5
Referências

1 Plutowski, M.; White, H. "Selecting Concise Training Sets from Clean Data", IEEE Transactions on Neural Networks, vol 4, no. 2, pp. 305-318, March 1993.

2 D. J. C. MacKay, "Information-based objective functions for active data selection", Neural Computation, vol 4, issue 4, pp. 590-604, July 1992.

3 Hasenjäger, M.; Ritter, H.; Obermayer, K. "Active Learning in Self-Organizing Maps", In Oja E. & Kaski S. editors, Kohonen Maps, pp 57-70, Elservier, 1999.

4 J. N. Hwang, J. J. Choi, S. Oh, R. J. Marks II, "Query-based learning applied to partially trained multi-layer perceptrons", IEEE Trans. Neural Networks, vol.2, no. 1, pp.131-136, Jan.1991.

5 J.J. Faraway, "Sequential design for the nonparametric regression of curves and surfaces", Proceedings of the 22nd Symposium on the Interface between Computing Science and Statistics, Springer, 104-110, 1990.

6 Pedreira, C.E., "Learning Vector Quantization with Training Data Selection", Publicação Interna, Departamento de Engenharia Elétrica, PUC-Rio, 2004.

7 Pedreira, C.E., **"**Uma Metodologia De Quantização Vetorial Com Escolha Seletiva De Dados" XV Congresso Brasileiro de Automática (CBA2004), Gramado, 2004.

8 T. Kohonen, "Self-Organizing Maps", Springer-Verlag, 2001.

9 Thiessen U, van Brakel R, de Weijer AP, Melssen WJ, Buydens LMC, "Using support vector machines for time series prediction" Chemometrics and Intelligent Laboratory Systems 69 (1-2): 35 – 49 Nov 28, 2003.

10 Kohonen, T., "Self-Organization and Associative Memory", Third edition, 1989.

11 Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., Torkkola, K., "LVQ PAK- The Learning Vector Quantization Program Package", Report A30,

Helsinki University of Technology, Laboratory of Computer and Information Science, January 1996.

12 Crammer, K., Gilad-Bachrach, R., Navot, A., Tishby, N. “Margin Analysis of the LVQ Algorithm”, In: NIPS 2002.

13 Duda, Hart, Stork “Pattern Classification”, Wiley-Interscience, Second Edition, 2001.

14 Kwak, N., Chong-Ho Choi. “Input Feature Selection by Mutual Information Based on Parzen Window”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 24, issue 14, pp.1667-1671, 2002.

15 Nicholson, A. “Generalization Error Estimates and Training Data Valuation”, Tese de Doutorado, 2002 – California Institute of Technology.

16 Karayiannis, N.B., Randolph-Gips, M.M., “Soft Learning Vector Quantization and Clustering Algorithms Based on Non-Euclidean Norms: Multinorm Algorithm”, IEEE Trans. on Neural Networks, vol.14, no.1, pp.89-102, Jan.2003.

17 Haykin, S. “Neural Networks, a Comprehensive Foundation”, Prentice Hall, 2 ed., 1999.

18 Cover, T.M.; Hart, P. E., “Nearest Neighbor Pattern Classification”, IEEE Trans. on Information Theory, vol. IT – 13, no. 1, pp. 21-27, Jan 1967.

19 Cover, T.M., “Estimation by the Nearest Neighbor Rule”, IEEE Trans. on Information Theory, vol. IT – 14, no. 1, pp. 50-55, Jan 1968.

20 Chang, C-L, “Finding Prototypes for Nearest Neighbor Classifiers”, IEEE Trans. on Computers, vol. C-23, no.11, pp. 1179-1184, Nov. 1974.

21 La Vigna, A., “Nonparametric Classification using LVQ”, M.sc., University of Maryland – 90-1, 1989.

22 Robins, H. & Monro, S., “A Stochastic Aproximation Method”, Annals of Mathematical Statistics, 22, 400-407.

23 Parzen, E., “On Estimation of a probability density function and mode”, Annals of Mathematical Statistics, 33(3):1065-1076, 1962.

24 Fisher, R.A. “The use of multiple measurements in taxonomic problems,” Annual Eugenics, 7, Part II, 179-188 7, 1936. Also in "Contributions to Mathematical Statistics" (John Wiley, NY, 1950).

6
Apêndice A – Seleção de Dados

Minimizar a taxa de erro no conjunto de generalização (fora-da-amostra) é o objetivo de qualquer estimador estatístico. Se o modelo ideal para um determinado problema fosse obtido, não existiriam preocupações com o erro. Entretanto, como é improvável a obtenção de um modelo ideal, almeja-se sempre o menor erro possível para o modelo estimado. Assim, todas as alternativas devem ser estudadas e, em muitos casos, pode ser possível alcançar uma performance melhor se o conjunto utilizado para treinamento do modelo for previamente trabalhado. Em um problema supervisionado [15], buscar no espaço de entrada os dados que são realmente representativos entre todos os exemplares disponíveis para treinamento passa a ser um problema importante para que o aprendizado do modelo obtenha o sucesso esperado. É importante citar que, nesse contexto, dados representativos são aqueles que representam bem sua distribuição e, além disso, não sejam ruídos[*].

Dessa forma, podem-se estabelecer critérios de seleção de dados que tornem o modelo escolhido mais eficiente, como abordado em [1]. Isso é importante porque é possível que modelos diferentes produzam resultados melhores com subconjuntos diferentes do conjunto de treinamento. Com isso, novas questões surgem, pois um modelo, obviamente, é dependente dos dados uma vez que é estimado a partir desses. Em contrapartida, os dados podem ser dependentes do modelo. Esse será o caso da abordagem desta dissertação, onde uma seleção de dados é feita posteriormente a estimativas do modelo.

Na verdade, o processo de modelagem é ainda mais amplo do que parece. Deve-se também observar a questão de seleção de características. Algoritmos de seleção de características [14] não precisam levar em consideração seleção de

[*] No sentido que será definido nas seções subseqüentes.

dados, e vice-versa. Entretanto, o que se busca é um modelo que apresente o melhor desempenho com um certo subconjunto dos dados de treinamento e com um subconjunto dos atributos. Seleção de dados e seleção de características são problemas que já vêm sendo abordados na literatura como em [1] e [14], respectivamente. Encontrar a combinação entre essas abordagens é ainda um problema em aberto e extremamente complicado. Nesta dissertação apenas a seleção de dados é abordada.

Abordagens diferentes têm sido encontradas na literatura para seleção de dados. Entretanto, todas as técnicas buscam de fato o mesmo objetivo, seleção inteligente de dados, ou seja, manipular o conjunto de treinamento de forma a alcançar determinados interesses futuros, seja ele o de melhor generalização dado um modelo prévio, economia de custo computacional ou encontrar um melhor modelo entre vários. Na verdade, a procura é feita sempre de modo que o aprendizado seja maximizado, ou seja, consiga-se encontrar o melhor resultado para um determinado problema.

Active Learning, Query-Based Learning e Sequential Design

Como essas técnicas são conhecidas na literatura pelos termos em inglês e não existem traduções usuais para elas na língua portuguesa, utilizaremos os termos em inglês ao longo do texto.

Os principais procedimentos de seleção inteligente de dados são “Active Learning” [1], [2], [3], “Query-Based Learning” [4] e “Sequential Design” [5].

Na verdade, há várias técnicas usando a filosofia de Active Learning. Em [1], busca-se a seleção de um subconjunto conciso dos dados de treinamento para que o número de pontos a serem treinados pelo modelo sugerido seja minimizado em relação ao conjunto original de exemplares disponíveis. Deseja-se fazer uma seleção de todos os candidatos para treinamento e, para isso, escolhe-se o ponto que minimiza o erro quadrático.

Outra abordagem bem diferente pode ser encontrada em [2], onde se apresentam três especificações diferentes em que se pode obter ganho de informação através de seleção de dados. Elas são:

- Para um modelo previamente selecionado, buscar dados no espaço de entrada que maximizem o resultado desse modelo, ou seja, permitam que ele generalize melhor;

- Selecionar dados em uma parte específica do espaço de entrada, onde não é possível amostrar diretamente. Aqui não existe a preocupação de encontrar uma função para todo o espaço, mas, sim, em uma região limitada;

- Entre dois ou mais modelos, selecionar os dados que possibilitem escolher entre eles, ou seja, procurar os dados que forneçam mais informações sobre esses modelos.

Outra forma de "active learning" está em [3], abordado no contexto de LVQ. Nesse caso, a intenção é selecionar dados que maximizem a redução da função de custo.

Query-Based Learning é um método cujo objetivo é realizar seleção de dados nas regiões em que futuros dados estarão agrupados. Segundo [1], a tarefa é minimizar o número de exemplares que se tem de juntar. [1] ainda menciona utilização de query learning para encontrar regiões de incerteza do espaço, procurando pelas funções nessas regiões.

Um exemplo do uso de Query-Based Learning está em [4]. O objetivo é encontrar pontos de incerteza e utilizá-los de maneira que o algoritmo informe a classificação certa.

O termo sequential design é usado em alguns artigos de econometria, por exemplo, em [5], para designar da mesma forma que active learning e query-based learning, a busca por pontos no espaço de entrada para a estimativa da função. Como o problema é tratado do ponto de vista econométrico, trata-se de estimar uma função presente numa regressão linear.

Uma Comparação Crítica entre os Três Métodos

Na verdade, todos os três métodos vistos na seção 2.2 buscam selecionar padrões específicos de acordo com uma proposta inicial, embora procedam de forma diferente durante o tratamento dos dados e, muitas vezes, possuam objetivos diferentes após o resultado. Mesmo utilizando a mesma nomenclatura, muitas vezes a forma como a técnica é desenvolvida difere totalmente de um autor para o outro, como se pode verificar em [1] e [2]. Com certeza, nada impede também que semelhanças entre técnicas de diferentes nomes apareçam.

Muitas vezes essa manipulação de dados pode ser útil em diferentes contextos. Entre os três exemplos selecionados como "active learning", [1] enfocou o custo computacional, mostrando, inclusive, que o resultado foi útil mesmo com o custo adicional de, a cada iteração, correr todo o conjunto de treinamento em busca do exemplar que minimizava o erro quadrático. [2] apresenta três especificações diferentes, cada uma levando a uma seleção de dados, inclusive, a oportunidade de selecionar um modelo entre vários, o que, com certeza, ressalta a importância desse procedimento para generalização. Finalmente, [3] utilizou "active learning" no contexto de LVQ.

Entretanto, a partir desses exemplos, fica claro que, independente do objetivo e da técnica utilizada, a seleção de dados em problemas estatísticos vem sendo utilizada como um recurso importante. O objetivo será estudá-la em um contexto supervisionado, especialmente em LVQ, fazendo com que a seleção de dados possa ser eficiente na atualização dos protótipos que designarão cada classe.

7
Apêndice B – LVQ

Aprendizado por quantização vetorial é um algoritmo proposto por Kohonen [8], usado em problemas supervisionados. Pela eficiência do método e facilidade de implementação, ele se tornou um algoritmo extremamente conhecido em problemas de classificação e reconhecimento de padrões. Além disso, variações dessa técnica surgiram acompanhadas do estudo de outros detalhes como a análise das margens [12]. Nesta seção os três modelos de LVQ mais conhecidos serão apresentados.

Segundo Kohonen [10], são selecionados um número fixo de vetores referenciais para cada classe, os chamados protótipos, vetores estes que serão atualizados durante a fase de aprendizado. Os principais passos do algoritmo mais simples de LVQ podem ser resumidos como especificado em [10]:

- Durante o treinamento, somente o vetor referência mais próximo é atualizado;
- O vetor referência será atualizado independente da classificação ter sido correta ou incorreta;
- O processo corretivo será metricamente compatível com o critério usado para classificação.

A idéia de criar protótipos associados a cada classe torna essa técnica muito interessante. Como lida com problemas supervisionados, a cada exemplar de treinamento, existe a possibilidade de compará-lo a indivíduos que fornecem as principais características de cada classe, pois um protótipo nada mais é do que isso, um representante padrão de uma certa classe. É importante citar que esses protótipos podem não existir fisicamente no banco de dados de interesse, são apenas elementos que servem de parâmetros para comparações entre os dados reais. Por exemplo, se imaginarmos os protótipos como os centróides das classes,

esses pontos podem não existir de fato, embora sejam os candidatos mais naturais a ocupar esse lugar.

LVQ 1

A abordagem inicial dessa técnica chama-se LVQ 1. Dado o conjunto de protótipos, a classificação na generalização será dada de acordo com a localização de cada elemento, ou seja, ele pertencerá à classe cujo protótipo estiver mais perto. Assim, como foi citado em [6] e [7], a escolha da métrica torna-se um problema chave dentro do contexto de LVQ, e a métrica Euclidiana nem sempre é a mais apropriada para a tarefa que se apresenta, pois considera que todos os atributos têm o mesmo peso [6], [7], [16]. Outro ponto a ser levado em consideração é que nada impede que um ponto pertencente à classe A, por exemplo, caia próximo de um protótipo representante da classe B. Por isso, o protótipo é atualizado durante o treinamento independente da classificação ser correta ou não, ou seja, para um determinado ponto, se ele pertence à mesma classe do protótipo que está mais próximo a ele, aproxima-se o protótipo, caso contrário, o protótipo é afastado, pois eles são de classes diferentes.

Usando a notação de [11] e formalizando a idéia, dado um conjunto de m protótipos e um vetor de entrada x, a classe de x na generalização será:

$$c = \arg\min_i \{ \| x - m_i \| \}$$

A classe c representada pelo protótipo mais próximo de x será a classe desse elemento. Como foi dito acima, para que esse processo seja realizado na generalização com sucesso, é necessário que durante o treinamento seja realizada a atualização dos protótipos. Seja uma amostra x apresentada ao algoritmo pela n-ésima vez, o algoritmo básico do LVQ 1 [11] é o seguinte:

$m_c(n+1) = m_c(n) + \alpha(n)[x(n) - m_c(n)]$, se x e m_c pertencem a mesma classe,

$m_c(n+1) = m_c(n) - \alpha(n)\left[x(n) - m_c(n)\right]$, se x e m_c pertencem a classes diferentes,

$m_i(n+1) = m_i(n)$, para i $\neq$ c,

onde 0 < $\alpha(n)$< 1. Temos, então, que, quando o elemento x pertence a mesma classe do protótipo mais próximo, eles são aproximados, caso contrário são afastados. A terceira condição garante que todos os protótipos de classes diferentes do mais próximo ao padrão permanecerão inalterados.

LVQ 2.1

A decisão de classificação nesse algoritmo é idêntica a do LVQ 1. A diferença, entretanto, está no aprendizado. No LVQ 2.1, são considerados o protótipo mais próximo do exemplar x que possua a mesma classe que ele e o protótipo mais próximo de x cuja classe seja diferente do exemplar. Assim, ao contrário do LVQ 1, a cada iteração dois protótipos são atualizados. O algoritmo é o seguinte [11]:

$$m_i(n+1) = m_i(n) - \alpha(n)\left[x(n) - m_i(n)\right]$$

$$m_j(n+1) = m_j(n) + \alpha(n)\left[x(n) - m_j(n)\right]$$

onde m_i representa o protótipo mais próximo cuja classe é diferente de x e m_j representa o protótipo mais próximo que possui a mesma classe de x.

Além disso, ainda por [11], x deve cair dentro de uma faixa de valores, denominada janela, que é definida por um plano que passa no ponto médio da distância entre m_i e m_j. Supondo que d_i e d_j são as distâncias Euclidianas de x a m_i e m_j, então x cairá numa janela de largura w, se

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s, \quad s = \frac{1-w}{1+w}$$

A recomendação é de que a janela tenha largura w entre 0.2 e 0.3.

Note que, ao atualizar os protótipos somente quando um padrão cair dentro da janela, o LVQ está fazendo seleção de dados. Se um padrão x tiver o protótipo m da mesma classe mais próximo a ele, então m será aproximado de x somente se x estiver a uma certa distância de m, que é estabelecida pela janela. Da mesma forma, se o protótipo de outra classe k for o mais próximo de x, k só será afastado de x se ele cair no intervalo da janela. Entretanto, essa forma de seleção de dados pode ser considerada limitada por garantir a utilização de um padrão apenas pelo fato de ele pertencer a um intervalo, que foi escolhido a priori, sem nenhuma garantia teórica de que esse intervalo se aproxima do ótimo para escolha de padrões.

LVQ 3

Segundo [11], havia um problema no LVQ 2.1 em relação a localização que m_i se encontraria se o processo fosse longo o bastante. A partir disso, algumas alterações tiveram que ser feitas para que m_i continuasse se aproximando da distribuição de sua classe. Surgiu, então, uma mudança no algoritmo que foi chamada de LVQ 3.

$$m_i(n+1) = m_i(n) - \alpha(n)\left[x(n) - m_i(n)\right]$$

$$m_j(n+1) = m_j(n) + \alpha(n)\left[x(n) - m_j(n)\right]$$

A atualização dos protótipos continuou da mesma forma, onde m_i representa o protótipo mais próximo cuja classe é diferente de x e m_j representa o protótipo mais próximo que possui a mesma classe de x. Além disso, x deve cair na janela:

$$m_k(n+1) = m_k(n) + \varepsilon\alpha(n)\left[x(n) - m_k(n)\right]$$

para k $\in$ {i,j}, se x, m_i e m_j pertencerem a mesma classe.

Análise de Margem

Em [12] vemos que o estudo da análise de margem é o que garante consistência teórica aos algoritmos de LVQ. Esse tipo de técnica foi muito usado em "machine learning" durante a década de 90.

Existem duas abordagens diferentes para definir margem. A primeira é defini-la como a distância entre uma instância e a fronteira de decisão, induzida pela regra de classificação. A essa definição dá-se o nome de "Sample Margin". Existe, entretanto, uma outra definição, chamada "Hypothesis Margin". Nesse caso, margem é a distância que o classificador pode viajar sem mudar a classificação de qualquer ponto. Embora a primeira definição seja mais intuitiva, ela é mais difícil de ser implementada pelo custo computacional que exige. Assim, em geral, foca-se em "Hypothesis Margin".

Uma Avaliação Crítica Sobre o Modelo LVQ

O modelo LVQ é baseado em quantização vetorial. Na verdade, o algoritmo deseja encontrar a posição ótima no espaço para representantes especiais dos dados, que são os protótipos, e classificar novos dados a partir da distância a esses protótipos. No contexto de generalização, a proposta dessa técnica se assemelha muito a classificação por vizinhos mais próximos [18], [19], [20]. Certamente, o fato de estabelecer a classificação mediante a distância a um pequeno grupo de elementos (os protótipos), permite uma comparação principalmente com knn (ou k vizinhos mais próximos), método muito parecido com o original vizinhos mais próximos, e que estabelece a classificação mediante a distância entre os k elementos mais próximos ao ponto. Os dois métodos são muito semelhantes quanto à forma de classificar.

Em relação à atualização dos protótipos durante o treinamento, o algoritmo proposto por Kohonen em [8], é baseado em gradiente descendente, que na verdade usa o clássico método de aproximação estocástica [21], [22] para estimar a posição ótima dos protótipos. Como em todos os métodos que utilizam gradiente descendente, a escolha da taxa de aprendizado é uma tarefa importante. No caso do LVQ, uma taxa muito pequena pode levar a uma demora muito grande de chegada do protótipo ao lugar ótimo no espaço dos dados. Já uma taxa muito grande pode levar o algoritmo a conduzir os protótipos a regiões distantes do ponto ótimo. Segundo [8], a taxa deve ser pequena, em geral, menor que 0.1, e pode, inclusive, decrescer monotonicamente de acordo com o processo de treinamento. Nesse caso deve se estipular uma seqüência k_n divergente, com qualquer potência maior ou igual a 2 de k_n convergente [22].

Conforme mencionado nas subseções 2.4.2 e 2.4.3, a partir do LVQ 2.1, o processo traz embutido uma proposta para seleção de dados, atualizando os protótipos somente se o padrão cair dentro de uma janela especificada a priori. Essa característica mostra a importância de obter dados consistentes para atualizar protótipos, embora a maneira como seja feita nos métodos clássicos torne essa seleção de dados muito limitada, já que é baseada numa largura de janela obtida sem nenhuma evidência teórica de sua validade.

Finalmente, tendo em vista os três métodos de LVQ apresentados nesse capítulo, de acordo com [11], LVQ 1 e LVQ 3 definem processos mais robustos e os protótipos assumem valores eficientes mesmo depois de longos períodos de treinamento. Em relação ao LVQ 2.1, não existe garantia de que os protótipos estarão localizados na posição ótima depois do treinamento. Assim, essa técnica é indicada para pequenos conjuntos de treinamento e deve ser usada com taxas de aprendizado apropriadas.