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

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Técnicas de Seleção de Protótipos

1.1 Motivação e Contextualização

Classificadores são (...)

> *"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web, the content, links, and transactions between people and computers. A **Semantic Web** which should make this possible has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The **intelligent agents** people have touted for ages will finally materialize."Tim Berners-Lee*

Tradução literal: *"Eu tenho um sonho para a Web [em que os computadores] tornam-se capazes de analisar todos os dados na Web, o conteúdo, links, e as transações entre pessoas e computadores. A **Web Semântica** que deve tornar isso possível ainda está para surgir, mas quando isso acontecer, os mecanismos dia-a-dia da burocracia do comércio e nossas vidas diárias serão tratados por máquinas falando com máquinas. Os **agentes inteligentes** que as pessoas têm falado por anos vão finalmente se concretizar."Tim Berners-Lee*

1.1.1 Seleção de Protótipos

1.1.2 Bases Desbalanceadas

1.2 Objetivo

1.3 Estrutura do Trabalho

1.3.1 Sessões

1.3.2 Metodologia Utilizada

1.3.3 Bases de dados

CAPÍTULO 2

Técnicas de Seleção de Protótipos

Neste capítulo, serão mostradas as técnicas de seleção de protótipos abordadas neste trabalho. Cada uma das sessões abaixo abordará uma técnica, será mostrado o conceito da técnica, assim como o pseudo-código e as caracterísicas de cada uma destas técnicas.

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, ela elimina pontos na região de fronteira, região de alta susceptibilidade a erros, e com isso elimina ruídos.

Por atuar apenas na região de fronteira, esta técnica possui uma baixa capacidade de redução, deixando as instâncias que não se encontram na região de fronteira intactas, exceto pelos ruídos extremos.

Uma desvantagem desta técnica é que ela possui uma baixa capacidade de redução de elementos, visto que ela não elimina redundância.

Segue abaixo o algoritmo da execução do ENN e, logo após, alguns comentários sobre este algoritmo.

Algorithm 1 ENN

Require: *list*: uma lista

1. **for all** instância e_i da base de dados original **do**
2. Aplique o KNN sobre e_i
3. **if** e_i foi classificado erroneamente **then**
4. salve e_i em *list*
5. **end if**
6. **end for**
7. Remova da base de dados todos os elementos que estão em *list*

O valor de K pode variar de acordo com o tamanho da base de dados, porém, tipicamente, utiliza-se o valor de K=3. Tipicamente, 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.

Na figura **??** pode-se observar uma base de dados com duas classes, no primeiro gráfico da figura, pode-se observar a base de dados original, antes da aplicação do ENN. No segundo

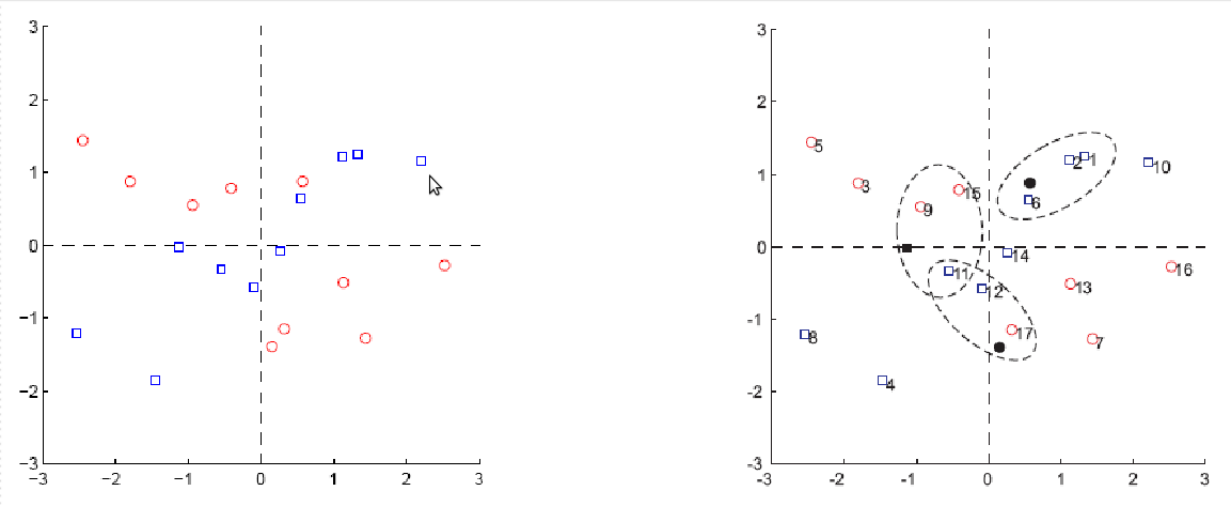

Figura 2.1 ENN aplicado com K=3

gráfico, foi aplicado o 1 com K=3 sobre a base de dados, os pontos pretos representam pontos que foram classificados errôneamente com a aplicação do KNN, a região circulada engloba os K elementos mais próximos. Observa-se que o elemento 11 foi utilizado para eliminar 2 instâncias ruidosas, um próprio ruído poderia ser utilizado para eliminar outro ruído, apesar de ser improvável.

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 independe da ordem que a base de dados foi apresentada, ou seja, 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 **??**, 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). Uma possível adaptação para o ENN em bases altamente desbalanceadas é eliminar apenas os elementos que sejam da base de dados marjoritária. O algoritmo adaptado está demonstrado em 2.

Com este algoritmo, as instâncias da classe minoritária seria mantida, e a região delimitada por ela seria mais bem definida.

Algorithm 2 ENN

Require: $list$: uma lista

1. **for all** instância e_i da base de dados original **do**
2. Aplique o KNN sobre e_i
3. **if** e_i foi classificado erroneamente **then**
4. **if** e_i for da classe marjoritária **then**
5. salve e_i em $list$
6. **end if**
7. **end if**
8. **end for**
9. Remova da base de dados todos os elementos que estão em $list$

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 estes que "são 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 de dados original corretamente. Fazendo isso, o algoritmo elimina os elementos mais afastados da região de indecisão, da fronteira de classificação.

O algoritmo do CNN será mostrado abaixo, e logo após, comentários a respeito do mesmo.

Algorithm 3 CNN

Require: $list$: uma lista

1. Escolha um elemento de cada classe $aleatoreamente$ e coloque-os em $list$
2. **for all** instância e_i da base de dados original **do**
3. $KNN(e_i, list)$
4. **if** e_i foi classificado erroneamente **then**
5. salve e_i em $list$
6. **end if**
7. **end for**
8. **return** $list$, os protótipos

Podemos observar que este algoritmo possui uma abordagem diferente do ENN, por exemplo, 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.

Uma coisa que pode-se observar no 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. 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.

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 abaixo o algoritmo desta adaptação:

Algorithm 4 CNN para bases desbalanceadas

Require: *list*: uma lista

1. Coloque todos os elementos da classe minoritária em *list*
2. **for all** instância e_i da base de dados original **do**
3. Aplique o KNN sobre e_i utilizando os elementos em *list* para treinamento
4. **if** e_i foi classificado erroneamente **then**
5. salve e_i em *list*
6. **end if**
7. **end for**
8. **return** base original - *list*

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. Esta adaptação do CNN irá reduzir a base de dados, ocasionando as vantagens de redução, e ainda reduzirá o desbalenceamento da base.

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 ser 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 $dist(e_i,e_j) < dist(e_i,e_k)$ e $dist(e_i,e_j) < dist(e_j,e_k)$. Segue o algoritmo detalhado em 5:

Os Tomek Links representam elementos da região de fronteira e prováveis ruídos, e a técnica de seleção de protótipos consiste em remover os Tomek Links da base de dados original. O algoritmo da seleção de protótipos Tomek Links é apresentado em 6:

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.

Observa-se facilmente que os Tomek Links pode remove todas as instâncias da fronteira, inclusive as instâncias da classe minoritária, assim sendo, uma possível adaptação dos Tomek Links é eliminar apenas os elementos das classes marjoritárias. Nesse caso, ainda ocorreria uma

Algorithm 5 Seleciona Tomek Links

Require: *list*: uma lista

1. **for all** instância e_i da base de dados original **do**
2. e_j = instância mais próxima de e_i
3. **if** instância mais p¿oxima de e_j for e_i **then**
4. **if** classe de e_i for diferente da classe de e_j **then**
5. salve o par *(e_i, e_j)* em *list*
6. **end if**
7. **end if**
8. **end for**
9. **return** *list*, Tomek Links

Algorithm 6 Tomek Links

Require: *list*: uma lista

1. *list* = *SelecionaTomekLinks* da base original
2. **for all** *(e_i, e_j)* em *list* **do**
3. remova e_i da base original
4. remova e_j da base original
5. **end for**
6. **return** base original filtrada

separação entre as classes, mas apenas as instâncias da classe marjoritária seriam removidos, dimunuindo assim o nível de desbalanceamento. Segue abaixo o algoritmo desta adaptação:

Algorithm 7 Tomek Links

Require: *list*: uma lista

1. *list* = *SelecionaTomekLinks* da base original
2. **for all** *(e_i, e_j)* em *list* **do**
3. **if** e_i for da classe marjoritária **then**
4. remova e_i da base original
5. **end if**
6. **if** e_j for da classe marjoritária **then**
7. remova e_j da base original
8. **end if**
9. **end for**
10. **return** base original - *list*

Com esta adaptação, a classe minoritária é mantida, evitando o aumento do desbalanceamento ou a remoção por alta probabilidade de ruído.

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 no algoritmo 8.

Algorithm 8 One-Sided Selection

Require: *list*: uma lista
Require: $tomek_links_list$: uma lista
1. $list = CNNparabasesdesbalanceadas$ sobre a base original
2. $list = TomekLinksparabasesdesbalanceadas$ sobre *list*
3. **return** *list*

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.

Uma desvantagem do One-Sided Selection é que ele não é determinístico, o CNN é não-determinístico e como o One-Sided Selection 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, 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]. O Learning Vector quantization 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 protótipos, de forma a posicionar cada instância em um ponto que seja possível estabelecer uma função discriminante baseada nestes protótipos.

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.

Algorithm 9 LVQ 1

Require: *prototypes*: uma lista para os protótipos
Require: *selection*: um algorítmo para seleção dos protótipos iniciais
1. *prototypes* = *selection* (base original)
2. **while** *prototypes* não estiver sub-ajustado **do**
3. x = *ChooseOne* (base original)
4. e_i = *SelectNearestFrom*(*prototypes*, x)
5. **if** classe de e_i != classe de 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** *prototypes*

2.6 LVQ 2.1

2.7 LVQ 3

2.8 SGP

2.9 SGP 2

2.10 CCNN

Referências Bibliográficas

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

[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.

[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.

[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.