do trabalho, será possível identificar quais técnicas são mais apropriadas para bases de dados desbalanceadas, e possivelmente propor adaptações para otimizar algumas delas.

1.3 Estrutura do Trabalho

O restante deste trabalho possui um capítulo com detalhes sobre diferentes técnicas de seleção de protótipos. Durante o capítulo, serão citadas as características e adaptações já conhecidas para tratar de bases desbalanceadas, assim como ilustrações e algoritmos.

Logo após, segue um capítulo mostrando casos de sucesso e falha de cada técnica em bases de dados artificiais, e por fim, um capítulo mostrado os resultados em bases de dados reais com conclusões.

As análises levarão em conta a disposição e a quantidade dos protótipos resultantes de cada técnica. Além disso, será calculada a taxa de acerto dos protótipos em relação ao próprio conjunto de treinamento para analisar a representatividade. Por fim, cada técnica será executada em bases de dados reais, onde será utilizada *K-Fold Cross-Validation* para calcular a taxa de acerto média de cada técnica.

CAPÍTULO 2

Técnicas de Seleção de Protótipos

Neste capítulo, serão explicadas as técnicas de seleção de protótipos analisadas neste trabalho. Cada uma das sessões abaixo aborda conceitos básicos, pseudo-código e características de cada técnica, assim como possíveis adaptações nos seus algoritmos para tratar bases desbalanceadas.

2.1 ENN

Edited Nearest Neighbor Rule[CPZ11] é uma técnica de seleção de protótipos puramente seletiva proposta por Wilson em 1976. De uma forma geral, esta técnica foi projetada para funcionar como um filtro de ruídos, eliminando instâncias na região de fronteira, região de alta susceptibilidade a erros.

Por atuar apenas na região de fronteira, esta técnica possui uma baixa capacidade de redução. Seu algoritmo mantém as instâncias que não estão localizadas nesta região, exceto no caso de instâncias com extrema probabilidade de erro.

O algoritmo do ENN está demonstrado em Algorithm 2.

Algorithm 2 ENN

Require: T: conjunto de treinamento
Require: L: uma lista
1. **for all** instância e_i em T **do**
2. Aplique o KNN sobre e_i utilizando T como treinamento
3. **if** e_i foi classificado erroneamente **then**
4. salve e_i em L
5. **end if**
6. **end for**
7. Remova de T os elementos de L
8. **return** T

O valor de K usado pelo KNN pode variar de acordo com o tamanho da base de dados, porém, tipicamente, utiliza-se K=3. Em geral, o valor de K é inversamente proporcional a quantidade de instâncias que serão eliminadas, ou seja, para que o filtro elimine todos os possíveis ruídos, deve-se utilizar K=1, mas com isso, elimina-se também instâncias não ruidosas.

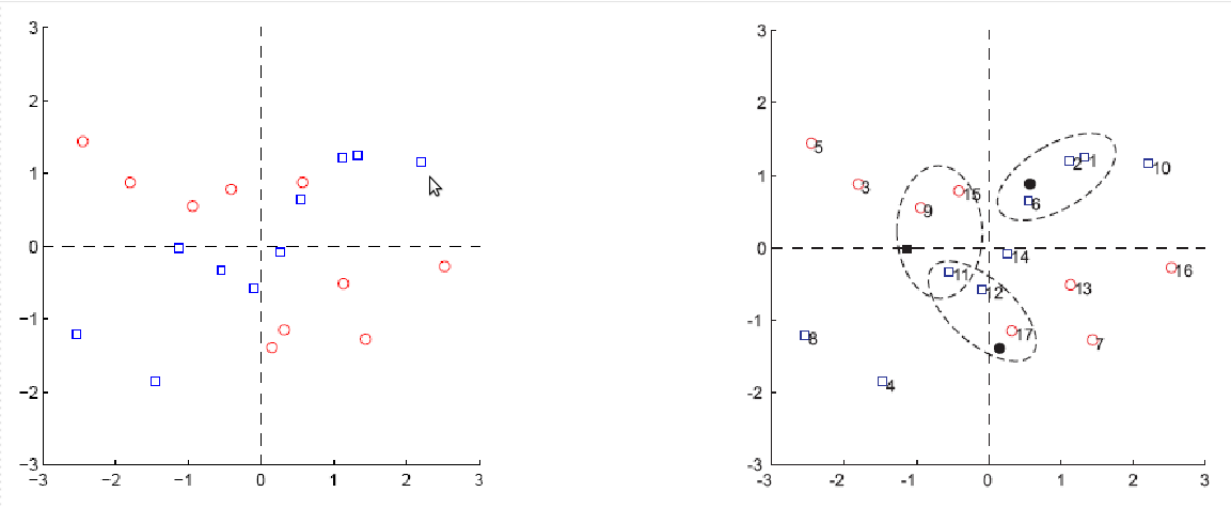

Figura 2.1 ENN aplicado com K=3

Na Figura 2.1 pode-se observar uma base de dados com duas classes. No primeiro gráfico da figura, observar-se a base de dados original, antes da aplicação do ENN. No segundo gráfico, foi aplicado o ENN, Algorithm 2, com K=3 sobre a base de dados. Os pontos pretos representam pontos que foram classificados erroneamente com a aplicação do KNN, a região circulada engloba os K elementos mais próximos do ruído. Mesmo um elemento que será posteriormente eliminado pode ser utilizado para eliminar ou manter outra instância, visto que as remoções são feitas apenas no fim da execução.

O mais interessante do caso acima é que, após a aplicação do ENN, as classes ficaram bem separadas pelos quadrantes pontilhados, mostrando a eficiência do ENN para a base de dados acima.

Uma vantagem do ENN é que ele é determinístico, ou seja, independe da ordem que a base de dados foi apresentada. Com isso, o ENN aplicado a uma base de dados, com o mesmo valor de K, sempre terá o mesmo resultado.

Porém, o ENN também apresenta desvantagens, ele possui uma baixa capacidade de redução, pois elimina apenas ruídos, mantendo instâncias que são desnecessárias, que apresentam apenas redundância de informação. No caso da Figura 2.1, a base poderia ser representada por 4 instâncias bem posicionadas ou, por se tratar de uma técnica seletiva, com 8 instâncias, porém, o ENN manteve 13 instâncias, eliminando apenas 3.

Pelas suas características, normalmente o ENN é utilizado como método de pré-processamento da base de dados, eliminando apenas instâncias que apresentam alta probabilidade de serem ruídos.

No caso de bases desbalanceadas, o ENN pode tratar todas as instâncias da base minoritária como ruídos, caso a base seja altamente desbalanceada (isto será demonstrado posteriormente com exemplos), aumentando ainda mais o nível de desbalanceamento. Uma possível adaptação para o ENN em bases altamente desbalanceadas é eliminar apenas os elementos que sejam da classe marjoritária. O algoritmo adaptado está demonstrado em Algorithm 3.

Algorithm 3 ENN

Require: T: conjunto de treinamento
Require: L: uma lista
1. **for all** instância e_i em T **do**
2. Aplique o KNN sobre e_i usando T como treinamento
3. **if** e_i foi classificado erroneamente **then**
4. **if** e_i for da classe marjoritária **then**
5. salve e_i em L
6. **end if**
7. **end if**
8. **end for**
9. Remova de T os elementos de L
10. **return** T

Com este algoritmo adaptado, as instâncias da classe minoritária seriam mantidas, e a região delimitada por ela ficaria mais bem definida.

2.2 CNN

Condensed Nearest Neighbor[Har68] é uma técnica de seleção de protótipos puramente seletiva que tem como objetivo eliminar informação redundante. Diferentemente do ENN [CPZ11], o CNN não elimina instâncias nas regiões de fronteira, a técnica mantém estes elementos pois considera que estes "são os importantes"para distinguir entre duas classes.

A ideia geral do CNN é encontrar o menor subconjunto da base de dados original que, utilizando o 1-NN, classifica todos os padrões da base corretamente. Fazendo isso, o algoritmo elimina os elementos mais afastados da região de indecisão, da fronteira de classificação.

O algoritmo do CNN está descrito em Algorithm 4.

Algorithm 4 CNN

Require: L: uma lista
Require: T: conjunto de treinamento
1. Escolha um elemento de cada classe $aleatoreamente$ e coloque-os em L
2. **for all** instância e_i de T **do**
3. Aplique o KNN sobre e_i utilizando L como treinamento
4. **if** e_i foi classificado erroneamente **then**
5. salve e_i em L
6. **end if**
7. **end for**
8. **return** L, os protótipos

Pode-se observar que este algoritmo possui uma abordagem totalmente diferente do ENN, pois ele começa com um conjunto mínimo de instâncias (uma de cada classe) e depois adiciona

instâncias conforme a necessidade de mantê-las para que todos os elementos da base de dados original sejam classificados corretamente.

Um ponto importante na descrição do algoritmo, é a palavra *aleatoriamente*, o que significa que o CNN aplicado numa mesma base de dados com um mesmo valor de K para o KNN, nem sempre resulta nos mesmos protótipos. O primeiro fato para que isso ocorra é a seleção aleatória dos protótipos iniciais, a segunda é a ordem em que as instâncias são visitadas pelo algoritmo.

Existem algumas adaptações para o CNN, onde os protótipos iniciais são escolhidos utilizando técnicas como o SGP[FHA07] para obter as instâncias mais centrais. Modificações no CNN são muito comuns [Tom76], porém, mesmo com estas modificações, o CNN ainda não é determinístico, pois a ordem em que as instâncias são classificadas afeta o resultado final.

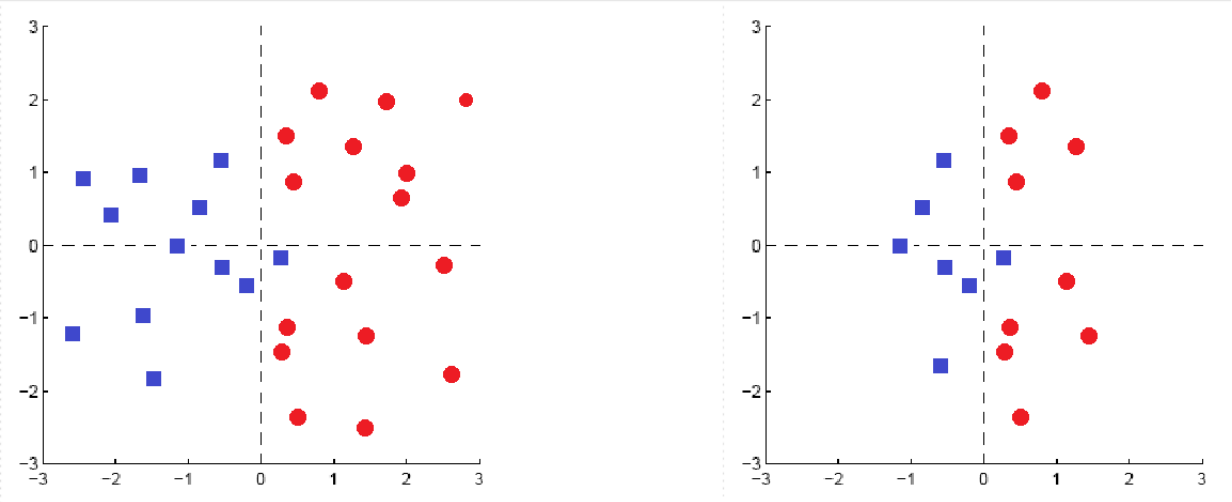

Figura 2.2 Exemplo da aplicação do CNN

No primeiro gráfico da figura 2.2 é mostrado uma base de dados qualquer, no segundo gráfico, é mostrado o resultado do CNN aplicado nesta base. Observa-se que uma grande quantidade de instâncias foi eliminada, mas praticamente todas as instâncias próximas da região de fronteira foram mantidas. Ainda existem instâncias redundantes nesta base, mas como citado anteriormente, isto acontece por conta da escolha aleatória dos protótipos iniciais e da ordem em que as instâncias são visitadas pelo algoritmo.

Para o caso de estudo abordado neste trabalho, o CNN pode ser utilizado de forma adaptada. A adaptação consiste em manter todos os elementos da classe minoritária e o mínimo possível da classe marjoritária. O próprio CNN se encarrega de remover os elementos redundantes da classe marjoritária, assim, basta apenas selecionar todos os elementos da classe minoritária aos protótipos iniciais. Segue em Algorithm 5, o algoritmo desta adaptação:

Algorithm 5 CNN para bases desbalanceadas

Require: L: uma lista
Require: T: um conjunto de treinamento
1. Coloque todos os elementos da classe minoritária em L
2. **for all** instância e_i de T **do**
3. Aplique o KNN sobre e_i utilizando os elementos em L para treinamento
4. **if** e_i foi classificado erroneamente **then**
5. salve e_i em L
6. **end if**
7. **end for**
8. Remova de T os elementos de L
9. **return** T

Com o algoritmo CNN adaptado para bases desbalanceadas, os elementos redundantes da classe marjoritária são removidos, e todos os elementos da classe minoritária são mantidos. Com esta adaptação, além da redução do número de instâncias, também é reduzido o nível de desbalanceamento da base de dados.

2.3 Tomek Links

Mantendo a mesma linha do ENN, *Tomek Links* é uma técnica de seleção de protótipos puramente seletiva que elimina os elementos das regiões de fronteiras e instâncias com probabilidade de serem ruído. Tomek Links podem ser definidos da seguinte forma: Dadas duas instâncias e_i e e_j, o par (e_i, e_j) é chamado de Tomek Link se não existe nenhuma instância e_k, tal que, para todo e_k $dist(e_i,e_j) > dist(e_i,e_k)$, $dist(e_i,e_j) > dist(e_j,e_k)$ e $Classe(e_i) \neq Classe(e_j)$. Segue o algoritmo detalhado em Algorithm 6:

Algorithm 6 Seleciona Tomek Links

Require: L: uma lista
Require: T: um conjunto de treinamento
1. **for all** instância e_i de T **do**
2. e_j = 1-NN de e_i, usando T
3. **if** 1-NN de e_j usando T for e_i **then**
4. **if** $Classe(e_i) \neq Classe(e_j)$ **then**
5. salve o par (e_i, e_j) em L
6. **end if**
7. **end if**
8. **end for**
9. **return** L, Tomek Links

Os Tomek Links representam elementos da região de indecisão e prováveis ruídos, e a técnica de seleção de protótipos consiste em remover os Tomek Links da base de dados original. O algoritmo *Tomek Links* é apresentado em Algorithm 7:

Algorithm 7 Tomek Links

Require: L: uma lista
Require: T: um conjunto de treinamento
1. $L = SelecionaTomekLinks(T)$
2. **for all** (e_i, e_j) em $list$ **do**
3. remova e_i de T
4. remova e_j da T
5. **end for**
6. **return** T, base original filtrada

Enquanto o CNN remove os elementos que estão longe da região de indecisão, o *Tomek Links* remove os elementos que estão próximos desta região, o que causa uma maior separação entre as classes.

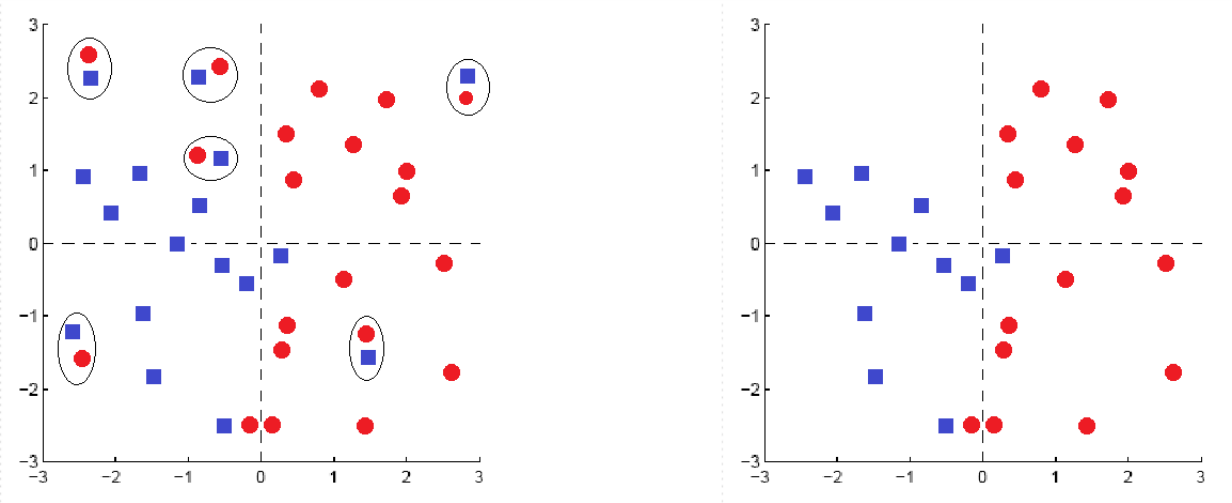

Figura 2.3 Exemplo da aplicação do Tomek Links

Na Figura 2.3 está exemplificada a aplicação do *Tomek Links*. No primeiro gráfico está a base original, e estão circulados os Tomek Links que, no segundo gráfico, foram removidos. No segundo gráfico, a base de dados foi filtrada e a maioria dos ruídos removidos. No exemplo desta figura, o *Tomek Links* fez uma separação entre as classes, eliminando parte de intersecção entre as mesmas.

Uma desvantagem do *Tomek Links* é que esta técnica elimina as duas instâncias presentes no Tomek Link, com isso, instâncias não ruidosas podem estar sendo removidas, instâncias estas que podem representar informação importante para a base de dados.

Observa-se facilmente que o *Tomek Links* pode remover todas as instâncias de regiões de indecisão, inclusive as instâncias da classe minoritária. Sendo assim, uma adaptação do *Tomek Links* é eliminar apenas os elementos das classes marjoritárias. Nesse caso, ainda ocorre uma separação entre as classes, mas as instâncias dos Tomek Links que forem das classes mi-

noritárias são mantidas, diminuindo o nível de desbalanceamento. O algoritmo desta adaptação está demonstrado em Algorithm 8.

Algorithm 8 Tomek Links

Require: L: uma lista
Require: T: um conjunto de treinamento
1. $L = SelecionaTomekLinks(T)$
2. **for all** (e_i, e_j) em L **do**
3. **if** e_i for da classe marjoritária **then**
4. remova e_i de T
5. **end if**
6. **if** e_j for da classe marjoritária **then**
7. remova e_j de T
8. **end if**
9. **end for**
10. **return** T, base original filtrada

Com esta adaptação, a classe minoritária é mantida, evitando o aumento do desbalanceamento ou mesmo a total remoção desta classe por aparentar ser composta apenas de ruídos.

2.4 OSS

One-Sided Selection[KM97] é um método seletivo de seleção de protótipos, surgido pela combinação das técnicas CNN e Tomek Links. O algoritmo consiste na aplicação do CNN e depois da aplicação do Tomek Links como um filtro. O *One-Sided Selection* combina características das duas técnicas. A aplicação do CNN é feita para eliminar instâncias desnecessárias, redundantes, ou seja, instâncias que estão longe da fronteira de classificação. Já a aplicação do Tomek Links tem a função de remover elementos na fronteira de classificação, fazendo uma aparente separação das classes e removendo ruídos.

O OSS é muito utilizado para bases desbalanceadas, utilizando a adaptação do CNN, como monstrado em Algorithm 9.

Algorithm 9 One-Sided Selection

Require: T: um conjunto de treinamento
Require: L: uma lista
1. $L = CNNParaBasesDesbalanceadas(T)$
2. $L = TomekLinksParaBasesDesbalanceadas(L)$
3. **return** L

Observando o algoritmo, é fácil concluir que o *One-Sided Selection* é uma técnica apropriada para bases desbalanceadas. A aplicação do CNN adaptado elimina as instâncias redundantes da base marjoritária, colaborando para, além de diminuir a quantidade de instâncias

longe da fronteira de classificação, diminuir o nível de desbalanceamento entre as classes. Já a aplicação do Tomek Links adaptado, elimina instâncias da classe marjoritária na fronteira de classificação, colaborando para maior delimitação da classe minoritária.

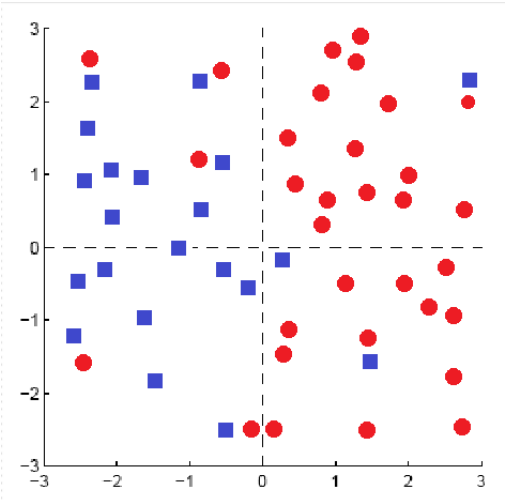

Figura 2.4 Base Original antes da aplicação do OSS

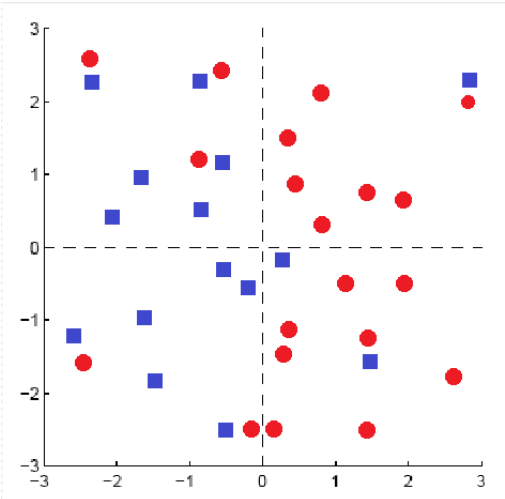

Figura 2.5 Aplicação da primeira etapa do OSS, o CNN

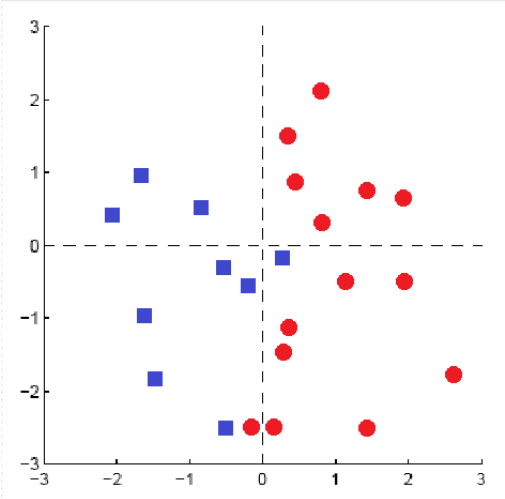

Figura 2.6 Resultado final do OSS, após aplicação da segunda etapa, o Tomek Links

Nas Figuras 2.4, 2.5 e 2.6 observamos que o OSS apresenta resultado satisfatível. No caso das figuras, não havia um desbalanceamento considerável entre as classes, porém, mas ainda assim, o OSS se mostra visualmente eficiente.

Uma desvantagem do *One-Sided Selection* é que ele não é determinístico, o CNN é não-determinístico e como o OSS faz a aplicação dele, torna o mesmo não-determinístico. O OSS poderia ser feito aplicando-se outro algoritmo no lugar do CNN, podendo assim, torna-lo determinístico. Este trabalho, porém, não abordará adaptações para o OSS, pois o mesmo já é apropriado para base de dados desbalanceados, e ser ou não determinístico, apesar de ser levado em consideração, não faz parte do escopo deste trabalho.

2.5 LVQ

Learning Vector Quantization, proposto por Kohonen[Koh88], é um algoritmo supervisionado de síntese de protótipos, ou seja, cria novas instâncias baseadas em instâncias já existentes. A ideia básica do algoritmo é que, dado um conjunto inicial de protótipos, o LVQ faz um ajuste dos destes, de forma a posicionar cada instância em um ponto que seja possível estabelecer uma função discriminante para as classes.

Uma desvantagem do LVQ é que a ordem das instâncias altera o resultado, ou seja, o algoritmo não é deterministico. Outra desvantagem é que, conforme será mostrado, o LVQ possui vários parâmetros, sendo necessário uma análise empírica dos valores apropriados para esses parâmetros.

Os protótipos iniciais podem ser escolhidos de qualquer forma, a idéia é que sejam protótipos que tenham boa representatividade da base de dados, mas também podem ser selecionados aleatoriamente, pois o próprio LVQ se encarrega de fazer os ajustes nestes protótipos.

2.5.1 LVQ 1

LVQ1 é a primeira versão do *Learning Vector Quantization* proposto por Kohonen. O algoritmo do LVQ1 basicamente seleciona alguns protótipos iniciais e ajusta esses protótipos utilizando a base original. Quando uma instância da base original é classificada erroneamente pelos protótipos, afasta-se o protótipo mais próximo, e quando é classificada corretamente, aproxima-se. O algoritmo está detalhado em Algorithm 10.

Algorithm 10 LVQ 1

Require: T: um conjunto de treinamento
Require: P: uma lista para os protótipos
Require: $Selection$: um algorítmo para seleção dos protótipos iniciais
1. $P = Selection(T)$
2. **while** P estiver sub-ajustado **do**
3. $x = ChooseOne(T)$
4. e_i = 1-NN de x sobre P
5. **if** $Classe(e_i) \neq Classe(x)$ **then**
6. $e_i = e_i + \alpha(t) \times [x - e_i]$
7. **else**
8. $e_i = e_i - \alpha(t) \times [x - e_i]$
9. **end if**
10. **end while**
11. **return** P, protótipos ajustados

A vantagem do LVQ1 é que ele estabiliza durante o treinamento, porém, ele possui um grande número de passos. Para a maioria dos problemas, o LVQ 1 possui um resultado satisfatório, mas além da demora, é necessário escolher os parâmetros corretamente.

Um dos parâmetros é o $\alpha(t)$, uma função de ajuste, que serve para aproximar ou afastar os protótipos. Este afastamento ou aproximação é regulado pelo valor de $\alpha(t)$, sendo $0 < \alpha(t) < 1$. Percebe-se que $\alpha(t)$ foi colocado como uma função do tempo. Normalmente, essa função é uma exponencial descrescente, e o algoritmo termina quando $\alpha(t)$ se torna insignificante.

Outra questão do LVQ1 é escolher a quantidade de protótipos iniciais adequada, visto que, esta quantidade não é alterada durante toda a execução do algoritmo.

No caso de bases desbalanceadas, pode-se utilizar fatores de ajustes diferenciados para cada classe, ou escolher uma quantidade aproximada de cada classe para os protótipos iniciais. Fazendo estas adaptações, o LVQ1 poderá ter resultados melhores para bases desbalanceadas.

2.5.2 Optimized-learning-rate LVQ

Optimized-learning-rate LVQ[dSdMSF09] é uma versão otimizada do LVQ1, proposto para aumentar a velocidade de convergência do LVQ1. O modelo consiste basicamente em cada protótipo ter taxas de aprendizado individuais, a dinâmica da taxa de aprendizado consiste no aumento da mesma caso o protótipo esteja classificando corretamente e na diminuição, caso contrário.

$$\alpha(t) = \frac{\alpha(t-1)}{1 + s(t) \times \alpha(t-1)}$$

$$s(t) = +1, \text{ se } x \text{ é classificado corretamente}$$
$$s(t) = -1, \text{ se } x \text{ é classificado erroneamente}$$
$$0 < \alpha(t) < 1$$

Esta alteração do valor de $\alpha(t)$ faz com que o LVQ convirja mais rapidamente, tornando o algoritmo OLVQ mais viável em termos de performace e mantendo as características do LVQ1. Neste trabalho, o OLVQ não será abordado, sendo citado apenas nesta sub-sessão.

2.5.3 LVQ 2.1

Kohonen propos duas outras versões melhoradas do LVQ, uma delas é o LVQ 2.1. Esta nova versão do LVQ faz atualização nos dois protótipos mais próximos desde que as condições de ajuste sejam atendidas.

A ideia do LVQ 2.1 é ajustar apenas os protótipos próximos das fronteiras de classificação, região de indecisão. Para evitar uma divergência entre estes protótipos, foi introduzida a Regra da Janela.

Diz-se que um elemento está na janela quando ele obedece a regra da janela, isso acontece quando um elemento está na região de indecisão.

Dado um elemento x, diz-se que ele está na janela se:

$$\min \frac{d_i}{d_j} \frac{d_j}{d_i} > s, \text{ onde } s = \frac{1-w}{1+w}$$
$$e_i \text{ e } e_j \text{ são os protótipos mais próximos de } x$$
$$d_i \text{ é a distância de } x \text{ para } e_i$$
$$d_j \text{ é a distância de } x \text{ para } e_j$$
$$w \text{ é a largura relativa}$$

Algorithm 11 LVQ 2.1

Require: T: um conjunto de treinamento
Require: P: uma lista para os protótipos
Require: $Selection$: um algorítmo para seleção dos protótipos iniciais

1. $P = LVQ1(T, Selection)$
2. **while** P estiver sub-ajustado **do**
3. $x = ChooseOne(T)$
4. e_i, e_j = 2-NN de x, usando P como treinamento
5. **if** $CaiuNaJanela(x, e_i, e_j)$ **then**
6. **if** $Classe(e_i) \neq Classe(e_j)$ **then**
7. **if** $Classe(e_i) = Classe(x)$ **then**
8. $e_i = e_i + \alpha(t) \times [x - e_i]$
9. $e_j = e_j - \alpha(t) \times [x - e_i]$
10. **else**
11. $e_i = e_i - \alpha(t) \times [x - e_i]$
12. $e_j = e_j + \alpha(t) \times [x - e_i]$
13. **end if**
14. **end if**
15. **end if**
16. **end while**
17. **return** P, protótipos ajustados

A algoritmo de LVQ 2.1 é aplicado depois do LVQ 1, tratando-se de um refinamento dos protótipos já criados. Esta técnica faz ajustes de protótipos apenas se e_i e e_j forem de classes diferentes, e se x obedecer a regra da janela. Pode-se ver o algoritmo detalhado em Algorithm 11.

Enquanto o LVQ1 provoca o afastamento dos protótipos nas regiões de indecisão, o LVQ 2.1 reduz esse afastamento atuando apenas sobre protótipos vizinhos pertencentes a classes diferentes.

Uma desvantagem do LVQ 2.1 é que além do custo ser maior, a aplicação do mesmo pode sobre-ajustar os protótipos nas regiões de indecisão. Para diminuir esse sobre-ajuste, foi criado o LVQ 3, abordado a seguir.

2.5.4 LVQ 3

A segunda melhora proposta por Kohonen foi o LVQ 3. Este método tenta evitar o sobre-ajuste do LVQ 2.1 atuando também quando o elemento já está sendo classificado corretamente pelos dois protótipos mais próximos, aproximando ambos da instância utilizada para ajuste. Além disso, a terceira versão do LVQ introduz um fator de estabização ε. O algoritmo do LVQ 3 está detalhando em Algorithm 12.

Algorithm 12 LVQ 3

Require: T: um conjunto de treinamento
Require: P: uma lista para os protótipos
Require: $Selection$: um algorítmo para seleção dos protótipos iniciais

1. $P = LVQ1(T, Selection)$
2. **while** P estiver sub-ajustado **do**
3. $x = ChooseOne(T)$
4. e_i, e_j = 2-NN de x, usando P como treinamento
5. **if** $CaiuNaJanela(x, e_i, e_j)$ **then**
6. **if** $Classe(e_i) \neq Classe(e_j)$ **then**
7. **if** $Classe(e_i) = Classe(x)$ **then**
8. $e_i = e_i + \alpha(t) \times [x - e_i]$
9. $e_j = e_j - \alpha(t) \times [x - e_i]$
10. **else**
11. $e_i = e_i - \alpha(t) \times [x - e_i]$
12. $e_j = e_j + \alpha(t) \times [x - e_i]$
13. **end if**
14. **else if** $Classe(e_i) = Classe(e_j) = Classe(x)$ **then**
15. $e_i = e_i + \varepsilon \times \alpha(t) \times [x - e_i]$
16. $e_j = e_j + \varepsilon \times \alpha(t) \times [x - e_i]$
17. **end if**
18. **end if**
19. **end while**
20. **return** P, os protótipos ajustados

O fator de estabilização serve para suavizar o ajuste quando os protótipos já estão classificando corretamente x. O valor de ε deve ser tal que $0 < \varepsilon < 1$.

Assim como as outras versões do LVQ, o LVQ 3 é robusto, mas seu maior problema é determinar o valor de tantos parâmetros como α, ε e w. Porém, o maior problema é saber quando os protótipos foram suficientemente ajustados.

2.6 SGP 1

Self-Generating Prototypes [FHA07] é uma técnica de síntese de protótipos muito eficiente no que se trata de redução do número de instâncias. Sua maior vantagem é que, enquanto muitas técnicas de seleção de protótipos dependem da escolha correta do número de protótipos iniciais ou possuem muitos parâmetros, o SGP encontra a quantidade de protótipos e a localização de cada uma em sua fase de treinamento, sem supervisão humana.

A ideia principal do SGP é formar um certo número de grupos e eleger representantes para esses grupos. Conforme necessário, o algorítmo divide os grupos ou move instâncias de um grupo para outro.

Inicialmente, para cada classe, é criado um grupo contendo todas as instâncias daquela classe. Com os grupos feitos, são obtidos os centróides de cada grupo e estes são chamados

representantes do grupo. Depois, para cada grupo, faça os passos abaixo até que não haja mais alterações em nenhum grupo.

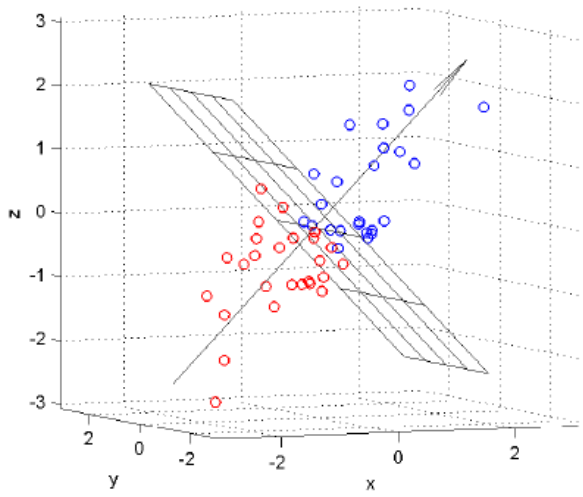

Figura 2.7 Etapa da divisão de grupos [dSPC08].

- Se para todos os padrões de um grupo o protótipo mais próximo é o centróide do grupo, então nenhuma operação é realizada.
- Se para todos os padrões de um grupo o protótipo mais próximo é de uma classe diferente da do grupo, ele é dividido em dois subgrupos Figura 2.7. Essa divisão é feita separando os padrões pelo hiperplano que passa pelo centroide do grupo, e cujo vetor normal é a primeira componente principal gerada pelos padrões do grupo.
- Se para alguns padrões de um grupo o protótipo mais próximo é diferente do centróide, mas da mesma classe, esses padrões são deslocados do grupo original para o grupo do protótipo mais próximo
- Se para alguns padrões de um grupo o protótipo mais próximo não é o centróide e é de uma classe diferente, estes padrões são removidos do grupo original e formam um novo grupo, sendo o centróide computado como um novo protótipo.

No final de cada iteração, o centróide de cada grupo é computado novamente. O processo se repete até que não haja alterações em mais nenhum grupo. Uma descrição mais formal do algoritmo é descrita em Algorithm 13

Algorithm 13 SGP 1

Require: T: um conjunto de treinamento
Require: PS: uma lista para os representantes
Require: GS: uma lista de grupos de instâncias
Require: $TUPLAS$: uma lista de duplas de instâncias

1. **for all** classe C **do**
2. G = $\bigcup$ instâncias da classe C
3. $Adicione(G, GS)$
4. $Adicione(Centroide(G), PS)$
5. **end for**
6. $count = 1$
7. **while** $count \neq 0$ **do**
8. $count$ = $Quantidade(GS)$
9. $Limpe(TUPLAS)$
10. **for all** G em GS **do**
11. P = Representante de G
12. **for all** e_i em $group$ **do**
13. $NearestP$ = $1NN(e_i, PS)$
14. $Adicione((e_i, NearestP), TUPLAS)$
15. **end for**
16. **if** $\forall\ e_i, NearestP$ em $TUPLAS$, $NearestPS$ = P **then**
17. $count$ = $count - 1$
18. **else if** $\forall\ e_i, NearestP$ em $TUPLAS$, a $Classe(NearestP) \neq Classe(P)$ **then**
19. $Vector = PrimeiraComponentePrincipal(G)$
20. $Hiperplano$ = hiperplano que passa pelo centróide de G e cujo vetor normal é a $Vector$.
21. Divida G em 2 grupos, instâncias acima e abaixo de $Hiperplano$
22. Atualize GS e PS.
23. **else if** $\exists\ e_i, NearestP$ em T tal que $NearestP \neq P$ e $Classe(NearestP) = Classe(P)$ **then**
24. Remova e_i de G e adicione a grupo de $NearestP$.
25. Atualize GPS e PS.
26. **else if** $\exists\ e_i, NearestP$ em $TUPLAS$ tal que o $Classe(NearestP) \neq Classe(P)$ **then**
27. Remova e_i de G.
28. Crie um novo grupo contendo as instâncias removidas.
29. Atualize PS e GS,
30. **end if**
31. **end for**
32. **end while**
33. **return** PS

Observando o algoritmo do SGP1, percebe-se que apesar do conceito ser bem simples, esta técnica possui alguns passos complexos, sendo necessário conhecimentos sobre extração de características. No caso, Principal Component Analysis [ref10] é a técnica utilizada para traçar

o vetor perpendicular ao hiperplano Figura 2.7.

A maior vantagem do SGP1 é que, conforme citado anteriormente, ele é não depende de parâmetros como quantidade de protótipos iniciais nem valores específicos. Por ser um algoritmo deterministico, o SGP1 executado numa mesma base de dados, sempre gerará os mesmos protótipos. Estes protótipos gerados possuem excelente representatividade do conjunto de treinamento, tanto que, utilizando-se os protótipos gerados como treinamento de um KNN, e classificando-se todas as instâncias de treinamento do SGP1, a taxa de acerto é de 100%. Apesar de o conjunto de treinamento não ser utilizado para teste, esta é uma forma de medir a representatividade das instâncias geradas por um algoritmo.

Porém, o SGP1 também apresenta desvantagens. Uma delas é que o SGP1 é muito custoso, exigindo, em geral, um treinamento mais longo que outras técnicas.

Outra desvantagem é que o SGP1 é sensível a ruídos, pois, se necessário, o algoritmo criará um grupo com apenas uma instância. No caso de um ruído, isto será muito desvantajoso, considerando que, em experimentos diversos, o SGP1 conseguiu reduzir em mais de 100 vezes o tamanho da base de dados. Com tamanha redução, um ruído passa a ter grande influência na classificação de uma instância.

Apesar das desvantagens citadas, bases desbalanceadas podem ser beneficiadas com o SGP1, considerando que ele considera os agrupamentos de classes, e não a quantidade de instâncias em cada classe.

2.6.1 Fator de Generalização

Para evitar o sobre-ajuste e perda de generalização introduziu-se dois parâmetros ao SGP chamados de R_n e R_s. O R_n representa um limite para o tamanho relativo de um grupo em relação ao maior grupo. Por exemplo, se R_n é 0.1 e o tamanho do maior grupo é 200, todos os grupos com tamanho menor que $20(0.1 \times 200)$ são descartados. O segundo parâmetro R_s é um limiar para a taxa de classificações incorretas de um grupo. Se o número de padrões classificados erroneamentes em um grupo dividido pelo tamanho do grupo é menor que R_s, então o grupo permanece inalterado. Este parâmetros reduzem o número de protótipos e aumentam a capacidade de generalização. Pereira e Cavalcanti recomendam $0.01 < R_n < 0.2$ e $0.01 < R_s < 0.2$ [dSPC08]. Com o fator de generalização, o SGP reduz ainda mais o número de protótipos, mas pode ser inviável em bases desbalanceadas. O fator de desbalanceamento não leva em conta tais bases, assim o SGP pode remover todas as instâncias da base minoritária. Na próxima sessão será descrito o SGP 2, e o algoritmo descrito fará uso do R_n, exemplificando o fator de generalização e mais detalhes sobre vantagens de desvantagens serão abordados.

2.7 SGP 2

O *Self-Generating Prototypes 2*[FHA07] é uma versão melhorada do SGP1 que reduz ainda mais a quantidade de protótipos e é menos sensível aos ruídos e outliers. Para fazer essas melhorias, o SGP2 possui etapas de *merge* e *pruning*.

Quando dois grupos A e B são da mesma classe, e para todas as instâncias de A o segundo protótipo mais próximo é o representante de B, e para todas as instâncias de B o segundo

protótipo mais próximo é o representante de A, os grupos A e B podem ser fundidos, e será computado um novo protótipo para este novo grupo maior. Esta etapa é chamada de *merge*, e ela é responsável por reduzir ainda mais a quantidade de protótipos do SGP1. O *pruning* consiste em remover grupos onde o segundo protótipo mais próximo de todos os padrões de um certo grupo possuem a mesma classe, neste caso, tanto as instâncias quanto o representante do grupo são eliminados.

Algorithm 14 Merge

Require: PS: uma lista para os representantes
Require: GS: uma lista de grupos de instâncias
1. **for all** G em GS **do**
2. **if** $\forall\ e_i$ em G, $2-Nearest(e_i,PS) = P$, $P \in PS$ **then**
3. **if** $\forall\ e_j$ em $Group(P)$, $2-Nearest(e_j,PS) = Prototype(G)$ **then**
4. Remova G de GS e P de PS.
5. **end if**
6. **end if**
7. **end for**
8. **if** Houve merge **then**
9. $Merge(GS,PS)$
10. **end if**

Algorithm 15 Pruning

Require: PS: uma lista para os representantes
Require: GS: uma lista de grupos de instâncias
1. **for all** G em GS **do**
2. **if** $\forall\ e_i$ em G, $Classe(2-Nearest(e_i,PS)) = Classe(G)$ **then**
3. Remova G de GS e P de PS.
4. **end if**
5. **end for**
6. **if** Houve pruning **then**
7. $Pruning(GS,PS)$
8. **end if**

Algorithm 16 SGP 2

Require: T: um conjunto de treinamento
Require: PS: uma lista para os representantes
Require: GS: uma lista de grupos de instâncias
Require: $TUPLAS$: uma lista de duplas de instâncias

for all classe C **do**
  G = $\bigcup$ instâncias de T da classe C
  $Adicione(G, GS)$
  $Adicione(Centroide(G), PS)$
end for
$count = 1$
while $count \neq 0$ **do**
  $count$ = $Quantidade(GS)$
  $Limpe(TUPLAS)$
  for all G em GS **do**
    P = Representante de G
    for all e_i em $group$ **do**
      $NearestP$ = $1NN(e_i, PS)$
      $Adicione((e_i, NearestP), TUPLAS)$
    end for
    if Quantidade de tuplas em $TUPLAS$ onde $NearestP \neq P \leq R_s$ **then**
      $count$ = $count - 1$
    else if $\forall$ $e_i, NearestP$ em T, a $Classe(NearestP) \neq Classe(P)$ **then**
      $Vector = PrimeiraComponentePrincipal(G)$
      $Hiperplano$ = hiperplano que passa pelo centróide de G e cujo vetor normal é a $Vector$.
      Divida G em 2 grupos, instâncias acima e abaixo de $Hiperplano$
      Atualize GS e PS.
    else if $\exists$ $e_i, NearestP$ em T tal que $NearestP \neq P$ e $Classe(NearestP) = Classe(P)$ **then**
      Remova e_i de G e adicione a grupo de $NearestP$.
      Atualize GPS e PS.
    else if $\exists$ $e_i, NearestP$ em T tal que o $Classe(NearestP)! = Classe(P)$ **then**
      Remova e_i de G.
      Crie um novo grupo contendo as instâncias removidas.
      Atualize PS e GS,
    end if
  end for
end while
$LargeGroupCount$ = Quantidade de instâncias do maior grupo em GS
for all G, P in GS, PS **do**
  $CurrentGroupCount$ = Quantidade de instâncias de G.
  if $CurrentGroupCount/LargeGroupCount \leq R_n$ **then**
    Remova G de GS e P de PS.
  end if
end for
$Merge(GS, PS)$
$Pruning(GS, PS)$
return PS

O algoritmo SGP2 16 é mais custoso e demorado que o SGP1, porém, em experimentos práticos, o SGP2 se mostrou mais eficiente, diminuindo a quantidade de protótipos ainda mais que o SGP1 e ainda assim, aumentando a taxa de acerto.

Porém, um defeito do SGP2 é que, assim como o SGP1, o fator de generalização não leva em conta bases desbalanceadas. Dependendo da distribuição das instâncias, pode acontecer de o SGP2 remover o representante da classe minoritária, podendo até remover todos os protótipos desta classe. Isto é perceptível, considerando o valor de $R_n = 0.15$, bases onde a classe minoritária possui apenas 3% das instâncias, mesmo com a base marjoritária multimodal, provavelmente todos os protótipos finais serão da classe marjoritária.

CAPÍTULO 3

Análise em Bases Artificiais

Neste capítulo será analisado o desempenho das técnicas acima em bases desbalanceadas artificiais. Além da taxa de acerto, será considerada a quantidade de protótipos gerados por cada técnica e a distribuição destes protótipos.

3.1 Bases Artificiais

Para avaliar de forma visual o comportamento de cada algoritmo de seleção de protótipos, foram selecionadas bases artificiais com diferentes níveis de sobrebosição entre a classe marjoritária e a classe minotirária. Também foram selecionados dois diferentes níveis de desbalanceamento, um com aproximadamente 10% de desbalanceamento e outro com aproximadamente 5%.

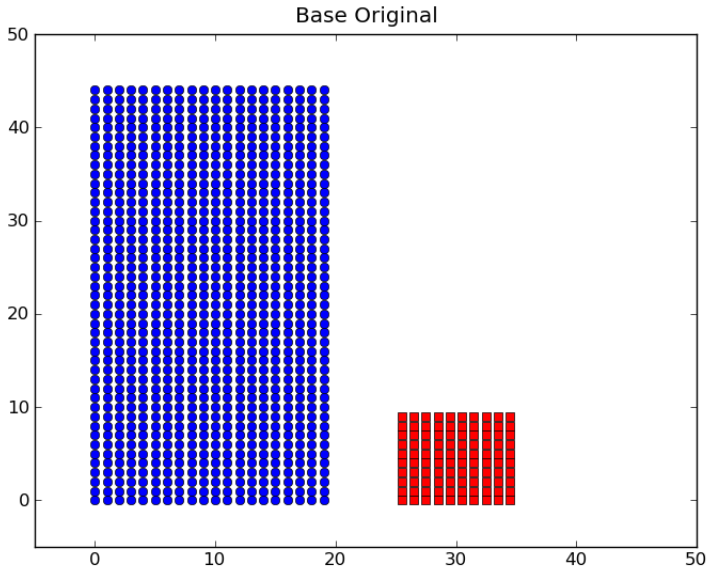

Figura 3.1 Base Original com 10% de desbalanceamento e intersecção mínima

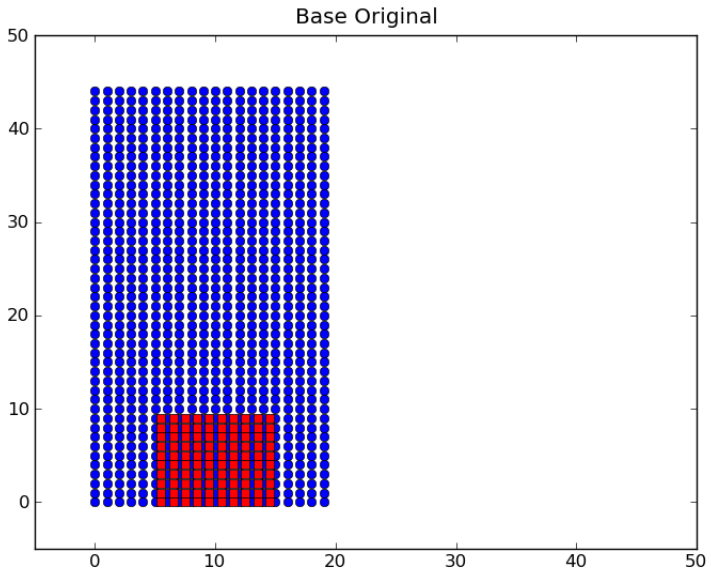

Figura 3.2 Base Original com 10% de desbalanceamento e intersecção máxima

Nas figuras 3.1 e 3.2 estão demonstrados os níveis de intersecção extremos. No primeiro caso, a classe marjoritária está totalmente separada da classe minoritária. No segundo caso, a classe marjoritária engloba toda a classe minoritária. Além destes dois casos, também serão mostrados níveis intermediários de sobreposição. Com estes diferentes níveis de intersecção, será possível avaliar o comportamento de cada técnica diante de diferentes casos de bases desbalanceadas.

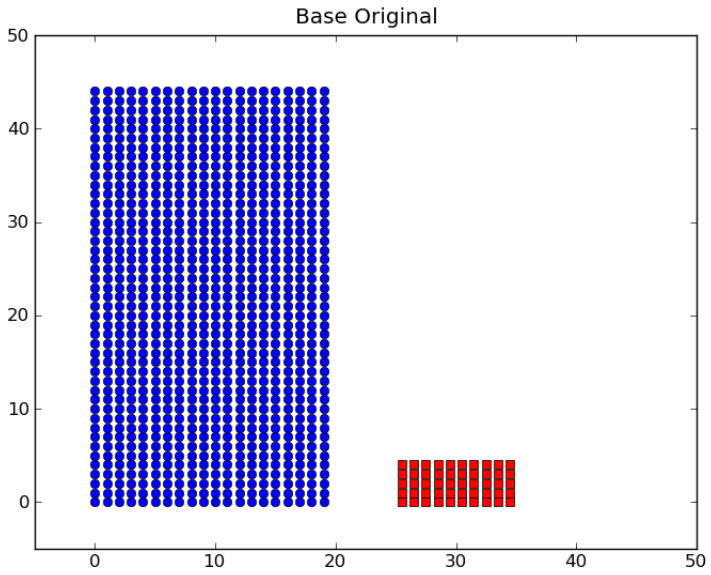

Figura 3.3 Base Original com 5% de desbalanceamento e intersecção mínima

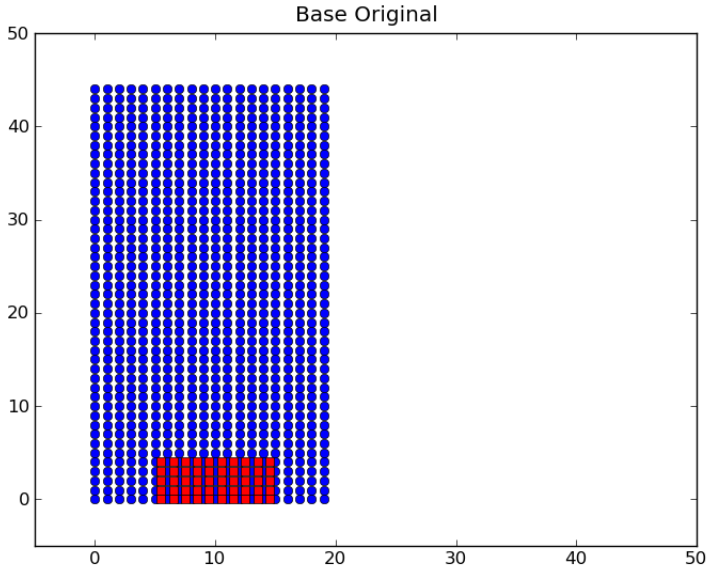

Figura 3.4 Base Original com 5% de desbalanceamento e intersecção máxima

Nas figuras 3.3 e 3.4 observa-se o mesmo caso, sendo que com um nível maior de desbalanceamento. Tendo as mesmas distribuições e intersecções, mas com níveis de desbalanceamentos diferentes, será possível avaliar a relevância dos níveis de desbalanceamento para cada técnica.

3.2 Análise Individual

Nesta sessão, cada técnica sera analisada isoladamente. Para exemplificar e demonstrar as falhas de cada técnica, serão utilizadas figuras e tabelas que comprovem o comportamento de cada algoritmo diante de bases desbalanceadas.

Todas as conclusões poderão ser validadas ou mesmo reavaliadas com a aplicação das técnicas de seleção de protótipos em bases reais no próximo capítulo.

3.2.1 ENN

Conforme dito no capítulo anterior, o ENN é uma técnica que elimina ruídos na fronteira de classificação, mantendo instâncias que apresentam redundância de informação. Na tabela 3.1, podemos ver a taxa de acerto do ENN em uma base com 10%, onde o nível de sobreposição varia gradualmente, sendo o último nível o caso onda a classe minoritária está totalmente imersa dentro da classe marjoritária, lembrando que, foi utilizado o próprio treinamento para teste, afim de avaliar a representatividade dos dados em relação a base original.

Observa-se que a o ENN manteve uma boa taxa de acerto geral, sendo o pior caso com 94.74%. Porém, vemos que esta taxa diminui conforme o nível de sobreposição aumenta, na mesma proporção em que instâncias são eliminadas. Porém, ao observar a taxa de acerto da classe minoritária, pode-se perceber que o ENN eliminou muito mais instâncias da classe minoritária que da marjoritária. Inicialmente o nível de sobreposição das classes é zero, então nenhuma instância é eliminada e a taxa de acerto é mantida. Conforme o nível de sobreposição das classes vai aumentnado, a taxa de acerto da classe minoritária vai decrescendo, chegando até 0% quando a classe minoritária está totalmente sobreposta. Na tabela 3.2 acontece o mesmo, porém, o decrescimento da taxa de acerto da classe minoritária é menor.

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	97.68%	94.74%	94.74%
Acerto marjoritária	100.00%	100.00%	99.00%	98.33%	100.00%
Acerto minoritária	100.00%	100.00%	74.00%	30.00%	0.00%
Tamanho resultante	100.00%	100.00%	95.79%	90.53%	90.00%

Tabela 3.1 ENN com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	95.80%	90.00%	90.00%
Acerto marjoritária	100.00%	100.00%	97.89%	95.56%	100.00%
Acerto minoritária	100.00%	100.00%	77.00%	40.00%	0.00%
Tamanho resultante	100.00%	100.00%	92.00%	82.00%	81.00%

Tabela 3.2 ENN com nível de desbalanceamento 10%

Quando não existe sobreposição de classes, o ENN pode ser uma boa opção para remoção de ruídos, porém, quando existe sobreposição, esta técnica pode remover praticamente todas as instâncias da classe minoritária, aumentando o nível de desbalanceamento e deixando o classificador inviável para identificação da classe minoritária.

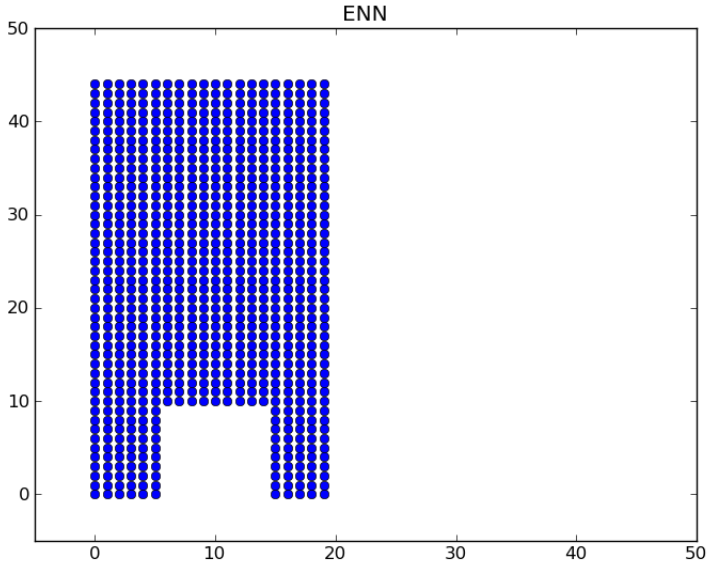

Figura 3.5 ENN sobre sobre base com total sobreposição de classes e 10% de desbalanceamento

A figura 3.5 mostra o que aconteceu após a aplicação do ENN. Todas as instâncias da classe minoritária foram removidas, pois foram tratadas como ruídos. Apesar de instâncias da classe marjoritária também serem removidas, quanto maior o nível de desbalanceamento, mais inviável a técnica se torna, pois conforme a sobreposição aumenta, mais instâncias são removidas, e a classe minoritária tende a desaparecer mais rapidamente que a marjoritária.

Assim, tendo como base este experimento, conclui-se que o ENN não é uma técnica eficiente para se utilizar em bases altamente desbalanceadas, pois, quanto maior o nível de desbalanceamento, mais instâncias da classe minoritária serão eliminadas.

Uma possível adaptação do ENN seria remover instâncias apenas da classe marjoritária, com isso, o nível de desbalanceamento seria diminuido e a taxa de acerto da classe minoritária não seria afetada em relação ao KNN sobre toda a base original.

3.2.2 CNN

O CNN, diferentemente do ENN, possui a abordagem de remover instâncias redundantes, e não ruídos. O comportamento esperado é que o CNN remova muitas instâncias da classe marjoritária e poucas instâncias da classe minoritária. Dependendo do nível de sobreposição das classes, porém, o CNN pode gerar diferentes resultados. Para os experimentos abaixo foi utilizado o valor de K = 3, pois com K = 1 a taxa de acerto seria de 100%.

Observando a tabela 3.3 percebe-se que o CNN obteve uma boa taxa de acerto total, mesmo no caso de bases desbalanceadas. O CNN também teve um alto nível de redução de instâncias. Quando não houve sobreposição, o CNN reduziu a base para 0.32% do tamanho original, aumentando para 7.47% quando com nível máximo de sobreposição.

O resultado da tabela não é uniforme pelo fato de o CNN não ser determinístico, dependendo da ordem em que os dados são apresentados. Porém, pode-se observar que, no geral, o CNN obteve uma boa taxa de acerto para a classe minoritária.

Diminuindo o nível de desbalanceamento (10%), houve uma melhora na taxa de acerto da classe minoritária em todos os níveis de sobreposição. Isso se deve a quantidade de instâncias adicionadas pelo CNN. Com mais instâncias da classe minoritária, a região delimitada por esta classe é maior, com isso, mais instâncias desta classe são mantidas pelo CNN. Pode-se observar isso na tabela 3.4.

Nível de Intersecção	I	II	III	IV	V
Acerto total	94.74%	99.26%	97.47%	94.74%	94.11%
Acerto marjoritária	100.00%	100.00%	98.67%	97.56%	95.89%
Acerto minoritária	0.00%	86.00%	76.00%	44.00%	62.00%
Tamanho resultante	0.32%	0.63%	4.32%	7.68%	7.47%

Tabela 3.3 CNN com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	90.00%	99.30%	94.70%	91.00%	90.40%
Acerto marjoritária	100.00%	99.44%	96.22%	94.44%	95.00%
Acerto minoritária	0.00%	98.00%	81.00%	60.00%	49.00%
Tamanho resultante	0.50%	1.20%	7.50%	13.30%	13.90%

Tabela 3.4 CNN com nível de desbalanceamento 10%

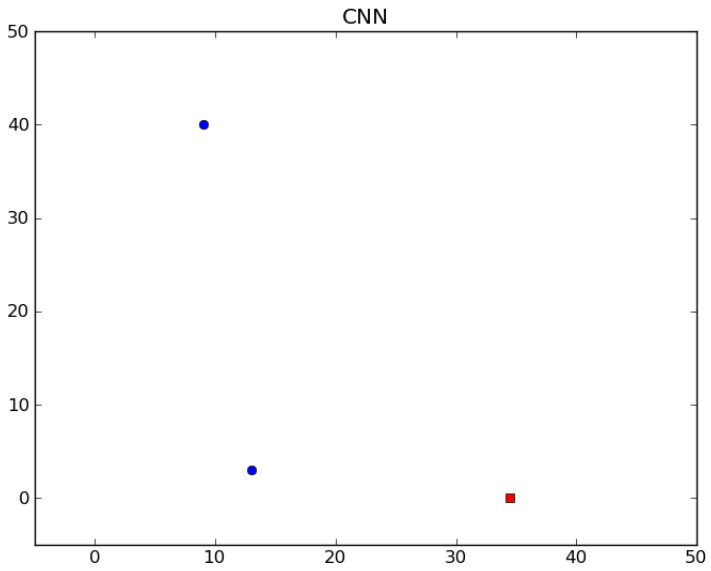

Figura 3.6 CNN sobre sobre base sem sobreposição de classes e 5% de desbalanceamento

Pode-se observar que o CNN reduz muito a base de dados quando não há sobreposição entre as classes, porém, garante que, com K=1, a taxa de acerto sobre o próprio treinamento será de 100%. Neste experimento, porém, a taxa de acerto foi mínima pois foi utilizado K=3, e o CNN deixou apenas 1 instância da classe minoritária, assim, todas as instâncias desta classe foram classificadas erroneamente. O caso pode ser visto na figura 3.6.

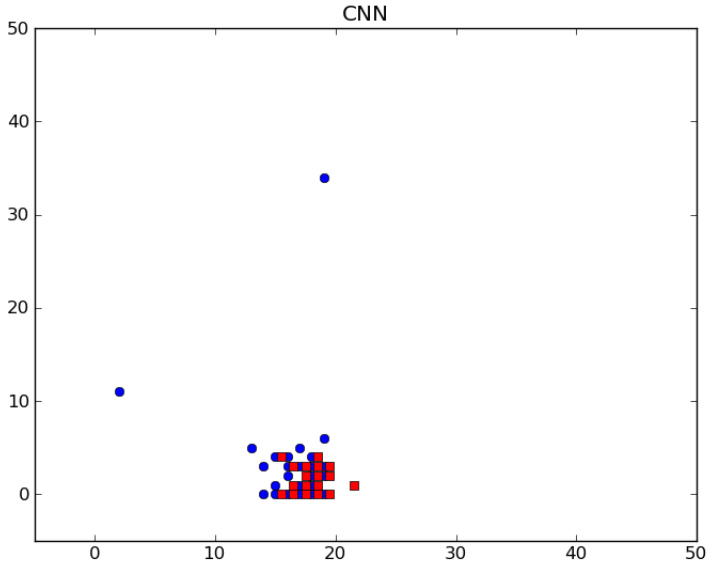

Figura 3.7 CNN sobreposição III de classes e 5% de desbalanceamento

Observando a figura 3.7, também é possível concluir que o CNN remove instâncias redundantes, então, a região de sobreposição é praticamente mantida. Isso mostra que, o CNN colabora para diminuir o nível de desbalanceamento, principalmente diante de classes sobrepostas. Todavia, em casos onde é importante identificar a classe minoritária, é necessário utilizar uma técnica de pré-processamento para melhor delimitar esta classe na região.

3.2.3 Tomek Links

O Tomek Links é uma técnica de seleção de protótipos determinística que atua na região de fronteira. Esta técnica é muito utilizada como pré-processamento de dados, visto que ela remove instâncias que podem ser tratadas como ruídos, mas não tem alto nível de redução de instâncias.

Observando a tabela 3.5, percebe-se que o Tomek Links não reduziu a base de dados quando não há sobreposição de classes, assim, a taxa de acerto é a máxima. Porém, conforme a região de sobreposição aumenta, mais instâncias são removidas, e com isso, a taxa de acerto começa a ser afetada.

Esta técnica possui uma boa taxa de acerto média, mas, conforme o nível de sobreposição aumenta, menor fica a taxa de acerto da classe minoritária. Isso acontece porque com a eliminação de um Tomek Link o nível de desbalanceamento aumenta, dificultando a classficação correta da classe minoritária.

A tabela 3.6 mostra que, com o mesmo nível de desbalanceamento, quando menor o nível de desbalanceamento (quanto mais instâncias da classe minoritária) mais Tomek Links são feitos, e mais instâncias são removidas. Assim, existe uma redução maior da base de dados.

Com um nível de desbalanceamento menor, mais instâncias da classe minoritária são mantidas em regiões onde não há sobreposição.

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	98.42%	97.47%	97.05%
Acerto marjoritária	100.00%	100.00%	99.33%	98.67%	99.00%
Acerto minoritária	100.00%	100.00%	82.00%	76.00%	62.00%
Tamanho resultante	100.00%	100.00%	96.84%	94.32%	94.32%

Tabela 3.5 Tomek Links com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	97.30%	94.60%	93.90%
Acerto marjoritária	100.00%	100.00%	98.56%	97.00%	97.56%
Acerto minoritária	100.00%	100.00%	86.00%	73.00%	61.00%
Tamanho resultante	100.00%	100.00%	94.60%	89.60%	89.60%

Tabela 3.6 Tomek Links com nível de desbalanceamento 10%

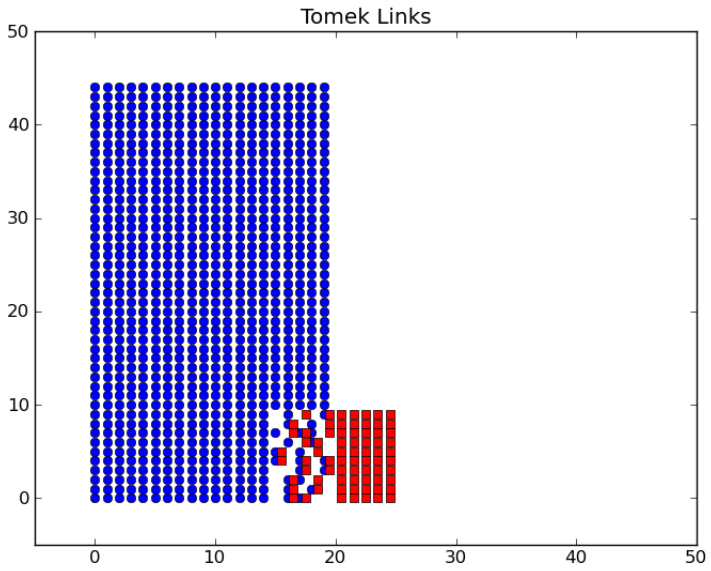

Figura 3.8 Tomek Links sobreposição III de classes e 10% de desbalanceamento

Na figura 3.8, está mostrado o resultado do *Tomek Links* aplicado numa base de dados com 10% da classe minoritária e 90% da classe marjoritária. O nível de sobreposição é o nível III, ou seja, a classe minoritária está parcialmente imersa na classe marjoritária. Observando a figura, percebe-se que o *Tomek Links* remove instâncias da região de sobreposição, e mantém instâncias redundantes.

Com isso, conclui-se que o Tomek Links deve ser utilizado como uma técnica de pré-processamento, pois ainda existe muitos dados redundantes, principalmente da classe marjoritária.

Uma opção é utilizar o Tomek Links alternativo apresentado em Algorithm 8, pois com isso, o nível de desbalanceamento seria reduzido. Outra vantagem é que a região de sobreposição seria classificada como da classe minoritária, caso de interesse na maioria das bases de dados.

3.2.4 OSS

Conforme dito anteriormente, o *One-Sided Selection* é uma técnica muito utilizada como pré-processamento e é apropriada para bases desbalanceadas. Inicialmente, com a aplicação do CNN adaptado para bases desbalanceadas, instâncias redundantes da classe marjoritária são removidas. Depois, com a aplicação do Tomek Links adaptado para bases desbalanceadas, as instâncias da classe marjoritária que estão na fronteira são removidas.

O OSS mantém as instâncias da classe minoritária, e com isso ele favorece a identificação desta classe. Observando a tabela 3.7, observa-se que a taxa de acerto total é alta, tendo 93.89% no pior caso, e isso independe do nível de intersecção. Para a classe minoritária, vê-se que a taxa de acerto foi alta, especialmente quando existe um baixo nível de sobreposição de classes.

O poder de redução do OSS também é alto, indo para 9.68% da base original no pior caso de redução. Neste mesmo caso, o nível de desbalanceamento é reduzido, chegando a ter

praticamente a mesma quantidade de instâncias em cada classe.

Observando a tabela 3.8, percebe-se que com a diminuição do nível de desbalanceamento para 10% não houve ganho de desepenho nem de redução. Com isso, conclui-se que o OSS é uma técnica eficiente independentemente do nível de desbalanceamento da base.

Nível de Intersecção	I	II	III	IV	V
Acerto total	98.74%	99.58%	97.05%	93.37%	93.89%
Acerto marjoritária	98.67%	99.56%	97.11%	93.44%	94.22%
Acerto minoritária	100.00%	100.00%	96.00%	92.00%	88.00%
Tamanho resultante	5.58%	5.89%	7.16%	8.63%	9.68%

Tabela 3.7 OSS com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	97.50%	98.80%	95.20%	85.90%	87.50%
Acerto marjoritária	97.22%	98.67%	95.22%	85.78%	88.00%
Acerto minoritária	100.00%	100.00%	95.00%	87.00%	83.00%
Tamanho resultante	10.40%	11.10%	13.60%	15.50%	17.40%

Tabela 3.8 OSS com nível de desbalanceamento 10%

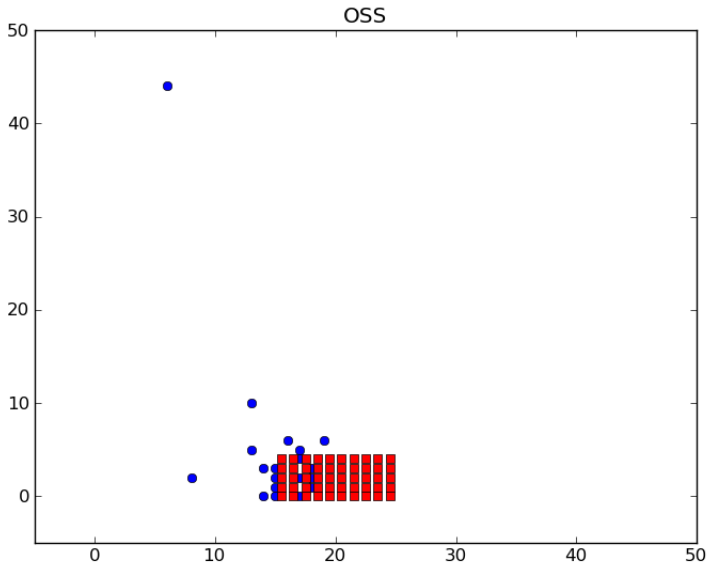

Figura 3.9 OSS com sobreposição III de classes e 5% de desbalanceamento

A figura 3.9 exemplifica a aplicação do *One-Sided Selection* sobre uma base de dados onde a classe minoritária possui apenas 5% das instâncias e metade das instâncias desta classe estão

na região de indecisão. Observa-se que as instâncias da classe marjoritária redudantes foram eliminadas, o que aumenta a capacidade de generalização dos protótipos. Outro detalhe perceptível é que a região de indecisão é tomada pela classe minoritária.

Por estas caracterísitcas, o OSS é uma técnica muito apropriada para bases desbalanceadas, além de ser eficiente no que se refere a redução de instâncias. Porém, uma desvantagem é que a classe minoritária fica com instâncias redudantes após a aplicação, por isso, o OSS é muito utilizado como técnica de pré-processamento de dados para outras técnicas de seleção de protótipos.

3.2.5 LVQ

O *Learning Vector Quantization* é uma técnica seletiva de seleção de protótipos. Diferentemente das técnicas examinadas nas subsessões anteriores, esta é uma técnica seletiva, ou seja, ela cria novas instâncias a partir do conjunto de treinamento.

Para os experimentos das versões do LVQ, foram utilizados 1000 iterações, $\alpha(t) = 0.02 \times e^{(-t/200)}$, sendo t o número da iteração, $w = 0.70$ e $\varepsilon = 0.30$. Os protótipos iniciais utilizados foram 5 protótipos para cada classe, sendo eles próximos da mediana da classe correspondente.

3.2.5.1 LVQ 1

O LVQ 1 obteve uma boa taxa de acerto geral, sendo sempre acima de 77.68%. Observa-se também que o taxa de acerto tende a decrescer conforme o nível de sobreposição de classes aumenta. observando a tabela 3.9, percebe-se que a taxa de acerto total decresce conforme o nível de sobreposição aumenta por conta da classe marjoritária, enquanto a classe minoritária se manteve com alto nível de acerto (neste experimento 100%).

A taxa de acerto da classe minoritária é alta porque as instâncias desta classe estão muito próximas, então, durante o ajuste de protótipos, os mesmos ficam bem ajustados, em posições estratégicas.

A figura 3.10 mostra que, em baixo nível de sobreposição, os protótipos ficam em torno do centróide, e o seu espalhamento depende totalmente do nível de espalhamento das classes. Já a figura 3.11 mostra que o mesmo acontece, porém, o LVQ 1 tende a dividir as classes. Neste caso, os protótipos da classe marjoritária ficaram mais espalhados, porém, afastados da região de indecisão, por isso, a taxa de acerto da classe minoritária é alta.

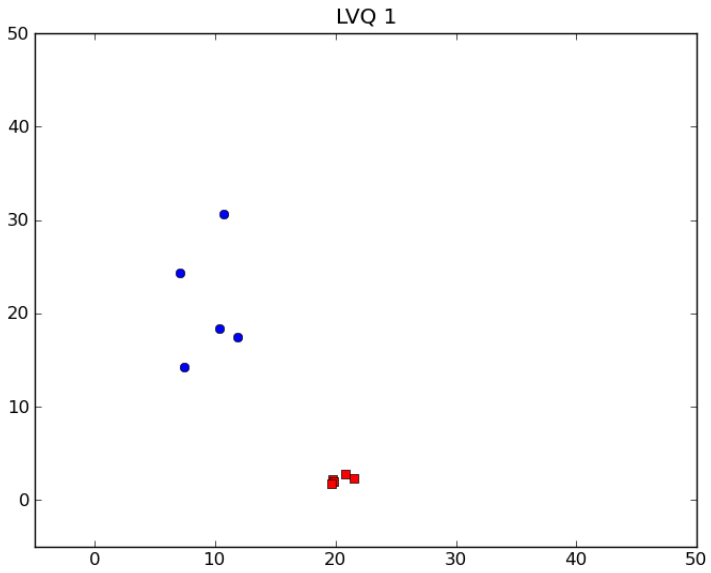

Figura 3.10 LVQ 1 com sobreposição de classes III e 5% de desbalanceamento

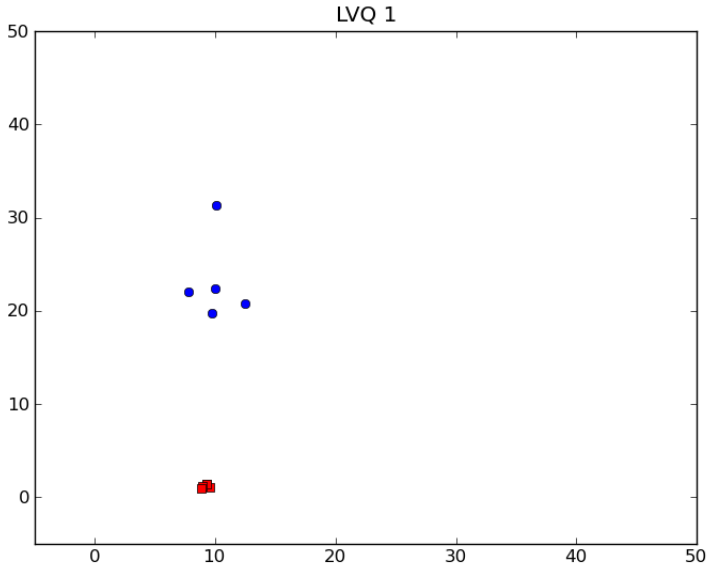

Figura 3.11 LVQ 1 com sobreposição de classes V e 5% de desbalanceamento

Nível de Intersecção	I	II	III	IV	V
Acerto total	93.47%	91.79%	84.84%	74.74%	77.68%
Acerto marjoritária	93.11%	91.33%	84.00%	73.33%	76.44%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.05%	1.05%	1.05%	1.05%	1.05%

Tabela 3.9 LVQ1 com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	95.30%	87.40%	80.80%	77.00%	71.90%
Acerto marjoritária	94.78%	86.00%	78.67%	74.44%	68.78%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.00%	1.00%	1.00%	1.00%	1.00%

Tabela 3.10 LVQ 1 com nível de desbalanceamento 10%

Na tabela 3.10, a classe minoritária é composta de 10% das instâncias, mas ainda assim, as características se mantiveram as mesmas, a taxa de acerto da classe minoritária permaneceu alta, enquanto a taxa da classe marjoritária descresceu conforme o nível de sobreposição aumentou. Porém, a figura 3.12 mostra que, conforme a quantidade de instâncias da classe minoritária aumenta, mais espalhados ficam os protótipos desta classe, ocasionando numa maior delimitação da região da mesma.

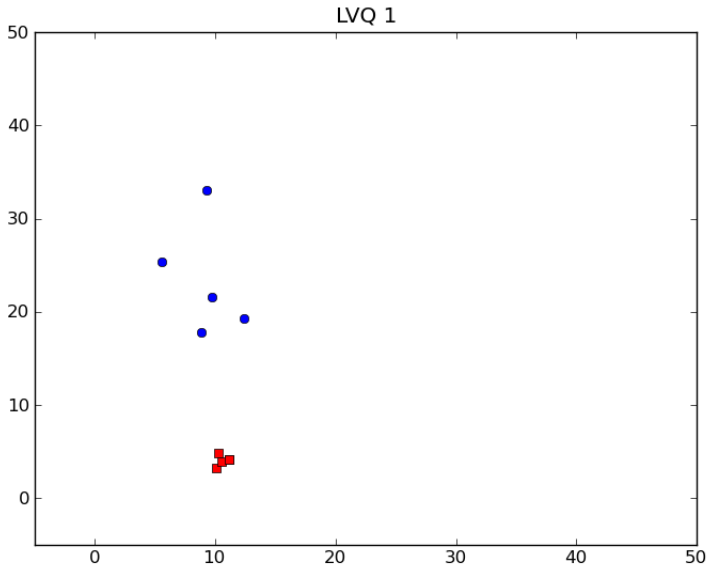

Figura 3.12 LVQ 1 com sobreposição de classes V e 10% de desbalanceamento

O LVQ1 se mostra eficiente para bases desbalanceadas, principalmente no que se refere a classificar bem instâncias da classe minoritária. Uma grande vantege é que a quantidade de protótipos resultante pode ser pré-definida, com isso, o desbalanceamento pode ser reduzido ou até mesmo eliminado. É recomendado também que a quantidade de protótipos seja proporcional ao nível de espalhamento de cada classe, quanto mais espalhada, mais protótipos são necessários para definir a classe.

3.2.5.2 LVQ 2.1

O LVQ 2.1 também teve uma boa taxa de acerto média, e assim como o LVQ 1, teve taxa de acerto máxima para classe minoritária em todos os níveis de sobreposição de classes. Esses dados podem ser observados na tabela 3.11 e 3.12.

Nível de Intersecção	I	II	III	IV	V
Acerto total	90.84%	89.68%	83.89%	80.11%	76.84%
Acerto marjoritária	90.33%	89.11%	83.00%	79.00%	75.56%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.05%	1.05%	1.05%	1.05%	1.05%

Tabela 3.11 LVQ 2.1 com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	95.80%	90.40%	81.60%	80.70%	76.20%
Acerto marjoritária	95.33%	89.33%	79.56%	78.56%	73.56%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.00%	1.00%	1.00%	1.00%	1.00%

Tabela 3.12 LVQ 2.1 com nível de desbalanceamento 10%

Conforme citado anteriormente, o LVQ 2.1 evita uma divergência entre os protótipos, isto pode ser observado em leves diferenças entre a figura 3.13 e a figura 3.14.

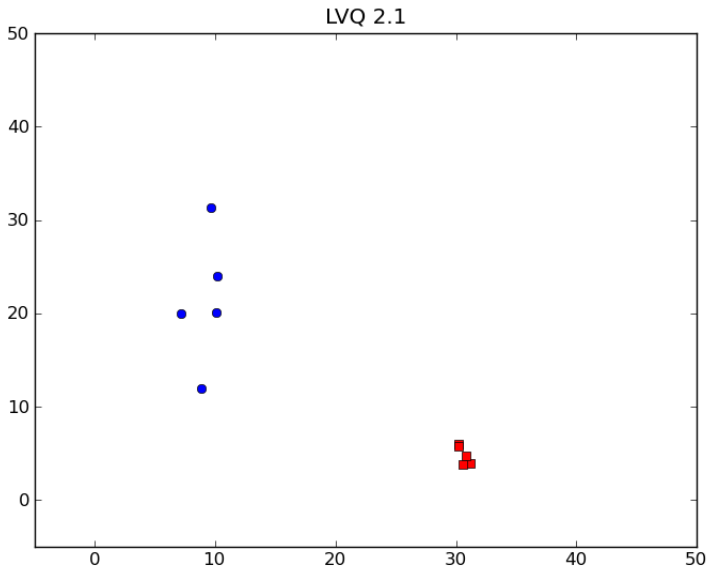

Figura 3.13 LVQ 2.1 sem sobreposição de classes e 10% de desbalanceamento

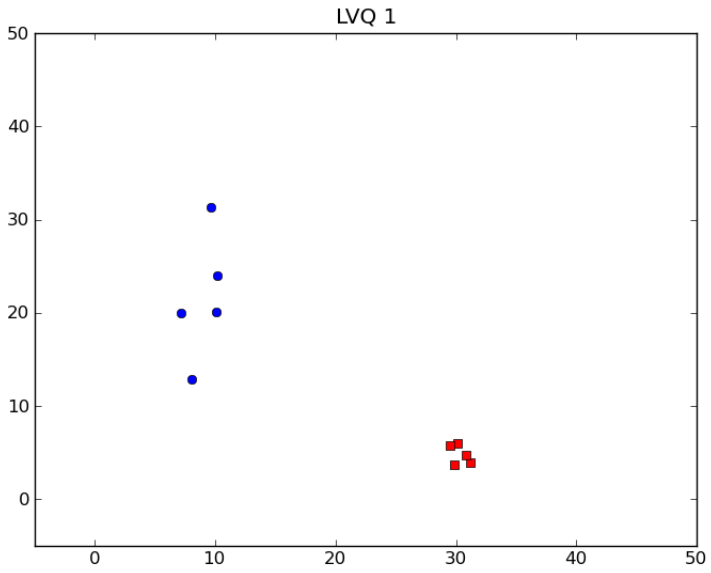

Figura 3.14 LVQ 1 sem sobreposição de classes e 10% de desbalanceamento

Observa-se que os protótipos do LVQ 2.1 estão levemente menos espalhados que o LVQ 1, isso afeta a taxa de acerto da classe marjoritária (comparar tabela 3.10 e 3.12). Assim, é interessante utilizar o LVQ 2.1 quando se deseja um menor espalhamento das instâncias, normalmente, quando existe uma distribuição uniforme das classes.

3.2.5.3 LVQ 3

O LVQ 3 tem o objetivo de evitar o sobreajuste do LVQ 2.1. Observando as tabelas 3.13 e 3.14, percebe-se que o LVQ 3 manteve em média o mesmo desempenho que o LVQ 2.1, porém, a diferença de desempenho está em bases de baixo nível de sobreposição entre as classes.

Nível de Intersecção	I	II	III	IV	V
Acerto total	95.79%	92.32%	83.68%	78.32%	74.53%
Acerto marjoritária	95.56%	91.89%	82.78%	77.11%	73.11%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.05%	1.05%	1.05%	1.05%	1.05%

Tabela 3.13 LVQ 3 com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	94.70%	88.10%	89.40%	75.60%	73.50%
Acerto marjoritária	94.11%	86.78%	88.22%	72.89%	70.56%
Acerto minoritária	100.00%	100.00%	100.00%	100.00%	100.00%
Tamanho resultante	1.00%	1.00%	1.00%	1.00%	1.00%

Tabela 3.14 LVQ 3 com nível de desbalanceamento 10%

Comparando a figura 3.15 com a figura 3.13 e a 3.14, observa-se que o LVQ 3 é o meio termo entre o LVQ 1 e o LVQ 2.1. Assim, recomenda-se o uso do LVQ 3 em bases desbalanceadas em casos onde o nível de sobreposição de classes é baixo e deseja-se conter o nível de espalhamento entre as classes.

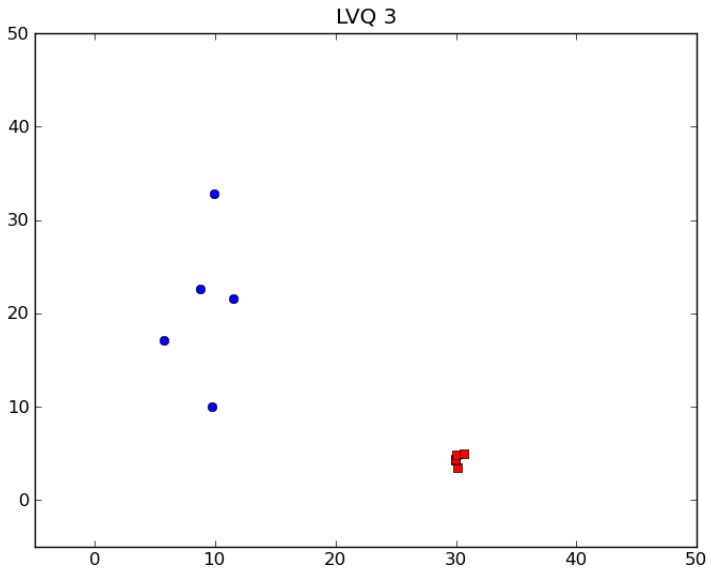

Figura 3.15 LVQ 3 sem sobreposição de classes e 10% de desbalanceamento

A escolhe da versão do *Learning Vector Quantization* é tão importante quanto a escolha dos parâmetros de forma apropriada. Nestes experimentos foram utilizados os mesmos valores para parâmetros em comum, então, as conclusões se restrigem ao uso de cada uma em bases desbalanceadas, mas, experimentalmente, verifica-se que a escolha adequada de parâmetros é tão importante quanto a escolha da versão do LVQ.

3.2.6 SGP 1

A primeira versão do *Self-Generating Prototypes* é uma técnica de alto poder de redução muito completa para bases com distribuições multi-modais de classes. Porém, após os experimentos realizados, foi possível identificar uma série de falhas nesta técnica.

Conforme citado no capítulo anterior, o SGP possui um fator de generalização. Neste experimento, foi utilizado $R_n = 0.05$ e $R_s = 0.05$, a escolha de valores foi feita de acordo com o proposto em trabalhos relacionados[dSPC08].

O poder de redução desta técnica é muito alto, observa-se na tabela 3.15 que, mesmo em alto nível de sobreposição, a quantidade de protótipos não pasou de 5.68% da base original no caso de alto nível de desbalanceamento.

O SGP, sem fator de generalização, apresenta uma taxa de 100% de acerto sobre o conjunto de treinamento[FHA07]. Porém, ao introduzir o fator de generalização o SGP perde esta propriedade. Observando a tabela 3.15, percebe-se que, mesmo com fator de generalização, o SGP 1 obteve boas taxas de acerto para baixo nível de sobreposição. Porém, para um alto nível de sobreposição (níveis IV e V), a classe minoritária obteve 0.00% de taxa de acerto, enquanto que a classe marjoritária obteve 100%. O mesmo acontece na tabela 3.16, porém, este comportamento ocorreu apenas no nível máximo de sobreposição.

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	97.68%	94.74%	94.74%
Acerto marjoritária	100.00%	100.00%	98.89%	100.00%	100.00%
Acerto minoritária	100.00%	100.00%	76.00%	0.00%	0.00%
Tamanho resultante	0.53%	2.74%	5.26%	5.16%	5.68%

Tabela 3.15 SGP 1 com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	95.40%	90.70%	90.00%
Acerto marjoritária	100.00%	100.00%	97.89%	95.78%	100.00%
Acerto minoritária	100.00%	100.00%	73.00%	45.00%	0.00%
Tamanho resultante	0.40%	3.50%	5.80%	8.20%	5.90%

Tabela 3.16 SGP 1 com nível de desbalanceamento 10%

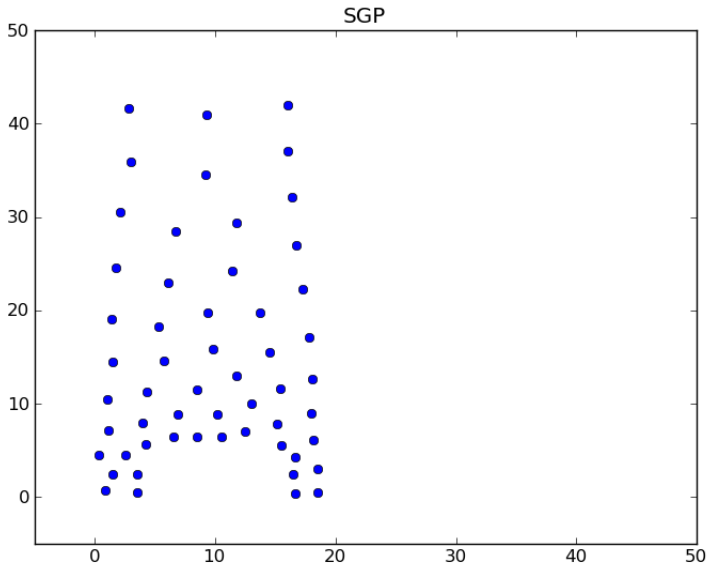

Figura 3.16 SGP 1 com total sobreposição de classes e 5% de desbalanceamento

Observando a figura 3.16, conclui-se que o fator de generalização fez com que todas as instâncias da classe minoritária fossem eliminadas. Em baixo nível de sobreposição isso não acontece pois as instâncias da classe minoritária são representadas por um mesmo protótipo, porém, conforme o nível de sobreposição aumenta, mais protótipos são necessários para representar esta classe, assim, menos instâncias por protótipos. Quando o algoritmo principal termina, o fator de generalização elimina os grupos que possuem menos que $R_s \times S$, sendo S a

quantidade de instâncias do maior grupo. Assim, as instâncias vão sendo removidas independentemente de classe.

Observa-se também que, quando o nível de desbalanceamento é mais acentuado, este efeito ocorre mais rapidamente, pois são menos instâncias para dividir entre os protótipos necessários. Assim, em bases desbalanceadas onde a classe minoritária representa 1%, por exemplo, provavelmente todas as instâncias destas classes seriam eliminadas, independente do nível de sobreposição. Caso não seja utilizado o fator de generalização, estas instâncias são mantidas, porém, o SGP 1 manterá todos os ruídos, levando a erros de classificação.

Esta é uma falha muito grave, porque apesar de ter uma alta taxa de acerto geral e capacidade de redução, o SGP possui péssimas taxas de acerto para a classe minoritária, chegando até a errar em todos os casos.

Uma solução para o SGP é que o fator de generalização seja mantido, porém, com uma alteração. o R_n eliminaria grupos com poucas instâncias dependendo, não da quantidade de instâncias do maior grupo, mas sim, da quantidade de instâncias do maior grupo da mesma classe. Com esta alteração, a classe minoritária não seria extinta e a taxa de acerto da mesma seria beneficiada.

3.2.7 SGP 2

A segunda versão do *Self-Generating Prototypes* apresenta os mesmos defeitos da primeira versão. Porém, ela apresenta algumas caracterísitcas interessantes que serão abordadas.

Observando as tabelas 3.17 e 3.18, percebe-se que o SGP 2 obteve uma excelente taxa de acerto geral, porém, a taxa de acerto da classe minoritária é altamente prejudicada no momento da generalização. As possíveis adaptações para este problema já foram citados na sub-sessão anterior.

Quanto a quantidade de protótipos, observase que o SGP 2 tem um poder de redução muito maior que o SGP 1, isso acontece devido ao acrescimo do *Pruning* e *Merge*. Em geral o SGP 2 obteve uma taxa de acerto semelhante ao SGP 1, porém, em geral apenas metade dos protótipos foram necessários.

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	97.58%	94.74%	94.74%
Acerto marjoritária	100.00%	100.00%	98.78%	100.00%	100.00%
Acerto minoritária	100.00%	100.00%	76.00%	0.00%	0.00%
Tamanho resultante	0.21%	1.26%	2.42%	2.63%	2.63%

Tabela 3.17 SGP 2 com nível de desbalanceamento 5%

Nível de Intersecção	I	II	III	IV	V
Acerto total	100.00%	100.00%	95.50%	90.00%	90.20%
Acerto marjoritária	100.00%	100.00%	98.22%	100.00%	95.56%
Acerto minoritária	100.00%	100.00%	71.00%	0.00%	42.00%
Tamanho resultante	0.30%	1.90%	2.90%	2.70%	4.40%

Tabela 3.18 SGP 2 com nível de desbalanceamento 10%

Comparando-se a figura 3.17 com a figura 3.16, observa-se que o SGP 2 manteve a mesma região representada com uma quantidade menor de instâncias. O *Merge* é um algoritmo determinístico e muito eficiente, já o *Pruning* não é determinístico, e é necessário um limite de poda para que não aconteça um sobreajuste.

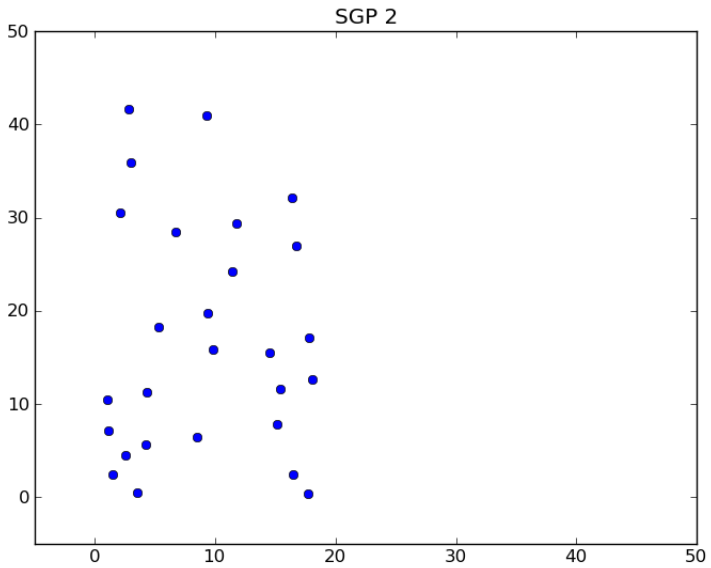

Figura 3.17 SGP 2 com total sobreposição de classes e 5% de desbalanceamento

Conclui-se que o SGP 2 é superior ao SGP 1 no que se refere a generalização e redução de protótipos. Porém, esta técnica apresenta os mesmos defeitos do SGP 1, sendo necessária a mesma adaptação citada anteriormente para que a classe minoritária, comumente o caso de maior interesse, possa ser identificada.

Referências Bibliográficas

[CPZ11] Ruiqin Chang, Zheng Pei, and Chao Zhang. A modified editing k-nearest neighbor rule. *JCP*, 6(7):1493–1500, 2011.

[dSdMSF09] Renata M. C. R. de Souza and Telmo de M. Silva Filho. Optimized learning vector quantization classifier with an adaptive euclidean distance. In Cesare Alippi, Marios M. Polycarpou, Christos Panayiotou, and Georgios Ellinas, editors, *ICANN (1)*, volume 5768 of *Lecture Notes in Computer Science*, pages 799–806. Springer, 2009.

[dSPC08] Cristiano de Santana Pereira and George D. C. Cavalcanti. Prototype selection: Combining self-generating prototypes and gaussian mixtures for pattern classification. In *IJCNN*, pages 3505–3510. IEEE, 2008.

[EJJ04] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[FdJH09] Alberto Fernández, María José del Jesús, and Francisco Herrera. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *Int. J. Approx. Reasoning*, 50(3):561–577, 2009.

[FHA07] Hatem A. Fayed, Sherif Hashem, and Amir F. Atiya. Self-generating prototypes for pattern classification. *Pattern Recognition*, 40(5):1498–1509, 2007.

[Har68] P. E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[HKN07] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In Zoubin Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 935–942. ACM, 2007.

[KM97] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In Douglas H. Fisher, editor, *ICML*, pages 179–186. Morgan Kaufmann, 1997.

[Koh86] Teuvo Kohonen. Learning vector quantization for pattern recognition. Report TKK-F-A601, Laboratory of Computer and Information Science, Department of Technical Physics, Helsinki University of Technology, Helsinki, Finland, 1986.

[Koh88] Teuvo Kohonen. Learning vector quantization. *Neural Networks*, 1, Supplement 1:3–16, 1988.

[PI69] E. A. Patrick and F. P. Fischer II. A generalization of the k-nearest neighbor rule. In *IJCAI*, pages 63–64, 1969.

[ref10] Principal component analysis. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, page 795. Springer, 2010.

[Sav] Sergei Savchenko. http://cgm.cs.mcgill.ca/ .

[TC67] P.E. Hart T.M. Cover. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13:21 – 27, 1967.

[Tom76] I. Tomek. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(2):679–772, 1976.

[WP01] G. Weiss and F. Provost. The effect of class distribution on classifier learning, 2001.