Referências Bibliográficas

[1] BANZHAF, W.; NORDIN, P.; KELLER, R. ; FRANCONE, F.. **Genetic Programming - An Introduction**. Morgan Kaufmann, 1998.

[2] BARES, W.; MCDERMOTT, S.; BOUDREAUX, C. ; THAINIMIT, S.. **Virtual 3D camera composition from frame constraints**. In: MULTIMEDIA, p. 177–186. ACM, 2000.

[3] BARES, W. H.. **A Photographic Composition Assistant for Intelligent Virtual 3D Camera Systems**. In: SMART GRAPHICS, p. 172–183, 2006.

[4] BLANZ, V.; TARR, M. J.; BÜLTHOFF, H. H. ; VETTER, T.. **What object attributes determine canonical views**. Perception, 28(5):575–600, 1999.

[5] BORDIGNON, A.; PEREIRA, R.; LOPES, H.; LEWINER, T. ; TAVARES, G.. **Exploratory visualization based on multidimensional transfer functions and star coordinates**. In: SIBGRAPI 2006 (XIX BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 273–280, Manaus, AM, october 2006. IEEE.

[6] BORDIGNON, A.; VATH, B.; VIEIRA, T.; FERREIRA, C.; CRAIZER, M. ; LEWINER, T.. **Scale-space for union of 3d balls**. In: SIBGRAPI 2009 (XXII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Rio de Janeiro, RJ, october 2009. IEEE.

[7] BOSER, B. E.; GUYON, I. M. ; VAPNIK, V. N.. **A Training Algorithm for Optimal Margin Classifiers**. In: COMPUTATIONAL LEARNING THEORY, p. 144–152. ACM, 1992.

[8] CHANG, C.-C.; LIN, C.-J.. **LIBSVM: a library for support vector machines**, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] CHRISTIE, M.; MACHAP, R.; NORMAND, J.-M.; OLIVIER, P. ; PICKERING, J.. **Virtual Camera Planning: A Survey**. In: SMART GRAPHICS, p. 40–52, 2005.

[10] CHRISTIE, M.; OLIVIER, P. ; NORMAND, J.-M.. **Camera Control in Computer Graphics**. Computer Graphics Forum, 27(7), 2008.

[11] CORTES, C.; VAPNIK, V.. **Support-vector networks**. Springer, 1995.

[12] DRUCKER, S. M.; ZELTZER, D.. **Intelligent Camera Control in a Virtual Environment**. In: GRAPHICS INTERFACE, p. 190–199, 1994.

[13] FEIJO, B.; PAGLIOSA, P. ; CLUA, E. W. G.. **Visualização, simulação e games**. In: XXIV JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA DO CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, volumen 1, p. 151–223, Rio de Janeiro, RJ, 2006. Editora PUC-Rio.

[14] GÓMEZ, F.; HURTADO, F.; SELLARÈS, J. A. ; TOUSSAINT, G.. **Nice Perspective Projections**. Visual Communication and Image Representation, 12(4):387–400, 2001.

[15] GOOCH, B.; REINHARD, E.; MOULDING, C. ; SHIRLEY, P.. **Artistic Composition for Image Creation**. In: RENDERING TECHNIQUES, p. 83–88. Springer, 2001.

[16] HALPER, N.; OLIVER, P.. **CamPlan: A Camera Planning Agent**. In: SMART GRAPHICS, p. 92–100, 2000.

[17] HE, L.-W.; COHEN, M. F. ; SALESIN, D. H.. **The virtual cinematographer: a paradigm for automatic real-time camera control and directing**. In: SIGGRAPH, p. 217–224. ACM, 1996.

[18] HE, T.; HONG, L.; KAUFMAN, A. ; PFISTER, H.. **Generation of transfer functions with stochastic search techniques**. In: VISUALIZATION, p. 227–234, Los Alamitos, CA, USA, 1996. IEEE.

[19] JANKUN-KELLY, T. J.; MA, K.-L.. **Visualization exploration and encapsulation via a spreadsheet-like interface**. IEEE Transactions on Visualization and Computer Graphics, 7(3):275–287, 2001.

[20] KAMADA, T.; KAWAI, S.. **A simple method for computing general position in displaying three-dimensional objects**. Computer Vision, Graphics, Image Processing, 41(1):43–56, 1988.

[21] LAGA, H.; NAKAJIMA, M.. **Supervised Learning of Salient 2D Views of 3D Models**. In: NICOGRAPH, 2007.

[22] LAGE, M.; LEWINER, T.; LOPES, H. ; VELHO, L.. **Chf: a scalable topological data structure for tetrahedral meshes**. In: SIBGRAPI 2005 (XVIII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Natal, RN, october 2005. IEEE.

[23] LEE, C. H.; VARSHNEY, A. ; JACOBS, D. W.. **Mesh saliency**. In: SIGGRAPH, p. 659–666. ACM, 2005.

[24] LEWINER, T.; AO D. GOMES, J.; LOPES, H. ; CRAIZER, M.. **Arc-length based curvature estimator**. In: SIBGRAPI 2004 (XVII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 250–257, Curitiba, PR, october 2004. IEEE.

[25] LIM, I. S.; DE HERAS CIECHOMSKI, P.; SARNI, S. ; THALMANN, D.. **Planar arrangement of high-dimensional biomedical data sets by Isomap coordinates**. In: IEEE SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS. IEEE, 2003.

[26] MARKS, J.; ANDALMAN, B.; BEARDSLEY, P. A.; FREEMAN, W.; GIBSON, S.; HODGINS, J.; KANG, T.; MIRTICH, B.; PFISTER, H.; RUML, W.; RYALL, K.; SEIMS, J. ; SHIEBER, S. M.. **Design galleries: a general approach to setting parameters for computer graphics and animation**. In: SIGGRAPH, p. 389–400. ACM, 1997.

[27] MAXIMO, A.; MARROQUIM, R.; ESPERANÇA, C.; DOS SANTOS, R. W.; BENTES, C. ; FARIAS, R.. **High-performance volume rendering for 3d heart visualization**. In: WORKSHOP ON HIGH PERFORMANCE COMPUTING IN THE LIFE SCIENCES - HPCLIFE 2006, OURO PRETO-MG, October 2006.

[28] MEYER, M.; DESBRUN, M.; SCHRÖDER, P. ; BARR, A.. **Discrete differential–geometry operators for triangulated 2–manifolds**. In: VISMATH, p. 35–57, 2002.

[29] NGAN, A.; DURAND, F. ; MATUSIK, W.. **Image-driven navigation of analytical brdf models**. In: RENDERING TECHNIQUES, p. 399–408, June 2006.

[30] PASSOS, E.; JOSELLI, M.; ZAMITH, M.; ROCHA, J.; MONTENEGRO, A. A.; CONCI, A.; FEIJO, B. ; CLUA, E. W. G.. **Supermassive crowd**

simulation on gpu based on emergent behavior. CIE, 7(2):145–155, 2009.

[31] PINTO, F.; FREITAS, C.. **Two-level interaction transfer function design combining boundary emphasis, manual specification and evolutive generation**. In: SIBGRAPI 2006 (XIX BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 281–288, Manaus, AM, october 2006. IEEE.

[32] PLATT, J. C.. **Fast training of support vector machines using sequential minimal optimization**. MIT Press, Cambridge, MA, USA, 1999.

[33] PLEMENOS, D.; BENAYADA, M.. **Intelligent Display in Scene Modeling: New Techniques to Automatically Compute Good Views**. In: GRAPHICON, 1996.

[34] POLONSKY, O.; PATANÈ, G.; BIASOTTI, S.; GOTSMAN, C. ; SPAGNUOLO, M.. **What's in an image?** The Visual Computer, 21(8–10):840–847, 2005.

[35] RICARDO MARROQUIM; ANDRE MAXIMO; RICARDO FARIAS ; CLAUDIO ESPERANÇA. **Volume and Isosurface Rendering with Gpu-accelerated Cell Projection**. Computer Graphics Forum, 27:24–35, 2008.

[36] SCHAREIN, R. G.. **Interactive Topological Drawing**. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.

[37] SCHÖLKOPF, B.; SMOLA, A. J. ; MÜLLER, K.-R.. **Kernel principal component analysis**, p. 327–352. MIT, 1999.

[38] SHAMIR, A.. **A survey on mesh segmentation techniques**. Computer Graphics Forum, 27(6):1539–1556, 2008.

[39] SHAPIRA, L.; SHAMIR, A. ; COHEN-OR, D.. **Image Appearance Exploration by Model-Based Navigation.** Computer Graphics Forum, 28(2):629–638, 2009.

[40] SHILANE, P.; FUNKHOUSER, T.. **Distinctive regions of 3D surfaces**. Transactions on Graphics, 26(2):7, 2008.

[41] STOEV, S.; STRASSER, W.. **A Case Study on Automatic Camera Placement and Motion for Visualizing Historical Data**. In: VISUALIZATION. IEEE, 2002.

[42] SYSTEMS, A.. **After effects cs4**. Software, Outubro 2008.

[43] SYSTEMS, A.. **Photoshop cs4**. Software, Outubro 2008.

[44] TARJAN, R. E.. **Efficiency of a good but not linear set union algorithm**. Journal of the ACM, 22(2):215–225, 1975.

[45] VAPNIK, V.. **The Nature of Statistical Learning Theory**. Springer, 2000.

[46] VÁZQUEZ, P.-P.; FEIXAS, M.; SBERT, M. ; HEIDRICH, W.. **Viewpoint Selection using Viewpoint Entropy**. In: VISION MODELING AND VISUALIZATION, p. 273–280. Aka, 2001.

[47] VIEIRA, T.. **Registro Automático de Superfícies usando Spin-Images**. Master's thesis, Universidade Federal de Alagoas, Feb. 2007. Advised by Adelailson Peixoto.

[48] VIEIRA, T.; BORDIGNON, A.; LEWINER, T. ; VELHO, L.. **Geometry super-resolution by example**. In: SIBGRAPI 2009 (XXII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Rio de Janeiro, RJ, october 2009. IEEE.

[49] VIEIRA, T.; BORDIGNON, A.; PEIXOTO, A.; TAVARES, G.; LOPES, H.; VELHO, L. ; LEWINER, T.. **Learning good views through intelligent galleries**. Computer Graphics Forum, 28:717–726, 2009.

[50] VIEIRA, T.; PEIXOTO, A.; VELHO, L. ; LEWINER, T.. **An iterative framework for registration with reconstruction**. In: VISION, MODELING, AND VISUALIZATION 2007, p. 101–108, Saarbrücken, november 2007. Pirrot.

A
Máquinas de Suporte Vetorial

A área de aprendizagem de máquina se preocupa com o desenvolvimento de algoritmos que permitam aos computadores aprender com dados de entrada provenientes, por exemplo, de sensores ou bancos de dados. O conhecimento adquirido pela máquina é utilizado para reconhecer automaticamente padrões complexos e tomar decisões inteligentes.

Uma das classes de algoritmos que tenta realizar aprendizagem de máquina é chamada de aprendizagem supervisionada. Esses algoritmos tentam deduzir uma função a partir de dados de treino contendo a resposta esperada da máquina. Em geral, esses dados de treino consistem de objetos (*input*), normalmente na forma de vetores, associados a saídas (*output*) desejadas. Dependendo do problema, essas saídas podem ser representadas por um valor contínuo, caracterizando um problema de regressão, ou valores discretos, também chamados de classes, caracterizando um problema de classificação.

As Máquinas de Suporte Vetorial são um conjunto de métodos de aprendizagem supervisionada usadas para classificação e regressão de dados. Em geral, esses métodos calculam um hiperplano, ou conjunto de hiperplanos, cujos parâmetros serão utilizados na função de classificação ou regressão. Vamos nos focar agora no problema de classificação binária. Para maiores detalhes, referenciamos o leitor ao livro de Vapnik (45).

A.1
Classificador linear

Inicialmente, a máquina recebe um conjunto de dados de treino (*training set*) na forma:

$$\Gamma = \{(x_i, c_i) | x_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^{N}, \tag{A-1}$$

onde x_i representa um dado de entrada p-dimensional, e c_i sua classificação.

Nosso objetivo é encontrar um hiperplano que separe, da melhor maneira possível, os pontos com classificação $c_i = -1$ dos pontos classificados com

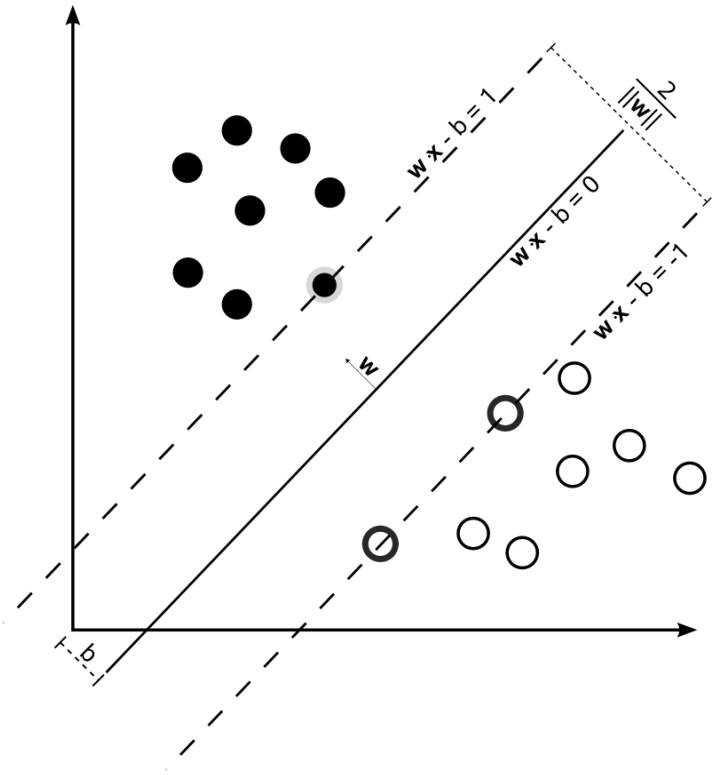

Figura A.1: Exemplo de hiperplano de margem máxima separando duas classes.

$c_i = 1$, como ilustrado na figura A.1. Este hiperplano será chamado de hiperplano de margem máxima.

Todo hiperplano pode ser descrito por uma equação do tipo

$$w \cdot x - b = 0, \tag{A-2}$$

onde $w \neq 0$ é o vetor normal do hiperplano e $\frac{b}{\|w\|}$ determina o deslocamento do hiperplano a partir da origem, ao longo do vetor w.

Queremos escolher w_{opt} e b_{opt} de modo que os hiperplanos paralelos dados pelas equações

$$w_{opt} \cdot x - b_{opt} = 1$$

e

$$w_{opt} \cdot x - b_{opt} = -1$$

maximizem a distância entre si, dada por $\frac{2}{\|w\|}$, respeitando a condição de separar os dados de acordo com suas classes. O hiperplano de margem máxima será dado pela equação $w_{opt} \cdot x - b_{opt} = 0$.

Estabelecemos, portanto, um problema de otimização com restrições, onde queremos maximizar a distância entre os planos $\frac{2}{\|w\|}$, considerando restrições que nos garantam que esses hiperplanos separem os dados de acordo com sua classe, ou seja:

$$c_i(w \cdot x_i - b) \geq 1, \quad i = 1 \ldots N. \tag{A-3}$$

Com uma pequena adaptação nesta formulação, chegamos à forma primal, caracterizada como um problema de otimização quadrática, que explicitamos abaixo:

Minimize (em w, b)

$$\frac{1}{2}\|w\|^2$$

sujeito a

$$c_i(w \cdot x_i - b) \geq 1, \quad i = 1 \ldots N.$$

Para deduzir a forma dual Lagrangeana, usamos a função Lagrangeana generalizada:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[c_i(w \cdot x_i + b) - 1\right], \tag{A-4}$$

onde α_i são os multiplicadores de Lagrange.

Minimizando $L(w, b, \alpha)$ em relação a w e b, chegamos às relações

$$w = \sum_{i=1}^{N} \alpha_i c_i x_i \tag{A-5}$$

e

$$\sum_{i=1}^{N} \alpha_i c_i = 0.$$

Substituindo estas relações de volta em $L(w, b, \alpha)$ chegamos à forma dual:

Maximize (em α_i)

$$\mathcal{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j c_i c_j x_i^T x_j \tag{A-6}$$

sujeito a

$$\alpha_i \geq 0, \quad i = 1 \ldots N \tag{A-7}$$

e

$$\sum_{i=1}^{n} \alpha_i c_i = 0, \quad i = 1 \ldots N. \tag{A-8}$$

Usando esta formulação dual, é possível encontrar w_{opt} substituindo-se os valores de α_{opt} na equação (A-5). Para encontrar b_{opt}, é necessário voltar à forma primal:

$$b_{opt} = -\frac{\max_{c_i=-1}(w_{opt} \cdot x_i) + \min_{c_i=1}(w_{opt} \cdot x_i)}{2}.$$

De acordo com a condição de complementaridade de Karush-Kuhn-Tucker, as soluções ótimas $\alpha_{opt}, w_{opt}, b_{opt}$ devem satisfazer

$$(\alpha_i)_{opt}\,[c_i(w_{opt}\cdot x_i + b) - 1] = 0, \quad i = 1 \ldots N.$$

Isto significa que um ponto x_i só pode ter seu multiplicador de Lagrange α_i não nulo se $c_i(w_{opt}\cdot x_i + b) = 1$, ou seja, quando este ponto pertence a um dos hiperplanos que define a margem. Este resultado é extremamente importante, pois, de acordo com a equação (A-5), só precisamos dos pontos x_i na margem para representar w. Esses pontos serão chamados de vetores de suporte. É interessante enfatizar que os vetores de suporte são geralmente poucos em relação ao tamanho do training set, pois só ocorrem na fronteira entre as classes.

Uma vez identificados os vetores de suporte x_i com seus respectivos α_i, podemos expressar nossa função classificadora como:

$$\hat{f}(x) = \sum_{i\in SV} [\alpha_i c_i (x_i \cdot x)] + b, \tag{A-9}$$

onde SV é o conjunto com os índices dos vetores de suporte.

Quando $\hat{f}(x) = 0$, o ponto x está contido no hiperplano que divide as amostras. Em qualquer outro caso, usamos $sign(\hat{f}(x))$ para classificar um ponto x como pertencente à classe 1 ou -1.

Até agora, assumimos que os dados são linearmente separáveis, ou seja, que é possível separar dados de diferentes classes usando apenas um hiperplano. Descreveremos agora duas generalizações deste método para tratar dados ruidosos e dados não linearmente separáveis.

A.2
Classificador linear com margem flexível

Em muitos cenários é necessário trabalhar com conjuntos de dados de treino que não são linearmente separáveis por causa de ruído nas classificações, por exemplo. Cortes *et al.* (11) sugeriram uma modificação da formulação original para tratar este tipo de problema. A idéia é que, quando não for possível encontrar um plano que divida os pontos das duas classes, pode ser suficiente encontrar o hiperplano que minimiza o erro de classificação dos pontos de treino.

Para medir o possível erro de classificação do hiperplano, são introduzidas variáveis ξ_i associadas às restrições da forma primal do problema original (figura A.2):

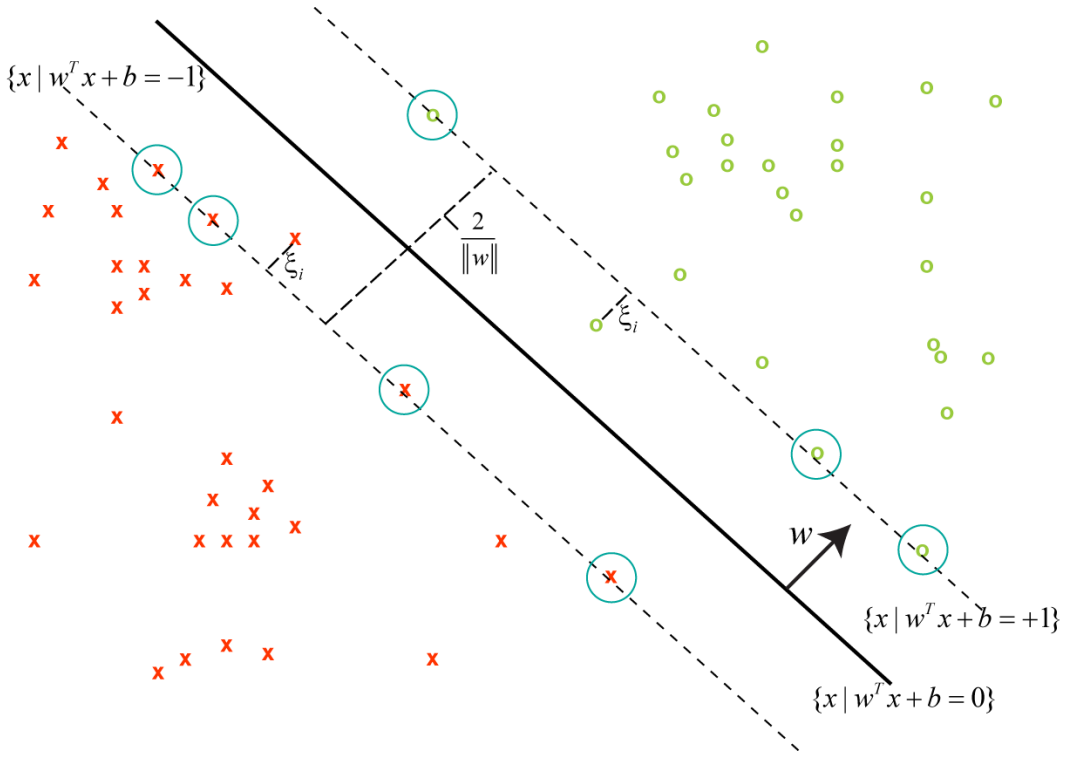

Figura A.2: Formulação das Máquinas de Suporte Vetorial usando margem flexível: a condição do hiperplano ótimo é relaxada, permitindo que algumas amostras ultrapassem a margem.

$$c_i(w \cdot x_i - b) \geq 1 - \xi_i, \quad i = 1 \ldots n. \tag{A-10}$$

Para minimizar este erro, introduzimos um novo termo na função objetivo da minimização:

$$\frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{A-11}$$

onde C é um peso associado ao erro tolerável. A solução deste problema de minimização é similar ao procedimento descrito na seção anterior.

A.3 Classificador não-linear

Quando os dados de treino não forem linearmente separáveis, é possível recorrer a uma extensão proposta por Boser *et al.* (7). Neste método, a função de classificação $\hat{f}$ corresponde a uma função linear em um espaço de Hilbert $\mathcal{F}$, chamado de *feature space*. Em particular, o hiperplano deduzido para criação de $\hat{f}$ estará contido não mais no espaço de entrada $\mathbb{R}^p$, mas em um espaço de alta dimensão ou dimensão infinita $\mathcal{F}$, o que na prática significa que ele pode não ser linear no espaço de entrada.

Inicialmente, é necessário definir uma função não-linear $\phi: \mathbb{R}^p \mapsto \mathcal{F}$, chamada de *feature map*, que mapeia os pontos de treino $x_i \in \mathbb{R}^p$ em $\mathcal{F}$.

Dada uma função ϕ, definimos um *kernel* como uma função $K: \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$, tal que $\forall x_i, x_j \in \mathbb{R}^p$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j).$$

Uma função do tipo *kernel* generaliza o produto interno usual no espaço de entrada $\mathbb{R}^p$, e uma de suas vantagens é que ele possibilita mapear dados implicitamente para o *feature space* e treinar a máquina, sem a necessidade de conhecer uma função ϕ potencialmente complicada e problemática computacionalmente. A existência de ϕ a partir de K é garantida pelo Teorema de Mercer, se K for simétrica positiva definida. Um exemplo bastante utilizado é o *kernel* Gaussiano, dado por

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right). \tag{A-12}$$

Neste caso, $\mathcal{F}$ é um espaço de dimensão infinita, mas K é simples, pois é multiplicativo em dimensão.

Na prática, obtemos um classificador não-linear usando o mesmo algoritmo descrito na seção A.1, apenas substituindo cada produto interno pelo *kernel* não linear escolhido. A função classificadora resultante será dada por

$$\hat{f}(x) = \sum_{i \in SV} [\alpha_i c_i K(x_i, x)] + b. \tag{A-13}$$

É interessante notar que em momento algum é necessário conhecer a função ϕ.

A.4
Adequação às Galerias Inteligentes

A forma dual das Máquinas de Suporte Vetorial caracteriza um problema de otimização quadrática convexa (equação (A-6)) com restrições lineares (equações (A-7) e (A-8)). Como neste tipo de problema a solução é única e não existem máximos locais, é possível encontrar esta solução eficientemente com custo proporcional à quantidade de dados de entrada N. Um método eficiente e paralelizável para este propósito é o SMO (32). A LibSVM (8) é uma boa alternativa de biblioteca de código aberto em C++ que implementa um método do tipo SMO.

Como mencionado anteriormente, a quantidade de vetores de suporte M é em geral bem menor que a quantidade de amostras N. Em particular, isso torna o custo de avaliar um dado de entrada x_i dado pela equação (A-9) muito

eficiente, com ordem de grandeza $O(M) \ll O(N)$.

Além disso, os problema de otimização e de avaliação dos dados de entrada não dependem diretamente da dimensão dos dados de entrada p, devido ao uso de *kernels*. Apenas a avaliação do *kernel* nos pontos é dependente da dimensão p. Isto implica que é possível acrescentar descritores sem prejudicar o custo computacional.

Finalmente, em termos de memória, uma máquina é representada apenas pelos vetores de suporte, além dos vetores w e b do hiperplano e parâmetros do *kernel* K, o que garante baixo custo de armazenamento.

B
Algoritmos Genéticos

Os algoritmos genéticos representam uma classe de algoritmos que tentam achar soluções aproximadas em problemas de otimização. Esses algoritmos recebem este nome porque se baseiam em princípios da biologia evolucionária como herança, mutação, seleção e *crossover*. Descreveremos a seguir os passos de um algoritmo genético, e o algoritmo usado neste trabalho no contexto de otimização de parâmetros. Recomendamos ao leitor o livro de Banzhaf *et al.* (1).

B.1
Algoritmo genético usual

Um algoritmo genético tem como objetivo encontrar, em uma população, o indivíduo que tem a melhor avaliação de acordo com uma função objetivo $\hat{f}: \Omega \mapsto \mathbb{R}$. Esta busca segue por diversas gerações de populações, seguindo o princípio de que a população evolui a cada nova geração, até que algum critério de parada seja atingido.

A elaboração de um algoritmo genético para resolver um problema específico requer:

1. estabelecer uma representação genética dos indivíduos, ou seja, definir o domínio Ω que representa os genes do indivíduo;

2. criar a função objetivo $\hat{f}$.

Uma vez definidos o domínio Ω e a função objetivo $\hat{f}$, um algoritmo genético deve seguir os passos descritos no algoritmo B.1. Descreveremos a seguir cada etapa do algoritmo.

B.1.1
Inicialização

Nesta etapa, é necessário estabelecer algum procedimento para gerar uma população inicial $\Psi \subset \Omega$, que será dada como entrada para o algoritmo. Uma estratégia comum é amostrar uniformemente o domínio Ω.

entrada: População inicial $\Psi \subset \Omega$
saída : Melhor indivíduo $\omega \in \Omega$
1 **enquanto** *nenhuma condição de parada atingida* **faça**
2 | Pais := Seleção(Ψ,$\hat{f}$);
3 | Ψ := Reprodução(Pais);
4 Retorne melhor parâmetro $\omega \in \Psi$ de acordo com $\hat{f}$.

Algoritmo B.1: Esqueleto de um algoritmo genético

B.1.2 Seleção

Em cada geração, indivíduos da população serão selecionados para reprodução. Para que a população evolua, devemos selecionar indivíduos portadores de bons genes, ou seja, indivíduos com boa avaliação de acordo com a função objetivo $\hat{f}$. Em geral, o processo de seleção ocorre como um processo estocástico, onde a probabilidade de um indivíduo ser selecionado depende de sua avaliação segundo $\hat{f}$. Isto tenta garantir que a qualidade média da população evolua com o tempo, visto que indivíduos com melhor avaliação tem maior probabilidade de se reproduzirem.

B.1.3 Reprodução

Nesta etapa, os pais selecionados na etapa anterior serão emparelhados para gerar uma nova geração de indivíduos. Cada par de pais pode passar por dois operadores genéticos: *crossover* e mutação. O primeiro operador recebe genes de dois indivíduos e gera um novo indivíduo, estabelecendo alguma estratégia para combinar genes dos pais. O segundo operador realiza pequenas mutações aleatórias no código genético, para garantir a variabilidade genética.

B.2 Algoritmo genético para otimização de parâmetros

Nesta seção descreveremos uma abordagem para resolver um problema de otimização de parâmetros usando um algoritmo genético baseado em combinações convexas.

Como descrito na seção anterior, precisamos inicialmente definir nossa representação genética dos indivíduos. Vamos considerar, como indivíduos, parâmetros $\omega \in \Omega \subset \mathbb{R}^n$, onde Ω é um hipercubo. Esta restrição é recomendável, visto que realizaremos combinações convexas entre elementos deste domínio. Entretanto, podemos usar esse hipercubo para representar domínios

de parâmetros mais complexos usando parametrizações. Deixaremos a função objetivo $\hat{f}$ em aberto.

Na fase de inicialização do algoritmo, é necessário criar um conjunto de amostras do domínio Ω. Para garantir que qualquer parâmetro $\omega \in \Omega$ possa ser gerado como resultado de combinações convexas do conjunto de amostras inicial, devemos incluir pelo menos os vértices do hipercubo Ω. Outras amostras podem ser geradas uniformemente no hipercubo.

Vamos considerar agora um conjunto de parâmetros $\Psi \subset \Omega$ selecionados do modo convencional para reprodução. Nesta fase, os parâmetros serão emparelhados aleatoriamente ν vezes, e cada par será reproduzido usando combinações convexas aleatórias. Desse modo, garantimos que cada pai vai gerar ν filhos, e a nova população será de $\frac{\nu}{2} \cdot |\Psi|$ indivíduos.

Seja (ω_i, ω_j) um dos pares criados aleatoriamente, e X uma variável aleatória uniforme em $[0, 1]$. Definimos nosso operador de *crossover* como $\xi: \Omega \times \Omega \mapsto \Omega$ dado por

$$\xi(\omega_i, \omega_j) = \omega_i + X(\omega_j - \omega_i).$$

Assim, o filho resultante tem probabilidade uniforme de estar em qualquer ponto da reta que liga os dois parâmetros no hipercubo.

Opcionalmente, podemos simular o efeito de uma mutação $\mathcal{M}: \Omega \mapsto \Omega$, perturbando aleatoriamente as coordenadas do novo indivíduo com uma pequena probabilidade. Na prática, este operador tem o poder de expandir a imagem $\xi(\Psi \times \Psi)$ para um subconjunto maior de Ω, de modo que $\xi(\Psi \times \Psi) \subset \mathcal{M}(\xi(\Psi \times \Psi))$, aumentando a variabilidade genética dos indivíduos.