Redes Neurais Artificiais

Estruturas *Feedforward* e Aprendizado Supervisionado

1. Leituras complementares

- **Páginas WEB:**

  ftp://ftp.sas.com/pub/neural/FAQ.html (comp.ai.neural-nets FAQ)

  http://nips.djvuzone.org/ (Todos os artigos on-line da conferência "Neural Information Processing Systems (NIPS)")

  http://ieeexplore.ieee.org/Xplore (Todas as publicação on-line do IEEE, inclusive de conferências em redes neurais artificiais, como a International Joint Conference on Neural Networks (IJCNN))

- **Periódicos:**

IEEE Transactions on Neural Networks and Learning Systems

Neural Computation (MIT Press)

International Journal of Neural Systems (World Scientific Publishing)

IEEE Transaction on Systems, Man, and Cybernetics (Part B)

Information Sciences (Elsevier)

Learning & Nonlinear Models (SBIC - Brasil)

Journal of Machine Learning Research (JMLR)

Neural Networks (Elsevier)

Neurocomputing (Elsevier)

Biological Cybernetics (Springer)

Neural Processing Letters (Springer)

Cognitive Science (CSS)

Machine Learning (Springer)

- **Livros:**

1. Arbib, M.A. (ed.) (2002) "The Handbook of Brain Theory and Neural Networks", The MIT Press, 2nd. edition, ISBN: 0262011972.
2. Bertsekas, D.P. & Tsitsiklis, J.N. (1996) "Neuro-Dynamic Programming", Athena Scientific, ISBN: 1886529108.
3. Bishop, C.M. (1996) "Neural Networks for Pattern Recognition", Oxford University Press, ISBN: 0198538642.
4. Bishop, C.M. (2007) "Pattern Recognition and Machine Learning", Springer, ISBN: 0387310738.
5. Braga, A.P., de Carvalho, A.P.L.F. & Ludermir, T.B. (2007) "Redes Neurais Artificiais – Teoria e Aplicações", Editora LTC, 2a. edição, ISBN: 9788521615644.
6. Chauvin, Y. & Rumelhart, D.E. (1995) "Backpropagation: Theory, Architectures, and Applications", Lawrence Erlbaum Associates, ISBN: 080581258X.
7. Cherkassky, V. & Mulier, F. (2007) "Learning from Data: Concepts, Theory, and Methods", 2nd edition, Wiley-IEEE Press, ISBN: 0471681822.
8. Cristianini N. & Shawe-Taylor, J. (2000) "An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods", Cambridge University Press, ISBN: 0521780195.
9. da Silva, I.N., Spatti, D.H. & Flauzino, R.A. (2010) "Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas", Artliber Editora Ltda., ISBN: 9788588098534.

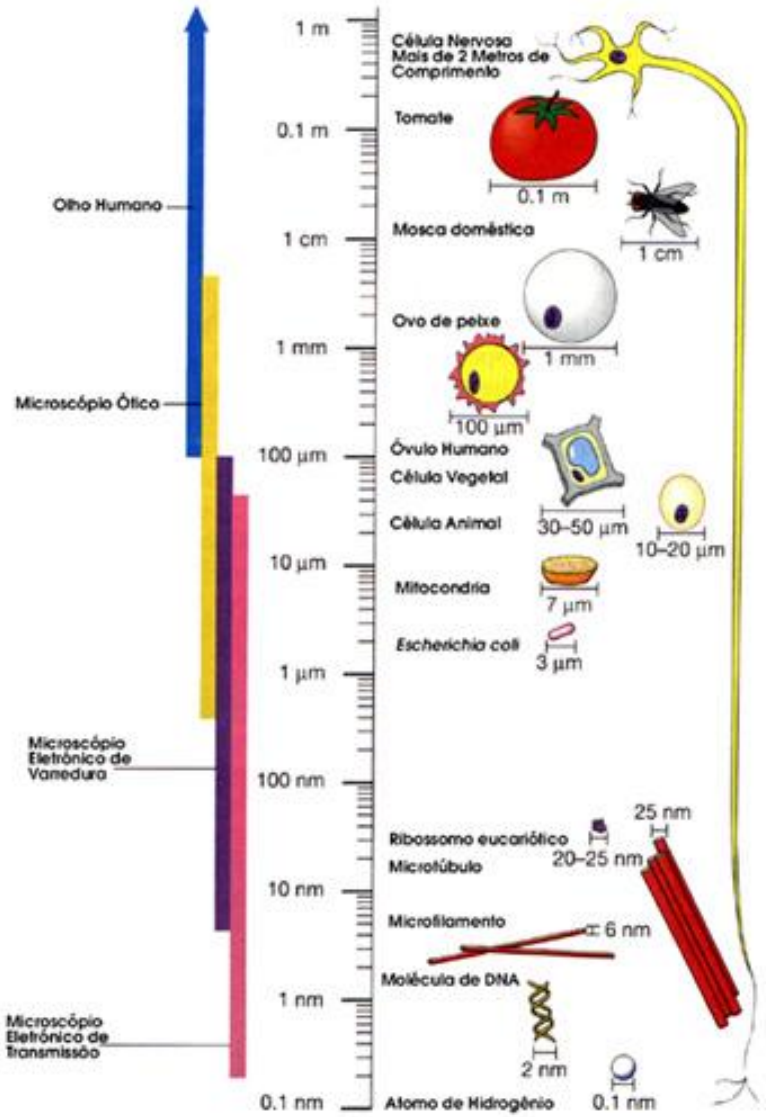
10. Dayan, P. & Abbot, L.F. (2001) "Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems", The MIT Press, ISBN: 0262041995.
11. Duda, R.O., Hart, P.E. & Stork, D.G. (2000) "Pattern Classification", 2nd edition, Wiley-Interscience, ISBN: 0471056693.
12. Edelman, G.M. (1988) "Neural Darwinism: The Theory of Neuronal Group Selection", Basic Books, ISBN: 0465049346.
13. Fausett, L. (2004) "Fundamentals of Neural Networks: Architectures, Algorithms, and Applications", Dorling Kindersley India, ISBN: 8131700534.
14. Fiesler, E. & Beale, R. (1996) "Handbook of Neural Computation", Institute of Physics Publishing, ISBN: 0750303123.
15. Gardner, H. (2011) "Frames of Mind: The Theory of Multiple Intelligences", 3rd edition, BasicBooks, ISBN: 0465024335.
16. Hassoun, M. (2003) "Fundamentals of Artificial Neural Networks", A Bradford Book, ISBN: 0262514672.
17. Hastie, T., Tibshirani, R. & Friedman, J.H. (2001) "The Elements of Statistical Learning", Springer, ISBN: 0387952845.
18. **Haykin, S. (2008) "Neural Networks and Learning Machines", 3rd edition, Prentice Hall, ISBN: 0131471392.**
19. Hecht-Nielsen, R. (1990) "Neurocomputing", Addison-Wesley Publishing Co., ISBN: 0201093553.

20. Hertz, J., Krogh, A. & Palmer, R. (1991) “Introduction to the Theory of Neural Computation”, Addison-Wesley, ISBN: 0201515601.
21. Kearns, M.J., Vazirani, U. (1994) “An Introduction to Computational Learning Theory”, The MIT Press, ISBN: 0262111934.
22. Kohonen, T. (1989) “Self-Organization and Associative Memory”, 3rd edition, Springer-Verlag, ISBN: 0387513876. (1st Edition: 1984; 2nd edition: 1988)
23. Kohonen, T. (2000) “Self-Organizing Maps”, 3rd Edition, Springer, ISBN: 3540679219.
24. Luenberger, D.G. (1984) “Linear and Nonlinear Programming”, 2nd edition, Addison-Wesley, ISBN: 0201157942.
25. Mackay, D.J.C. (2003) “Information Theory, Inference and Learning Algorithms”, Cambridge University Press, ISBN: 0521642981.
26. Mardia, K.V., Kent, J.T., Bibby, J.M. (1980) “Multivariate Analysis”. Academic Press, ISBN: 0124712525.
27. Marsland, S. (2009) “Machine Learning: An Algorithmic Perspective”, Chapman and Hall/CRC, ISBN: 1420067184.
28. Masters, T. (1995) “Advanced Algorithms for Neural Networks: A C++ Sourcebook”, John Wiley and Sons, ISBN: 0471105880.
29. Minsky, M.L. (1988) “The Society of Mind”, Simon & Schuster, ISBN: 0671657135.
30. Minsky, M.L. & Papert, S.A. (1988) “Perceptrons: Introduction to Computational Geometry”, Expanded edition, The MIT Press, ISBN: 0262631113. (1st edition: 1969)
31. Mitchell, T.M. (1997) “Machine Learning”, McGraw-Hill, ISBN: 0071154671.

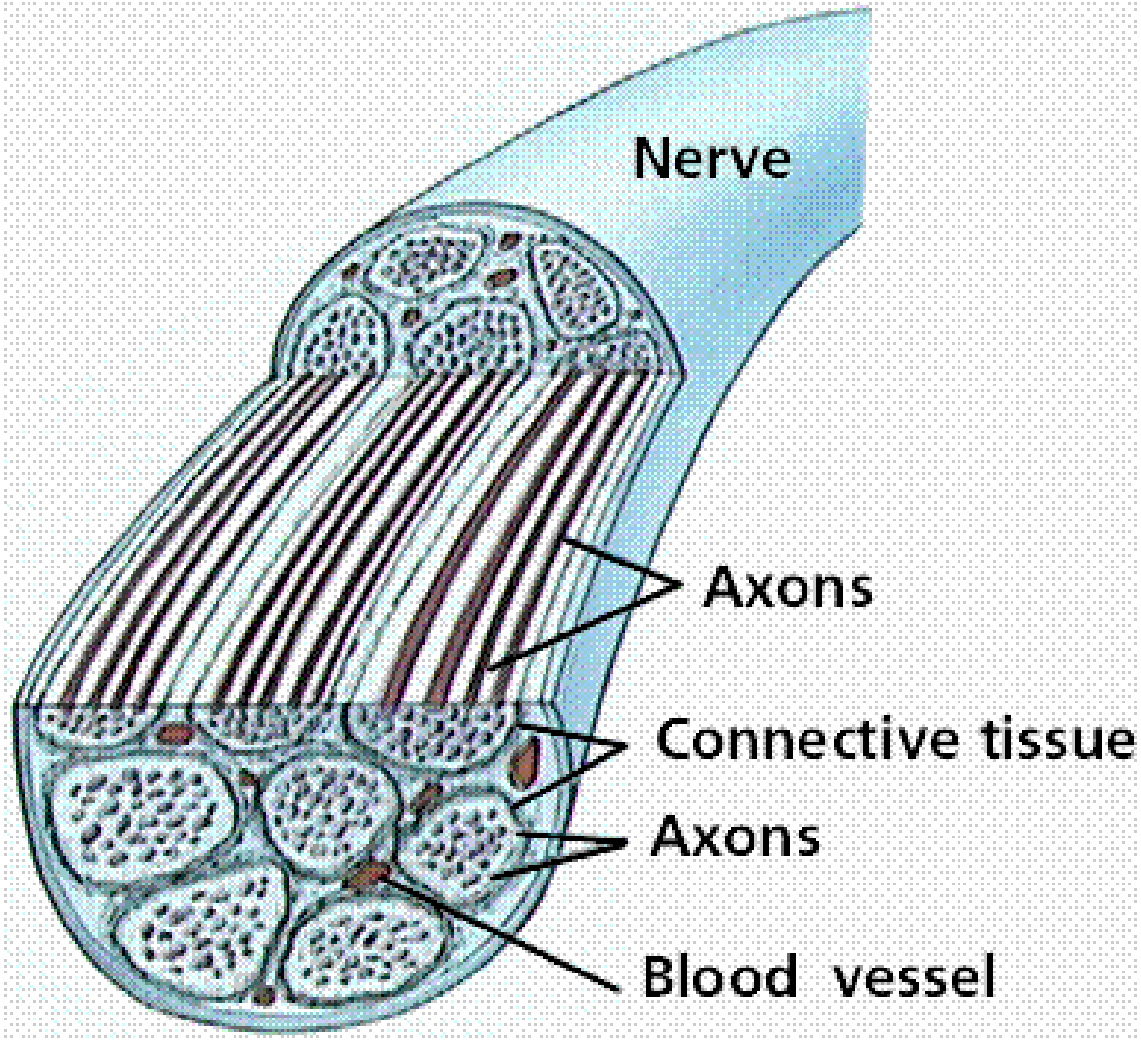
32. Ripley, B.D. (2008) “Pattern Recognition and Neural Networks”, Cambridge University Press, ISBN: 0521717701.
33. Rumelhart, D.E. & McClelland, J.L. (1986) “Parallel Distributed Processing: Explorations in the Microstructure of Cognition”, volumes 1 & 2. The MIT Press, ISBN: 026268053X.
34. Schalkoff, R.J. (1997) “Artificial Neural Networks”, The McGraw-Hill Companies, ISBN: 0071155546.
35. Schölkopf, B. & Smola, A.J. (2001) “Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond”, The MIT Press, ISBN: 0262194759.
36. Sutton, R.S. & Barto, A.G. (1998) “Reinforcement Learning: An Introduction”, The MIT Press, ISBN: 0262193981.
37. Vapnik V.N. (1998) “Statistical Learning Theory”, Wiley-Interscience, ISBN: 0471030031.
38. Vapnik V.N. (1999) “The Nature of Statistical Learning Theory”, 2nd edition, Springer, ISBN: 0387987800.
39. Weigend, A.S. & Gershenfeld, N.A. (eds.) (1993) “Time Series Prediction: Forecasting the Future and Understanding the Past”, Perseus Press, ISBN: 0201626020.
40. Wilson, R.A. & Keil, F.C. (eds.) (2001) “The MIT Encyclopedia of the Cognitive Sciences”, The MIT Press, ISBN: 0262731444.

2. Números e nomenclatura referentes ao sistema nervoso

- Descoberta do microscópio: ~1590
- Descoberta da célula: ~1680
- Célula como unidade constituinte dos seres vivos: ~1830
- Constituintes básicos do cérebro são os neurônios: Ramón y Cajál, ~1909
- O cérebro humano pesa ~1,5 quilos e consome ~20% da energia do corpo;
- 100 gramas de tecido cerebral requerem ~3,5ml de oxigênio por minuto;
- O cérebro humano apresenta ~10^{11} neurônios e ~10^{14} sinapses ou conexões, com uma média de ~1000 conexões por neurônio, podendo chegar a ~10000 conexões.
- Em seres humanos, 70% dos neurônios estão localizados no córtex;
- Tipos de células neurais: horizontal, estrelada, piramidal, granular, fusiforme.
- Classificação de acordo com a função: sensoriais, motoras, intrínsecas.



- O diâmetro do corpo celular de um neurônio mede de ~5µm (célula granular) a ~60µm (célula piramidal);
- Em termos fisiológicos, um neurônio é uma célula com a função específica de receber, processar e enviar informação a outras partes do organismo.
- Um nervo é formado por um feixe de axônios, com cada axônio associado a um único neurônio;
- Os nervos apresentam comprimentos variados, podendo chegar a metros.



Estrutura de um nervo

- A estrutura e as funcionalidades do cérebro são governadas por princípios básicos de alocação de recursos e otimização sujeita a restrições.

LAUGHLIN, S.B. & SEJNOWSKI, T.J. (2003) "Communication in neuronal networks", Science, vol. 301, no. 5641, pp. 1870–1874.

- O ser humano pode reagir simultaneamente a uma quantidade bem limitada de estímulos, o que pode indicar que mecanismos de alocação de recursos (e.g. glicose, oxigênio) baseados em prioridades são implementados no cérebro.

NORMAN, D.A. & BOBROW, D.G. (1975) "On data-limited and resource-limited processes", Cognitive Psychology, vol. 7, pp. 44-64.

- Alguns autores defendem que o córtex humano pode ser modelado na forma de uma rede "mundo pequeno" (BASSETT & BULLMORE, 2006; SPORNS & HONEY, 2006; SPORNS, 2010) ou então uma rede complexa (AMARAL & OTTINO, 2004).

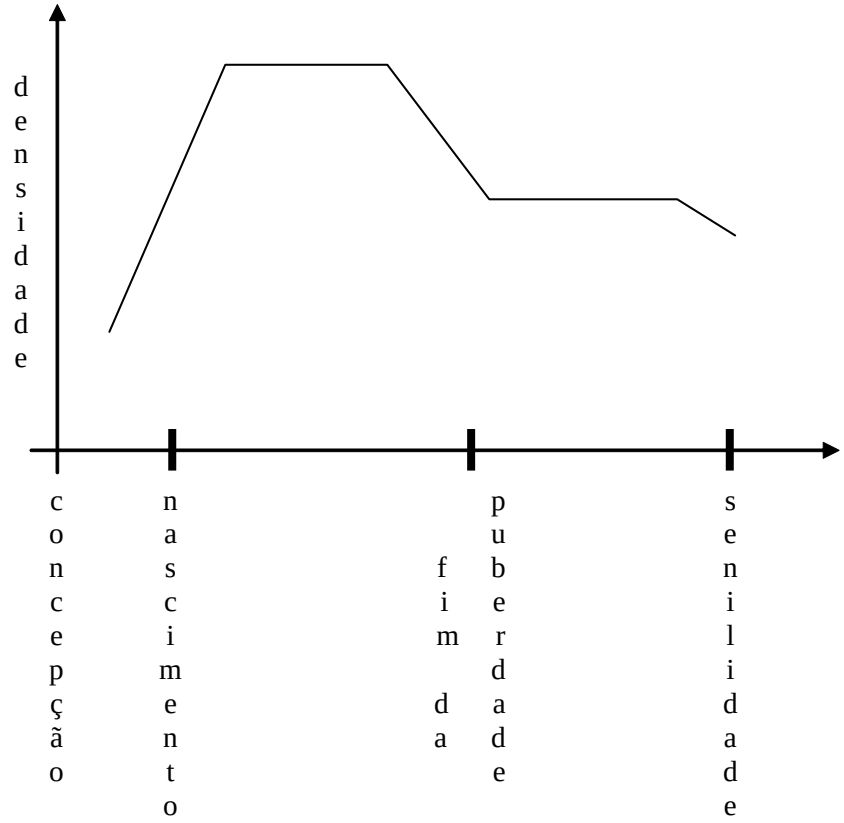
AMARAL, L. & OTTINO, J. (2004) "Complex networks", The European Physical Journal B – Condensed Matter and Complex Systems, vol. 38, no. 2, pp. 147-162.
BASSETT, D.S. & BULLMORE, E. (2006) "Small-world brain networks", Neuroscientist, vol. 12, no. 6, pp. 512-523.
SPORNS, O. & HONEY, C.J. (2006) "Small worlds inside big brains", Proceedings of the National Academy of Science, vol. 103, no. 51, pp. 19219-19220.
SPORNS, O. (2010) "Networks of the Brain", The MIT Press, ISBN: 0262014696.

- Há um expressivo aumento na densidade de conexões sinápticas da vida embrionária até a idade de 2 anos. Quando se atinge a idade de 2 anos, o ser humano apresenta a maior concentração de sinapses, a qual se mantém num nível elevado até o início da puberdade. Até o término da puberdade, há uma queda acentuada no número de sinapses.

- Esse processo de ampliação e redução de sinapses, contudo, não é homogêneo, pois nas regiões sensório-motoras este processo ocorre mais cedo, enquanto que ele é retardado em áreas associadas aos processos cognitivos.

- A redução de sinapses é dramática: o número de sinapses ao término da puberdade pode chegar a 50% do número existente com a idade de 2 anos. Há uma perda de até 100.000 sinapses por segundo na adolescência.

KOLB, B & WHISHAW, I.Q. (2008) "Fundamentals of Human Neuropsychology", Worth Publishers, 6th. edition, ISBN: 0716795868.



Evolução da densidade de sinapses ao longo da vida de um ser humano

- Acredita-se ser impossível que o código genético de um indivíduo seja capaz de conduzir todo o processo de organização topológica do cérebro. Apenas aspectos gerais dos circuitos envolvidos devem estar codificados geneticamente.

- Logo, para explicar as conformações sinápticas, recorre-se a dois mecanismos gerais de perda de sinapses: *experience expectant* e *experience dependent* (KOLB & WHISHAW, 2008).
- Podas baseadas em *experience expectant* estão vinculadas à experiência sensorial para a organização das sinapses. Geralmente, os padrões sinápticos são os mesmos para membros de uma mesma espécie. Exemplo: A formação de sinapses no córtex visual depende da exposição a atributos como linha de orientação, cor e movimento.
- Podas baseadas em *experience dependent* estão vinculadas a experiências pessoais únicas, tal como falar uma língua distinta. Com isso, defende-se que o padrão de conexões do lobo frontal seja formado por podas baseadas em *experience dependent*.
- De fato, a atividade do córtex pré-frontal tende a ser até 4 vezes mais intensa em crianças do que em adultos, o que permite concluir que poda de parte das conexões e fortalecimento de outras contribuem para a maturação cognitiva.

CASEY, B.J., TOTTENHAM, N., LISTON, C. & DURSTON, S. (2005) "Imaging the developing brain: what have we learned about cognitive development?", Trends in Cognitive Science, vol. 9, no. 3, pp. 104-110.

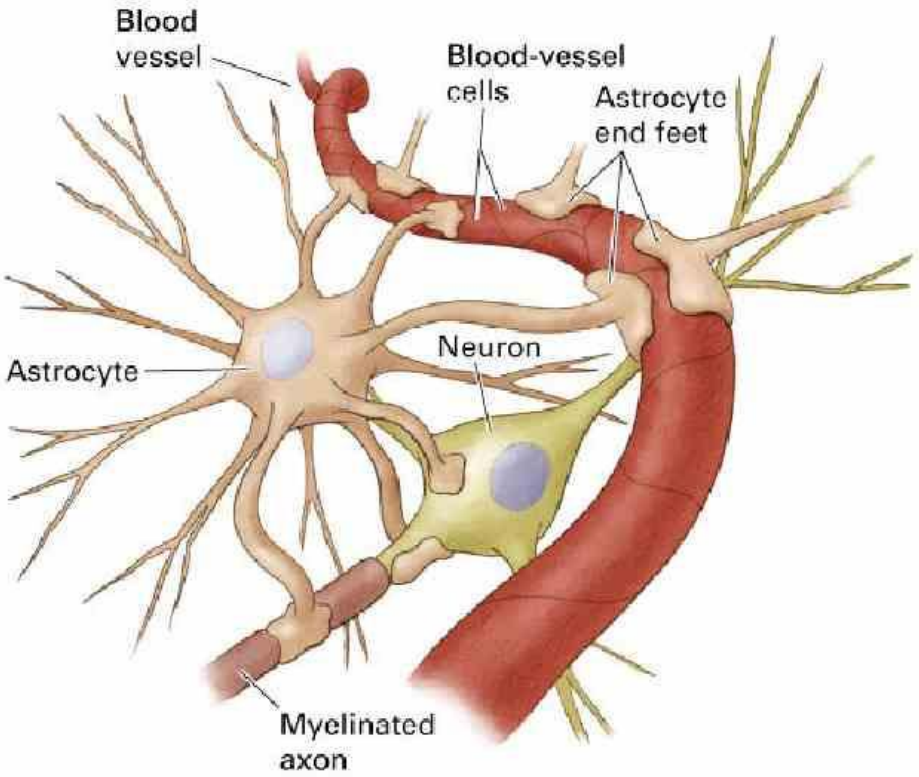
- Em síntese, é possível afirmar que o padrão de conexões no cérebro se inicia sem muita organização e com uma grande densidade de sinapses. Com a experiência de vida, um equilíbrio é atingido. Logo, como o padrão de conexões de um ser humano adulto é obtido a partir da experiência de vida, cada pessoa vai apresentar um padrão de conexões diferente, particularmente nas áreas especializadas em cognição.
- Por outro lado, o sistema sensório-motor em um adulto normal deve apresentar uma conformação similar a de outros adultos normais, visto que a poda nessas áreas é *experience expectant*.

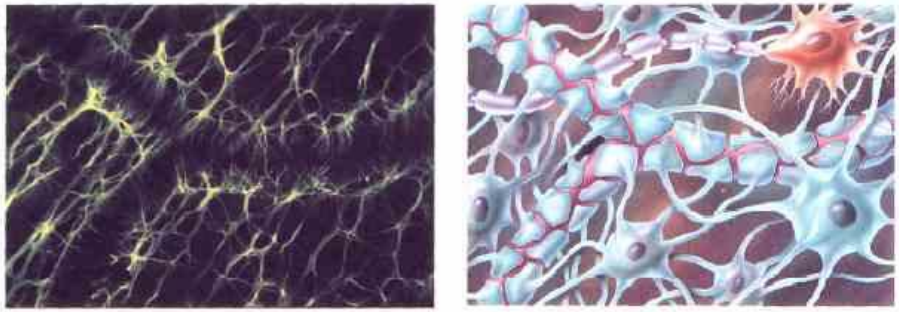
FRANCO, A.R. (2009) "Resource Allocation of the human brain: a competitive equilibrium approach", Ph. D. Thesis, The University of New Mexico, Albuquerque, New Mexico, USA.

- Voltando agora a atenção para o neurônio biológico, pode-se afirmar que se trata de uma célula especializada em transmitir pulsos elétricos, sendo que as suas principais partes constituintes são:
  - ✓ Membrana celular: é a "pele" da célula;
  - ✓ Citoplasma: tudo que está envolvido pela membrana;

- ✓ Núcleo: contém os cromossomos (DNA);
- ✓ Ribossomos: geram proteínas a partir de mRNAs;
- ✓ Mitocôndria: gera energia para a célula (produz ATP);
- ✓ Soma: corpo celular, excluindo dendritos e axônio;
- ✓ Dendritos: parte do neurônio que recebe informação de outros neurônios;
- ✓ Axônio: parte do neurônio que transmite informação para outros neurônios;
- ✓ Bainha de mielina: revestimento externo lipídico do axônio, responsável por evitar a dispersão dos sinais elétricos, como uma capa isolante;
- ✓ Terminais pré-sinápticos: área do neurônio que armazena neurotransmissores, os quais são liberados por potenciais de ação.

- Os neurônios sensoriais normalmente têm longos dendritos e axônios curtos. Por outro lado, os neurônios motores têm um longo axônio e dendritos curtos (transmitem informação para músculos e glândulas). Já os neurônios intrínsecos ou interneurônios realizam a comunicação neurônio-a-neurônio e compõem o sistema nervoso central.

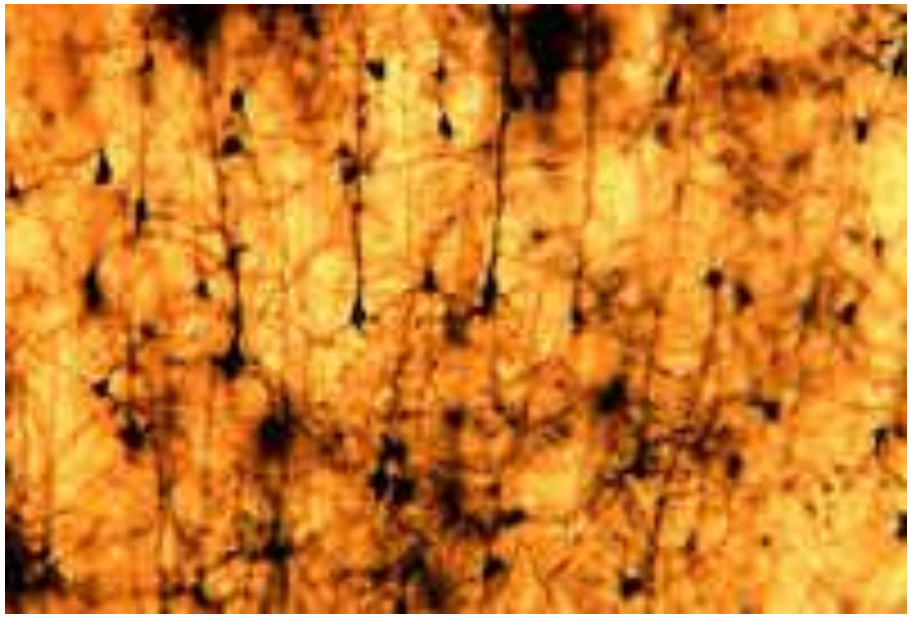




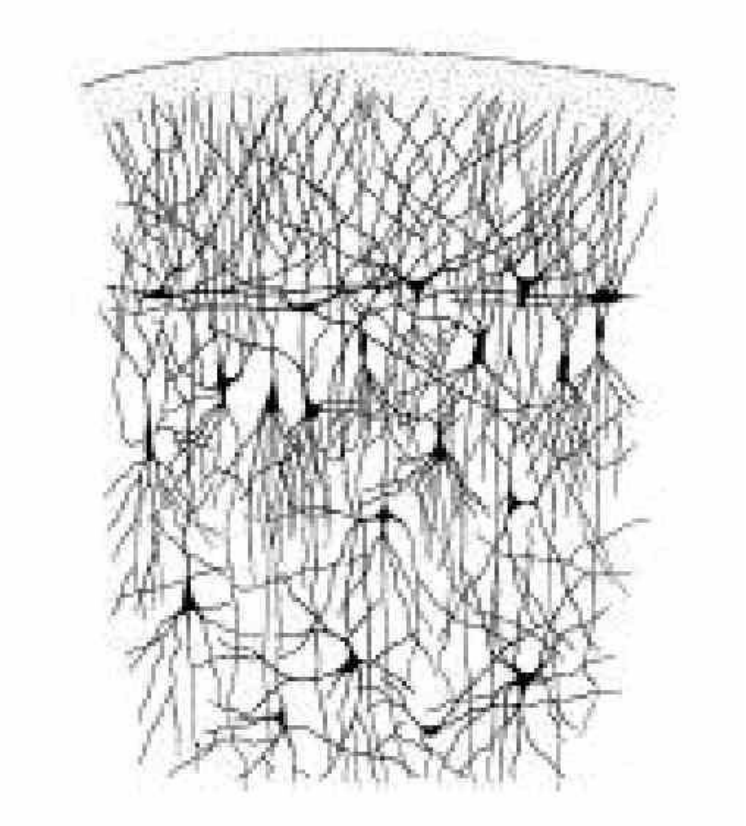
Brain photo

Cartoon reconstruction

- Além das células condutoras, o cérebro possui as células não-condutoras, formando a glia (neuróglia).
- Os astrócitos se caracterizam pela riqueza e dimensões de seus prolongamentos cito-plasmáticos, distribuídos em todas as direções. Funções: prover suporte estrutural, nutrientes e regulação química.
- Máxima distância de um neurônio a um vaso sanguíneo: ~50μm.



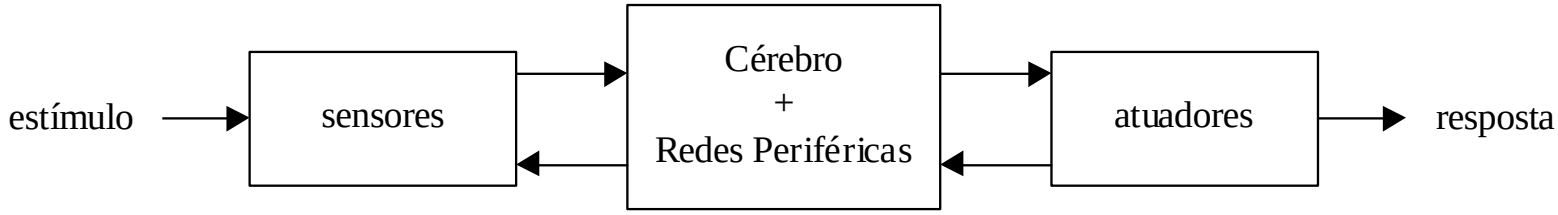
Rede de neurônios piramidais do córtex de um hamster



Desenho do córtex realizado por RAMÓN Y CAJÁL (1909)

- Hemisfério esquerdo: aprendizado verbal, memória verbal, processamento serial, processamento simbólico, linguagem, inferência, noção de tempo.
- Hemisfério direito: aprendizado espacial, memória espacial, processamento global (imagens), síntese da percepção, pensamento associativo, coordenação motora.
- O cérebro é capaz de perceber regularidades no meio e gerar abstrações que capturam a estrutura destas regularidades, possibilitando a predição de observações futuras e o planejamento de ações visando o atendimento de múltiplos objetivos.

- Organização básica do sistema nervoso (Visão de engenharia)



3. Alguns fatos históricos relevantes

- "Idade da Ilusão"
  - MCCULLOCH & PITTS (1943)
  - WIENER (1948): cibernética
  - MINSKY & PAPPERT (1969): a disputa entre as portas lógicas e os neurônios artificiais para determinar a unidade básica de processamento.
- "Idade das Trevas"
  - Entre 1969 e 1984, houve muito pouca pesquisa científica envolvendo redes neurais artificiais
- "Renascimento"
  - HOPFIELD (1982)
  - RUMELHART & MCCLELLAND (1986)
  - Desenvolvimento da capacidade de processamento e memória dos computadores digitais (simulação computacional / máquina virtual) (anos 80 e 90)

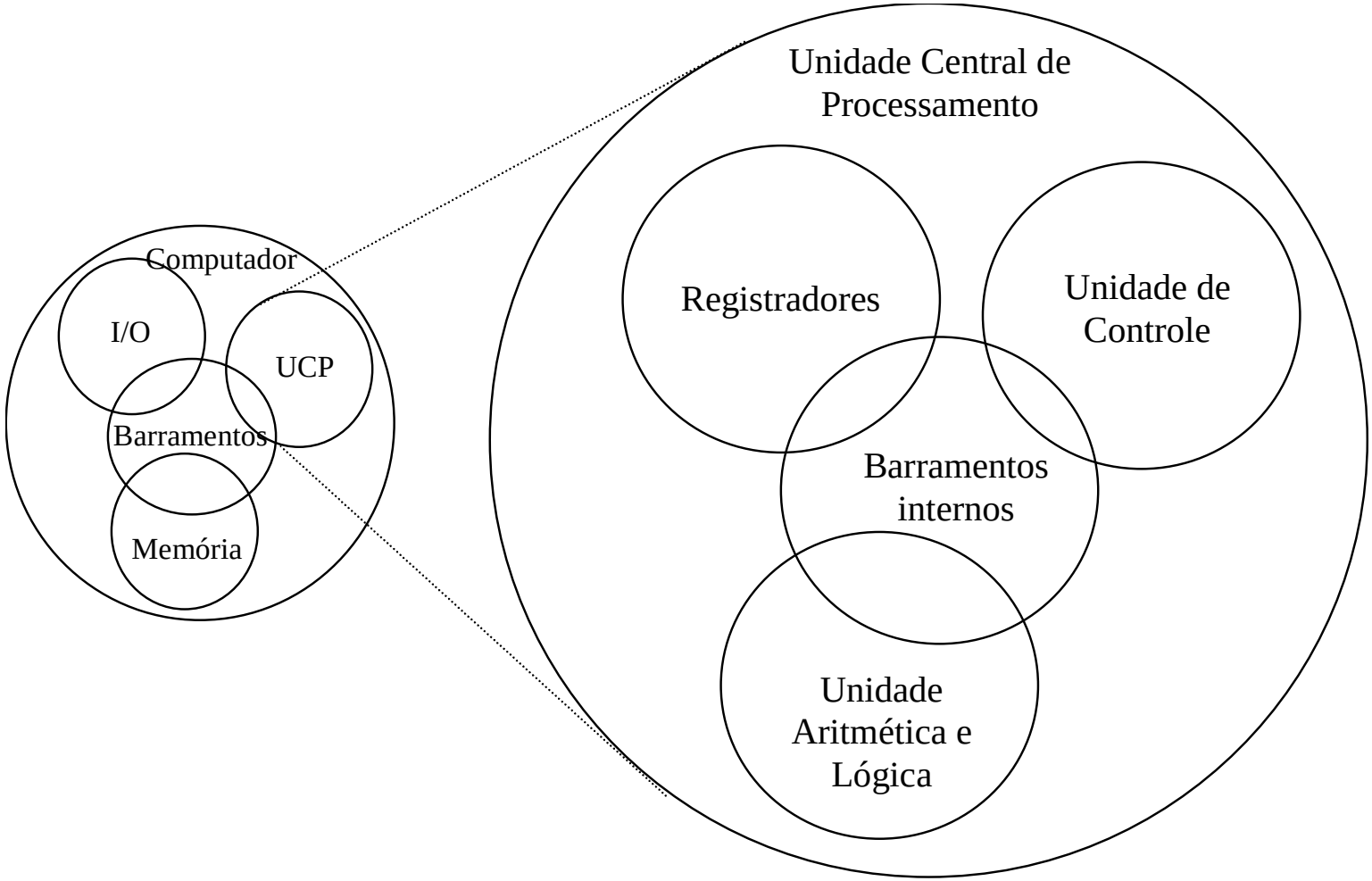
- GARDNER (1983): múltiplas inteligências

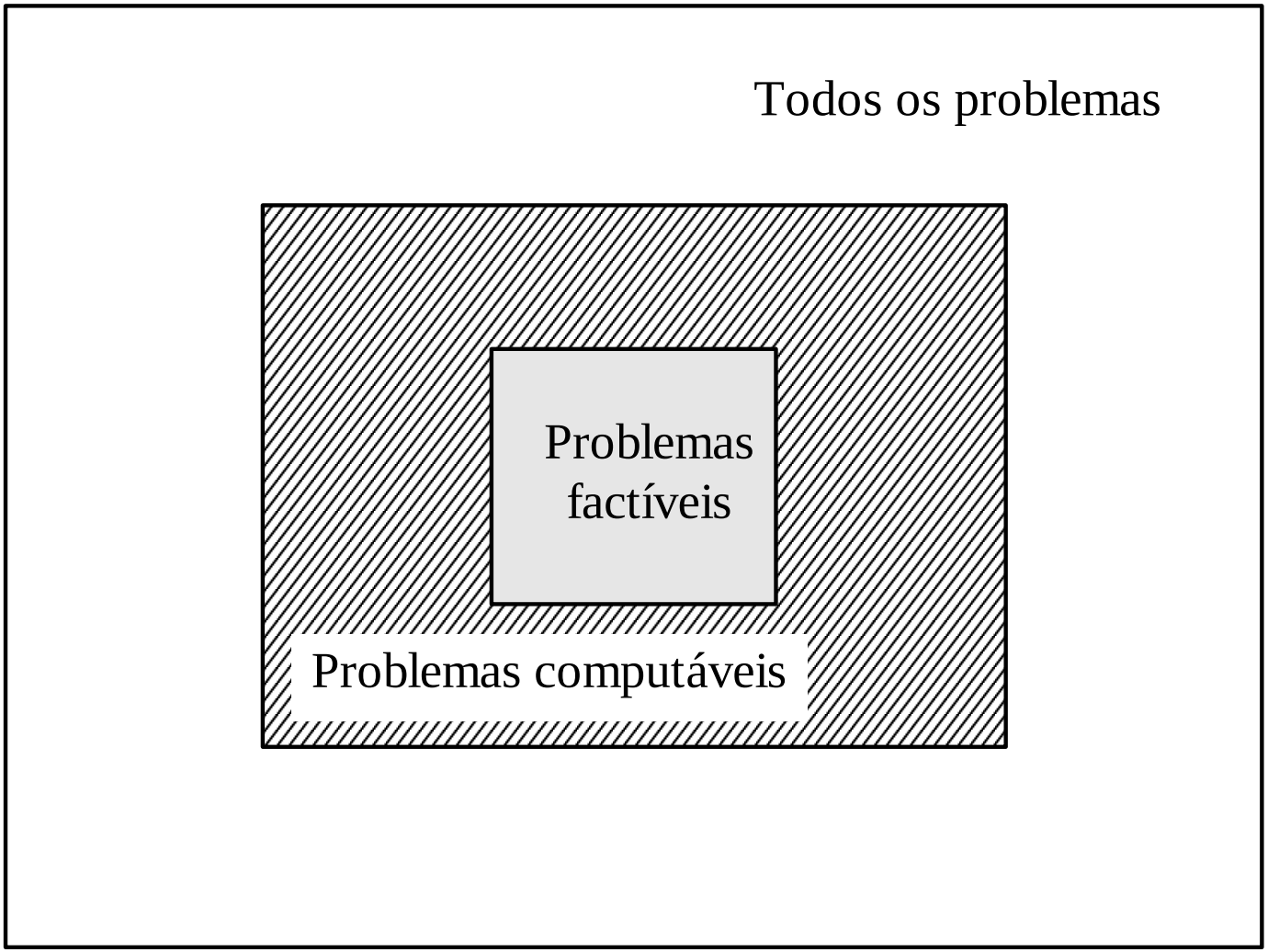
  1. Vivacidade verbal
  2. Vivacidade matemático-lógica
  3. Aptidão espacial
  4. Gênio cinestésico
  5. Dons musicais
  6. Aptidão interpessoal (liderança e ação cooperativa)
  7. Aptidão intrapsíquica (modelo preciso de si mesmo)

4. Computação digital × Neurocomputação (Conexionismo)

- É muito comum a associação do conceito de computação com aquele predominante no caso dos computadores com arquitetura do tipo Von Neumann: algoritmos são elaborados e em seguida implementados na forma de programas de computador a serem executados. No entanto, a computação realizada pelo cérebro requer um outro tipo de definição, que contemple processamento paralelo e distribuído, além de aprendizado.

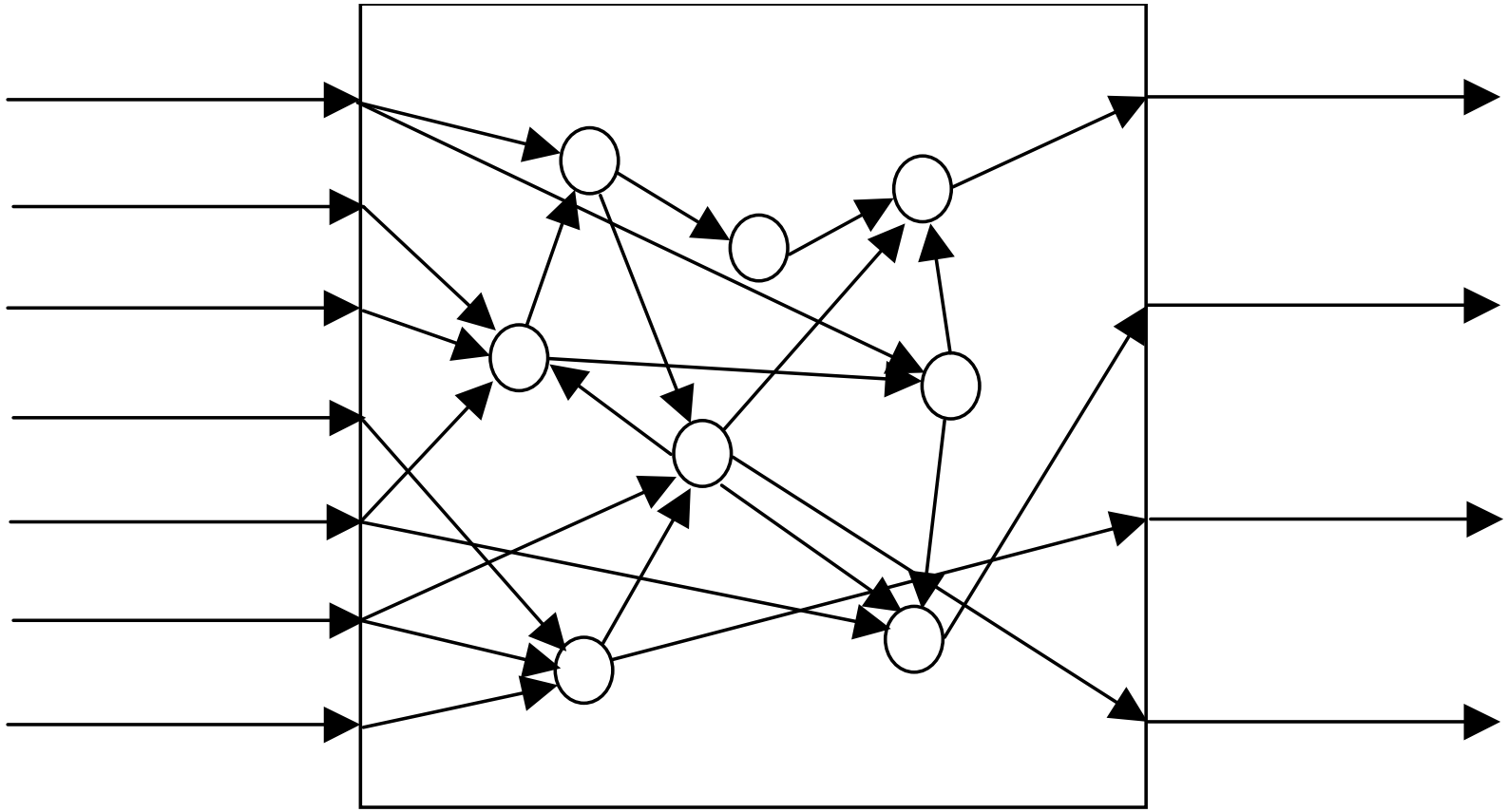
- Uma arquitetura neurocomputacional é baseada na interconexão de unidades de processamento simples e similares, denominadas neurônios artificiais e dotadas de grande poder de adaptação.
- Há uma diferença de paradigmas entre computadores com arquitetura do tipo Von Neumann e redes neurais artificiais (RNAs): os primeiros realizam processamento e armazenagem de dados em dispositivos fisicamente distintos, enquanto RNAs usam o mesmo dispositivo físico para tal.
- A motivação que está por trás deste paradigma alternativo de processamento computacional é a possibilidade de elaborar mecanismos distintos de solução para problemas intratáveis ou ainda não-resolvidos com base na computação convencional, além de criar condições para reproduzir habilidades cognitivas e de processamento de informação muito desejadas em aplicações de engenharia.
- Paradigmas *bottom-up* (Ex: Inteligência computacional) × Paradigmas *top-down* (Ex: Sistemas especialistas)


Computador
I/O
UCP
Barramentos
Memória
Unidade Central de Processamento
Registradores
Unidade de Controle
Barramentos internos
Unidade Aritmética e Lógica



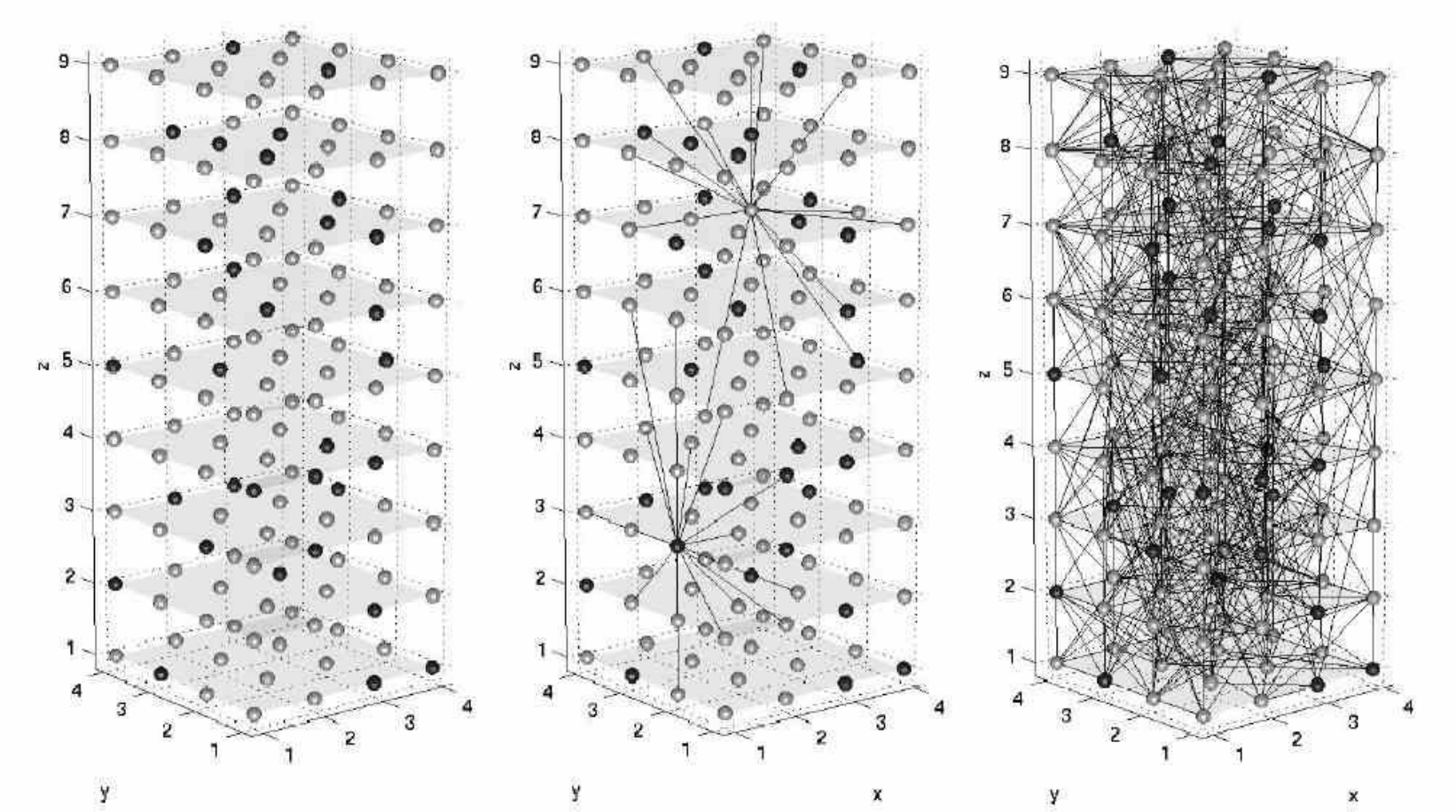
Cenário desafiador para a computação digital

Como abordar os problemas na região hachurada?



Exemplo genérico de um neurocomputador

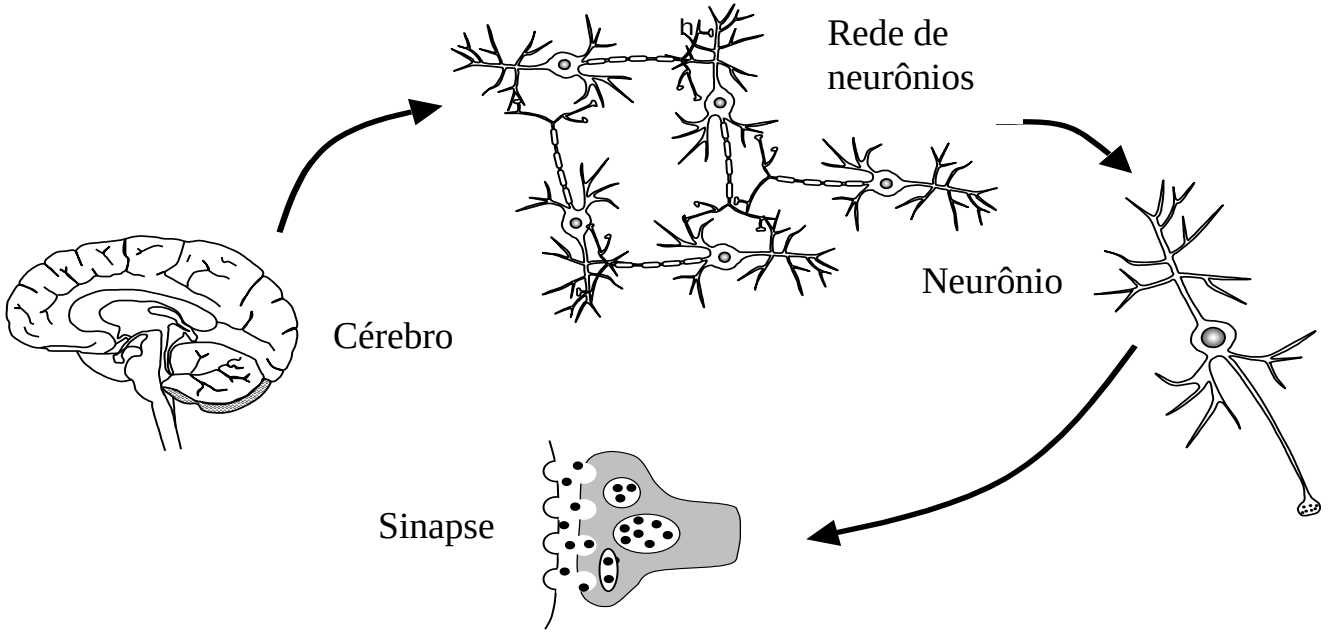
- Rede neural × Rede neuronal × Rede neurônica × Rede neuronial



Rede = nós + conexões (paradigma conexionista)

5. Níveis de Organização no Sistema Nervoso

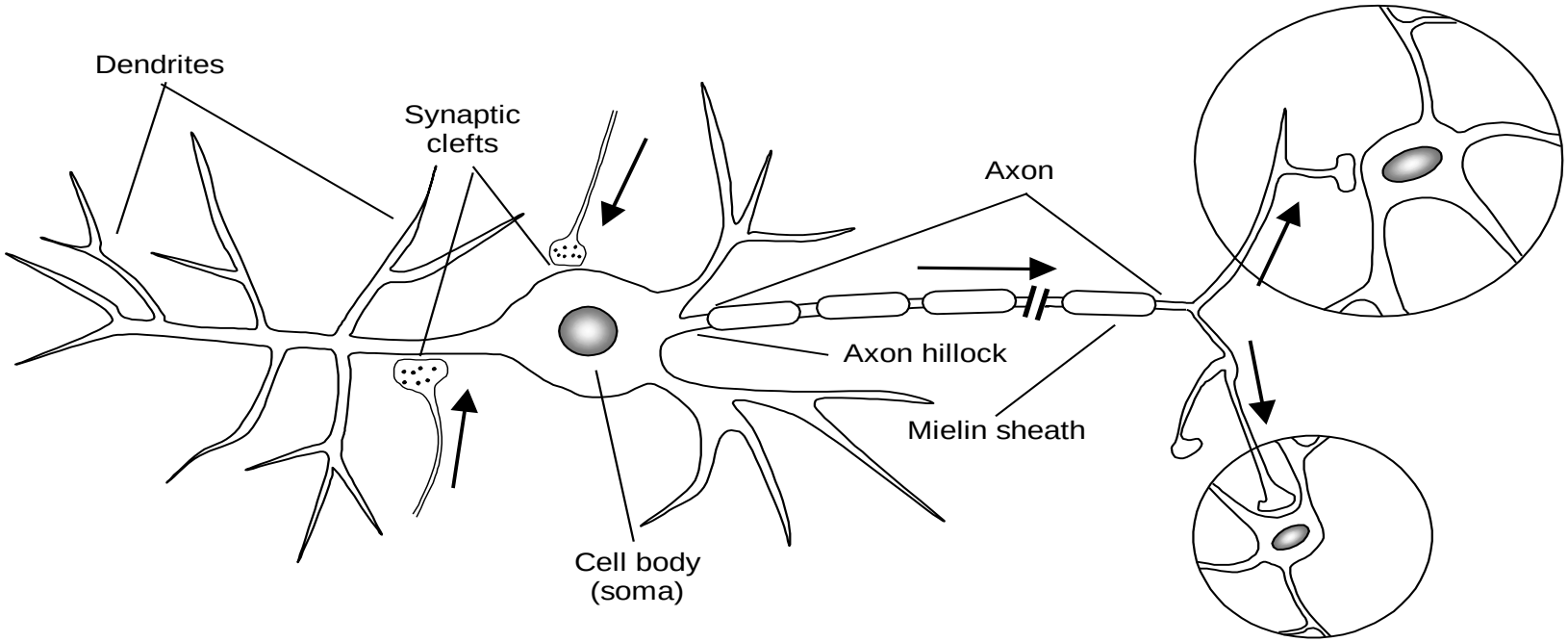
- O sistema nervoso pode ser organizado em diferentes níveis: *moléculas*, *sinapses*, *neurônios*, *camadas*, *mapas* e *sistemas*.
- Uma estrutura facilmente identificável no sistema nervoso é o neurônio, especialista em processamento de sinais.
- Dependendo das condições de operação, os neurônios são capazes de gerar um *sinal*, mais especificamente um *potencial elétrico*, que é utilizado para transmitir informação a outras células.



5.1 Neurônios e Sinapses

- Os neurônios utilizam uma variedade de mecanismos bioquímicos para o processamento e transmissão de informação, incluindo os *canais iônicos*.
- Os canais iônicos permitem um fluxo contínuo de entrada e saída de *correntes (elétricas)*, a liberação de *neurotransmissores* e a geração e propagação de *potenciais de ação*.
- O processo de transmissão de sinais entre neurônios é fundamental para a capacidade de processamento de informação do cérebro.
- Uma das descobertas mais relevantes em neurociência foi a de que a *efetividade* da transmissão de sinais pode ser modulada, permitindo que o cérebro se adapte a diferentes situações.
- A *plasticidade sináptica*, ou seja, a capacidade de as sinapses sofrerem modificações, é o ingrediente-chave para o aprendizado da maioria das RNAs.
- Os neurônios podem receber e enviar sinais de/para vários outros neurônios.

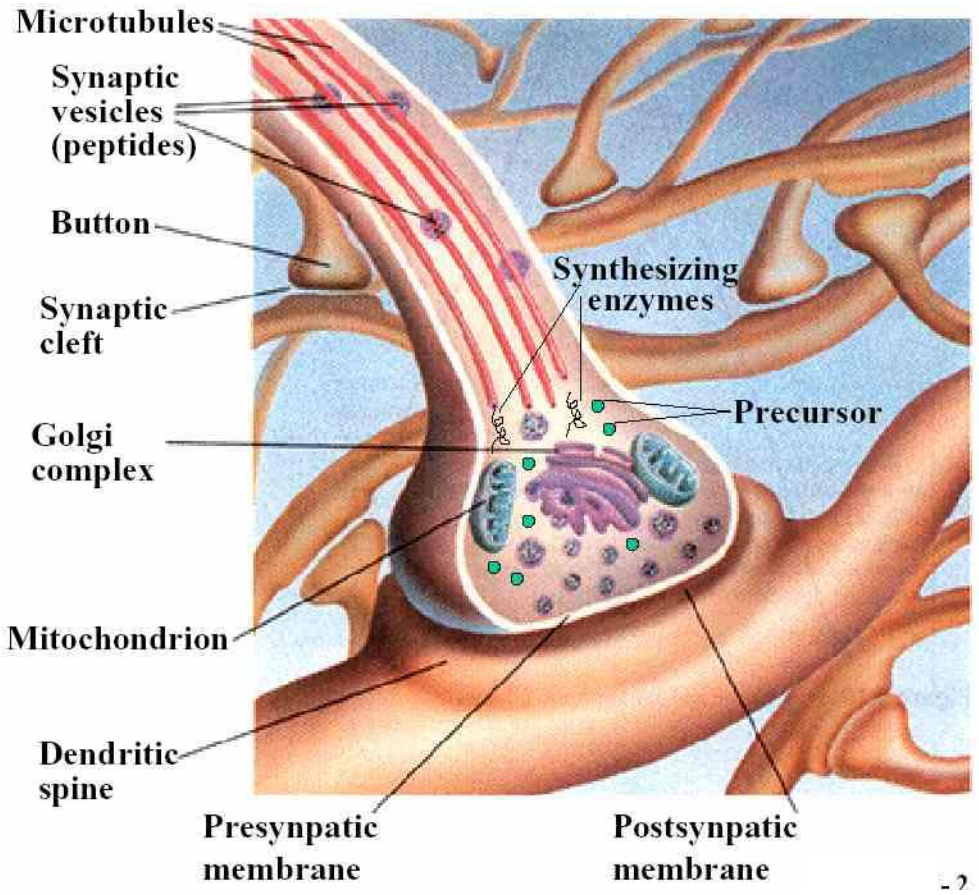
- Os neurônios que enviam sinais, chamados de neurônios *pré-sinápticos* ou "*enviadores*", fazem contato com os neurônios *receptores* ou *pós-sinápticos* em regiões especializadas, denominadas de *sinapses*.



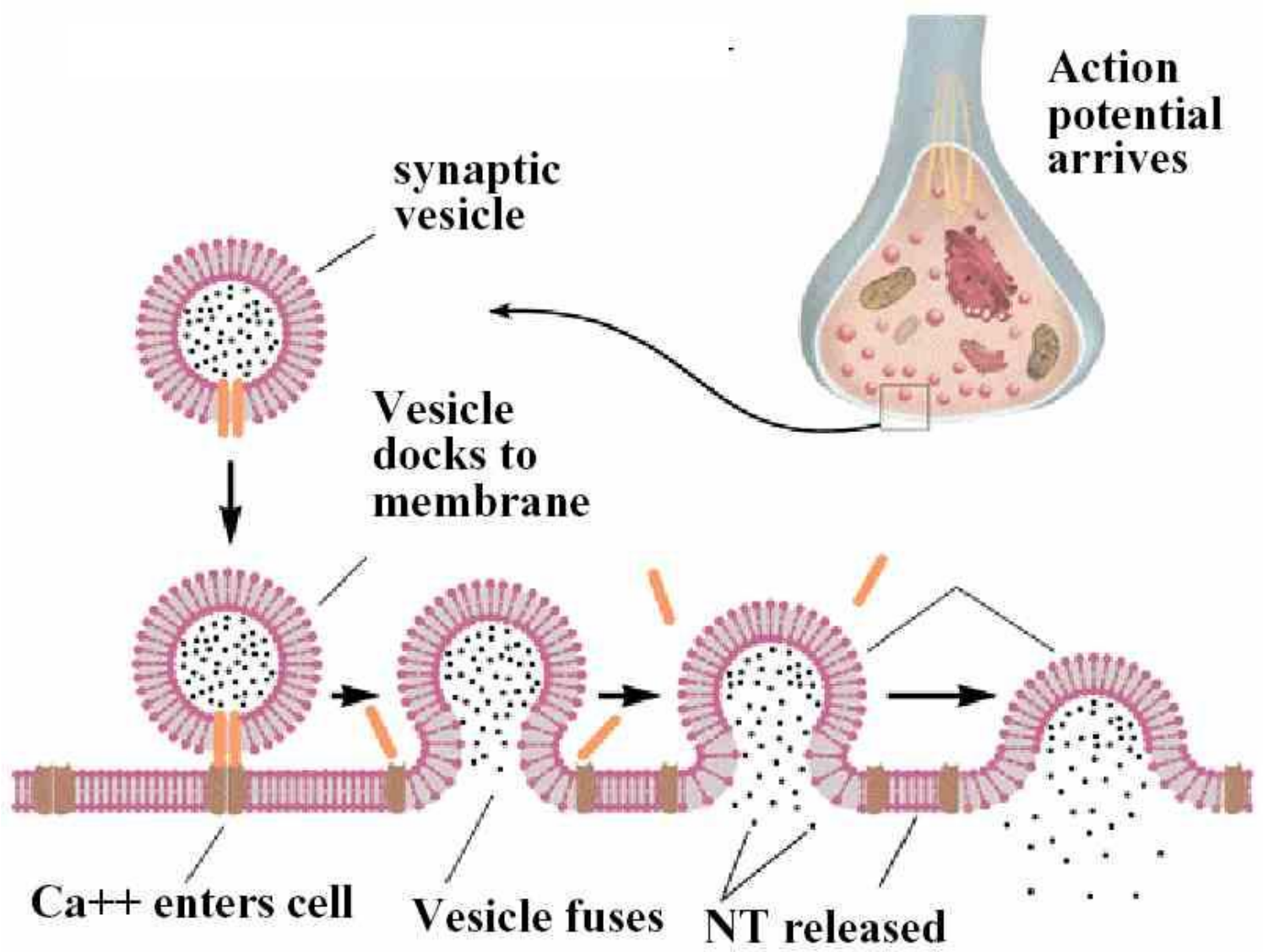
- A sinapse é, portanto, a junção entre o axônio de um neurônio pré-sináptico e o dendrito ou corpo celular de um neurônio pós-sináptico.

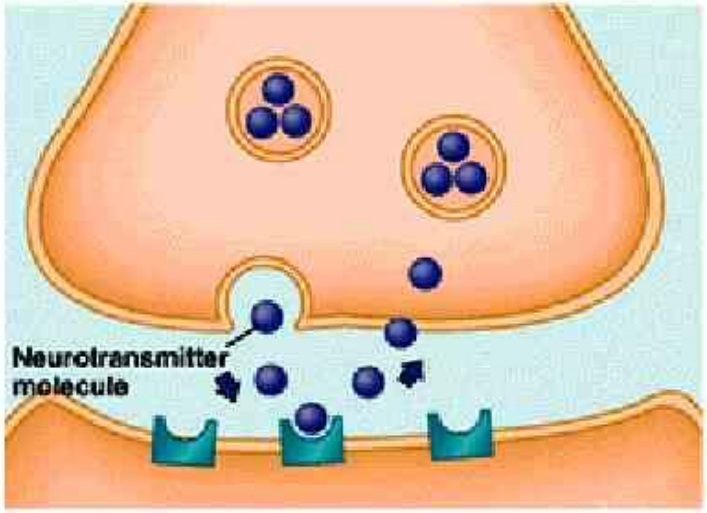
- A capacidade de processamento de informação das sinapses permite que elas alterem o estado de um neurônio pós-sináptico, eventualmente gerando um pulso elétrico, denominado *potencial de ação*, no neurônio pós-sináptico.
- Logo, um neurônio pode ser visto como um dispositivo capaz de receber estímulos (sinais de entrada) de diversos outros neurônios e propagar sua única saída, função dos estímulos recebidos e do estado interno, a vários outros neurônios.
- Existem diversos mecanismos envolvidos na transmissão de informação (sinais) entre neurônios. Como os neurônios são células encapsuladas por membranas, pequenas aberturas nestas membranas (*canais*) permitem a transferência de informação entre eles.
- Os mecanismos básicos de processamento de informação são baseados no movimento de átomos carregados, ou *íons*:
  - ✓ Os neurônios habitam um ambiente líquido contendo uma certa concentração de íons, que podem entrar ou sair do neurônio através dos canais.

- ✓ Um neurônio é capaz de alterar o potencial elétrico de outros neurônios, denominado de *potencial de membrana*, que é dado pela diferença do potencial elétrico dentro e fora do neurônio.
- ✓ Quando um potencial de ação chega ao final do axônio, ele promove a liberação de neurotransmissores (substâncias químicas) na fenda sináptica, os quais se difundem e se ligam a receptores no neurônio pós-sináptico.
- ✓ Essa ligação entre neurotransmissores e receptores conduz à abertura dos canais iônicos, permitindo a entrada de íons na célula. A diferença de potencial resultante apresenta a forma de um pulso elétrico.
- ✓ Esses pulsos elétricos se propagam pelo neurônio pós-sináptico e são integrados no corpo celular. A ativação do neurônio pós-sináptico irá se dar no caso de o efeito resultante destes pulsos elétricos integrados ultrapassar um dado *limiar*.
- ✓ Alguns neurotransmissores possuem a capacidade de *ativar* um neurônio enquanto outros possuem a capacidade de *inibir* a ativação do neurônio.

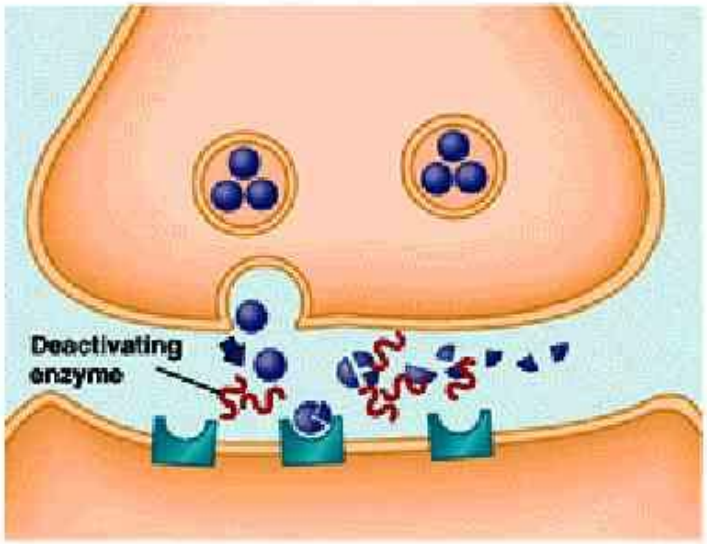


- A sinapse é uma fenda entre os terminais pré-sináptico e pós-sináptico, medindo ~20 nm.

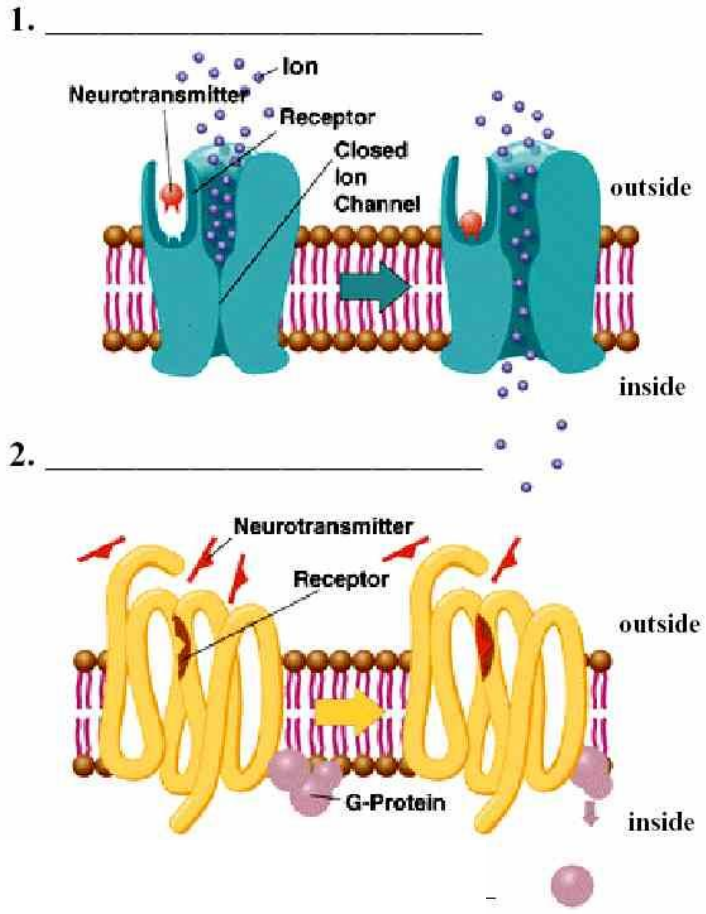

Action potential arrives
synaptic vesicle
Vesicle docks to membrane
Ca++ enters cell
Vesicle fuses
NT released



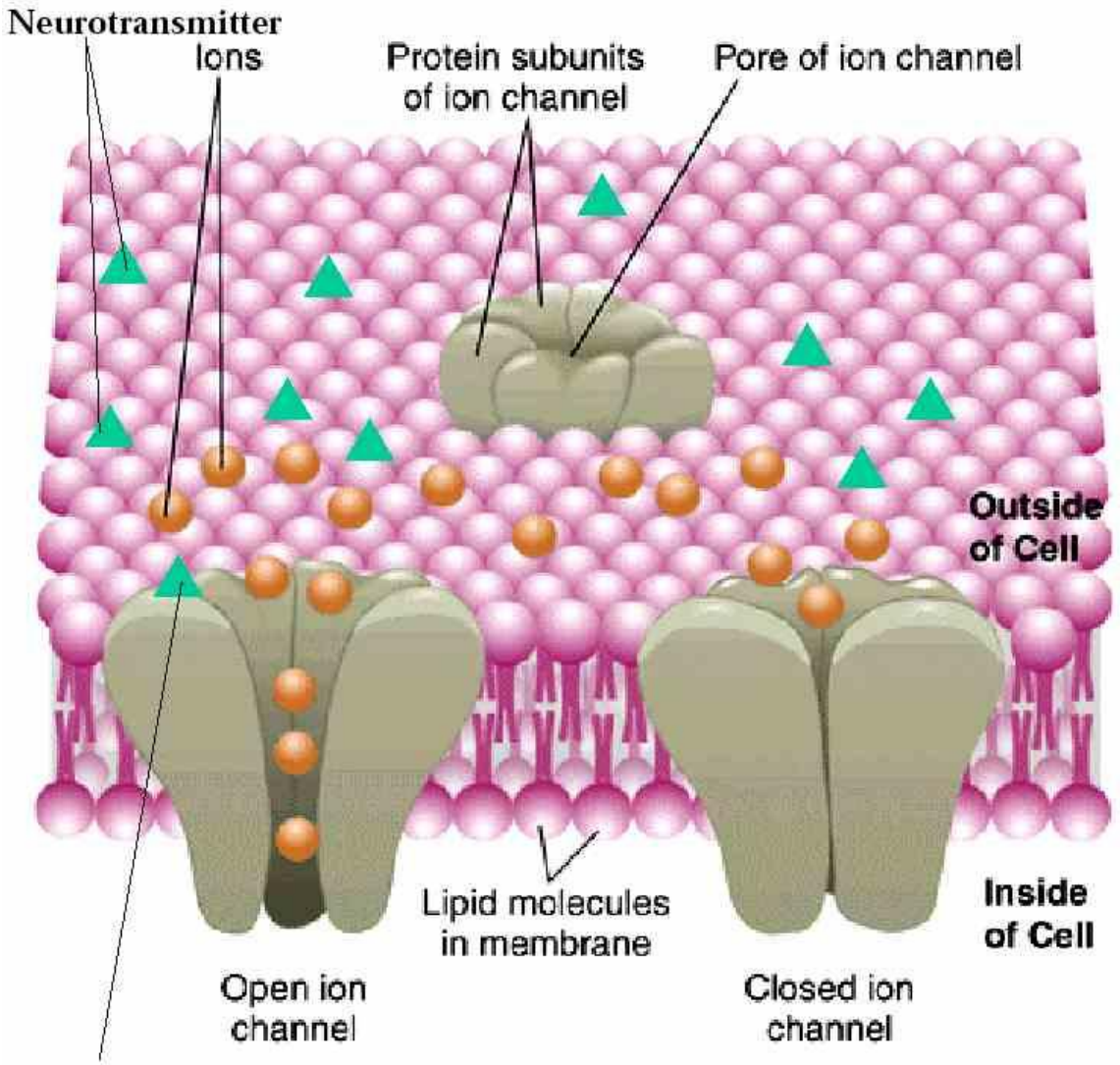
Reuptake



Deactivating Enzymes

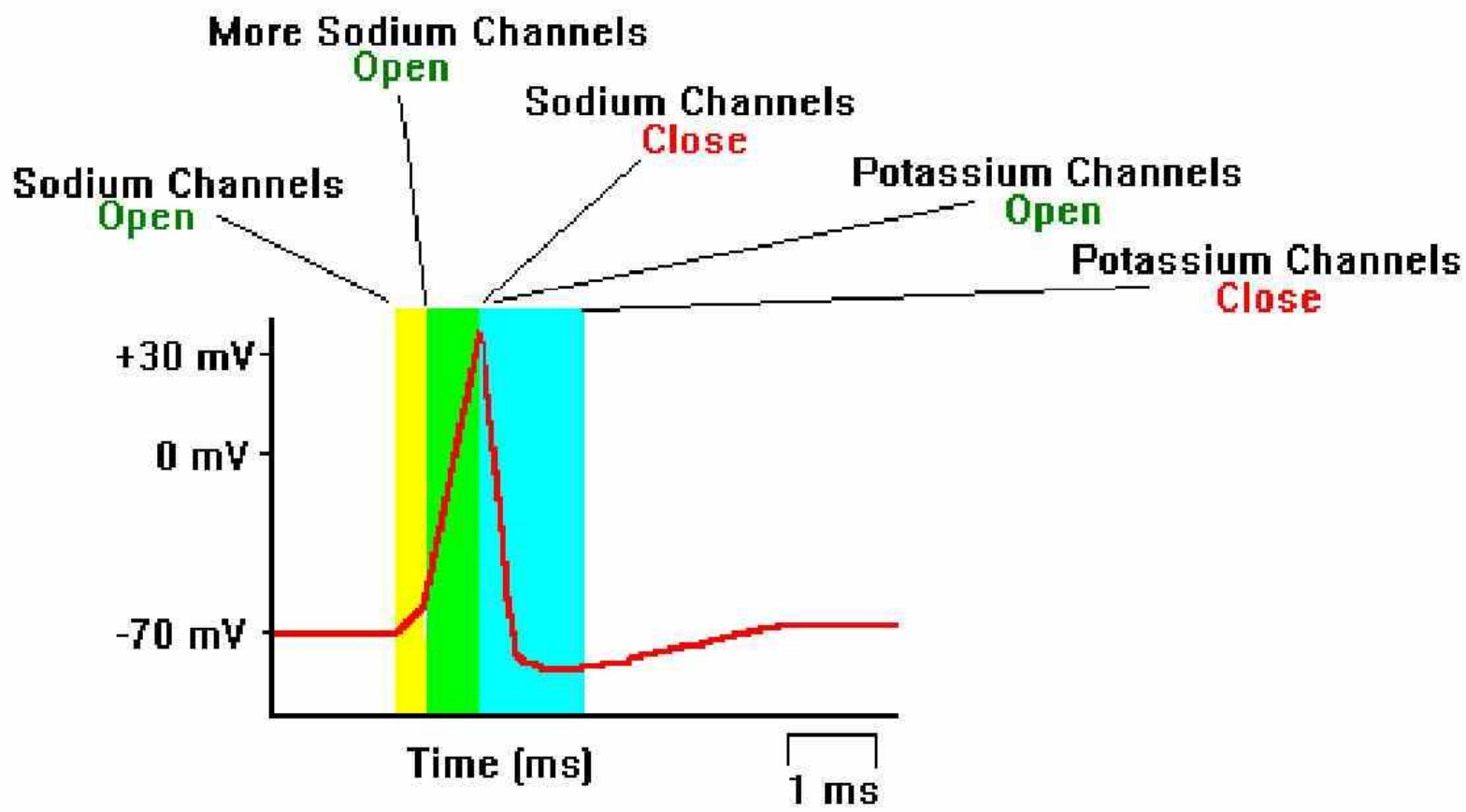


- Neurotransmissores putativos: serotonina, endorfina, dopamina, etc. Ao todo, são mais de 100 compostos orgânicos.
- O mal de Parkinson, por exemplo, é atribuído a uma deficiência de dopamina.



Binding of Neurotransmitter to ion channel

- A ativação de um neurônio é também denominada de *spiking*, *firing*, ou *disparo de um potencial de ação* (*triggering of an action potential*).

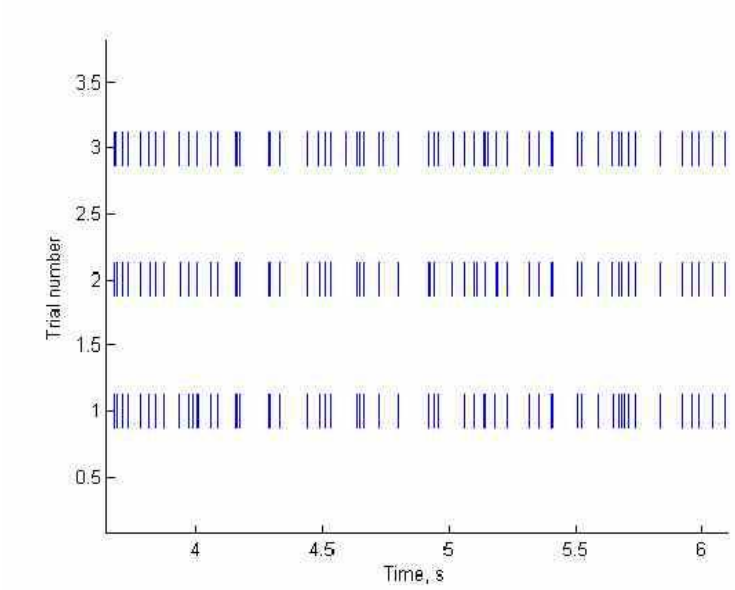


- Passos envolvidos no estabelecimento e extinção do potencial de ação:

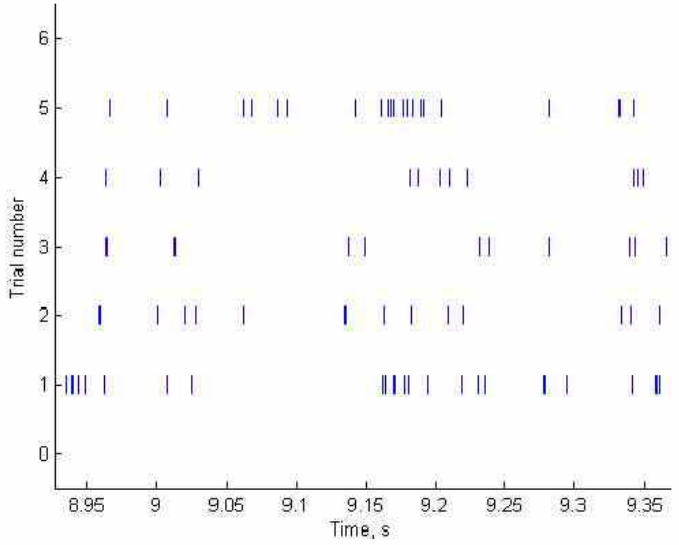
    1. Em uma célula em repouso, a parte externa da membrana é mais positiva que a parte interna, havendo mais íons de potássio dentro da célula e mais íons de sódio fora da célula.
    2. Pela ação dos neurotransmissores na sinapse, íons de sódio se movem para dentro da célula, causando uma diferença de potencial denominada potencial de ação. Com esta entrada de íons de sódio, o interior da célula passa a ser mais positivo que o exterior.
    3. Em seguida, íons de potássio fluem para fora da célula, restaurando a condição de interior mais negativo que exterior.
    4. Com as bombas de sódio-potássio, é restaurada finalmente a condição de maior concentração de íons de potássio dentro da célula e maior concentração de íons de sódio fora da célula.

- Segue-se um período refratário, durante o qual a membrana não pode ser estimulada, evitando assim a retropropagação do estímulo.

- Bombas de sódio e potássio: os íons de sódio que haviam entrado no neurônio durante a despolarização, são rebombeados para fora do neurônio mediante o funcionamento das bombas de sódio e potássio, que exigem gasto de energia.
- Para cada molécula de ATP empregada no bombeamento, 3 íons de sódio são bombeados para fora e dois íons de potássio são bombeados para dentro da célula. Esta etapa ocorre após a faixa azul da figura anterior.

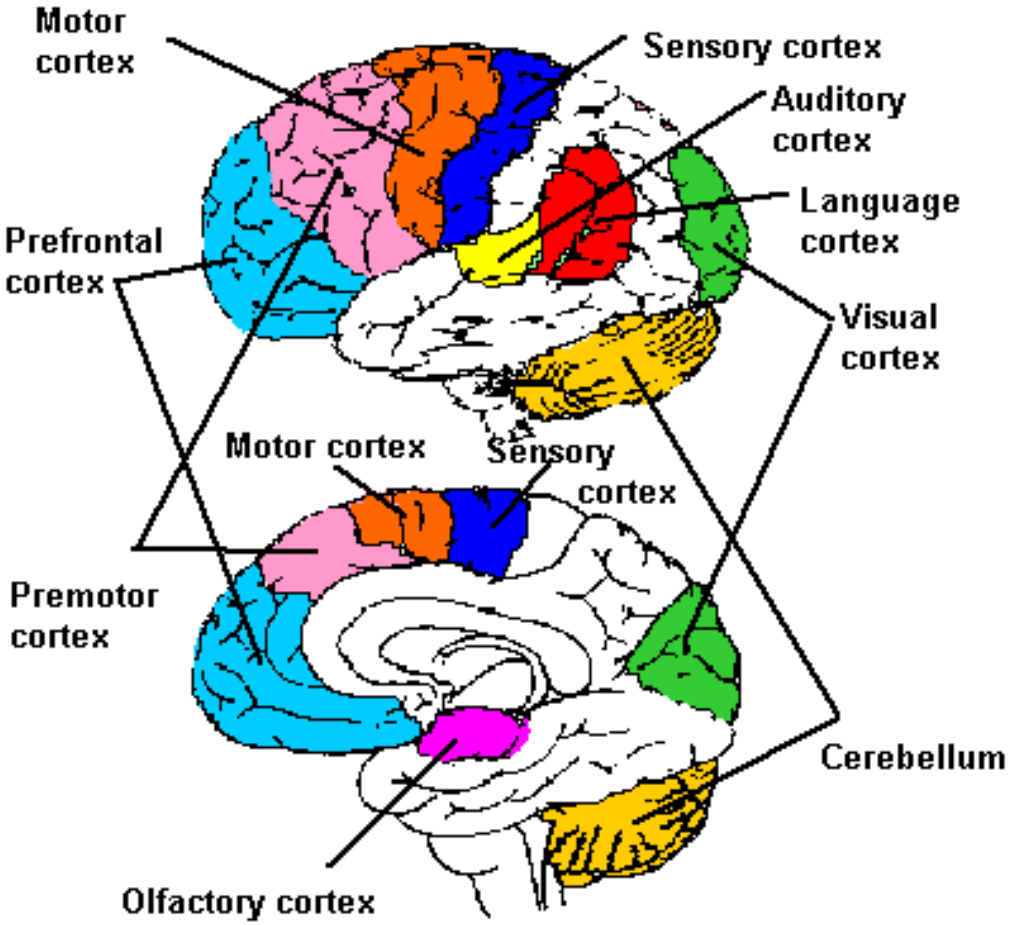


Neurônio periférico



Neurônio do córtex

- O córtex corresponde à superfície externa do cérebro: uma estrutura predominantemente bidimensional com vários dobramentos, fissuras e elevações.
- Diferentes partes do córtex possuem diferentes funções (ver figura abaixo).



6. Neurônio artificial

- Modelo matemático: simplificação da realidade com o propósito de representar aspectos relevantes de um sistema em estudo, sendo que detalhes de menor significância são descartados para viabilizar a modelagem.

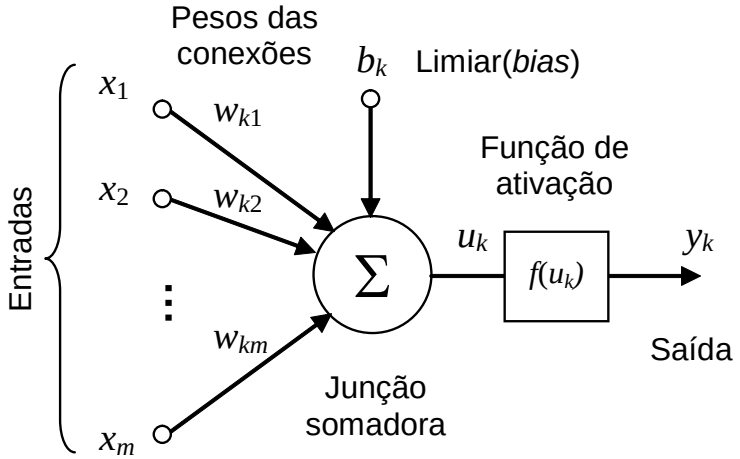


Figura 1 – Modelo matemático de um neurônio artificial.

- A saída do neurônio *k* pode ser descrita por:

$$y_k = f(u_k) = f\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right)$$

- É possível simplificar a notação acima de forma a incluir o *bias* simplesmente definindo um sinal de entrada de valor $x_0 = 1$ com peso associado $w_{k0} = b_k$:

$$y_k = f(u_k) = f\left(\sum_{j=0}^{m} w_{kj}x_j\right) = f(\mathbf{w}^T\mathbf{x}).$$

7. Exemplos mais usuais de funções de ativação

$$y = f(u_k) = \frac{e^{pu_k}}{e^{pu_k+1}} = \frac{1}{1+e^{-pu_k}} \qquad \frac{dy}{du_k} = py(1-y) > 0$$

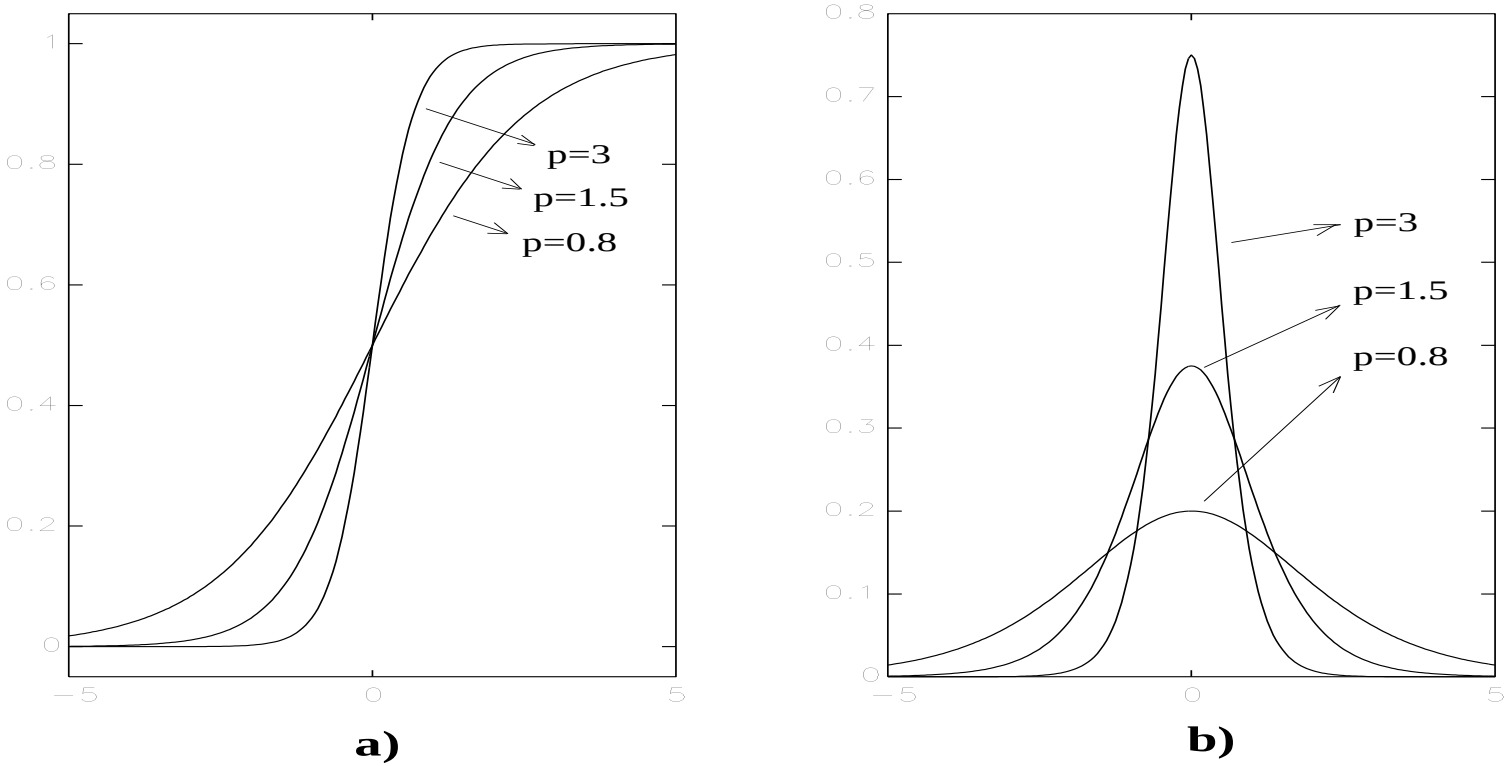


Figura 2 – Função logística (a) e sua derivada em relação à entrada interna (b).

$$y = f(u_k) = \tanh(pu_k) = \frac{e^{pu_k} - e^{-pu_k}}{e^{pu_k} + e^{-pu_k}} \qquad \frac{dy}{du_k} = p\left(1 - y^2\right) > 0$$

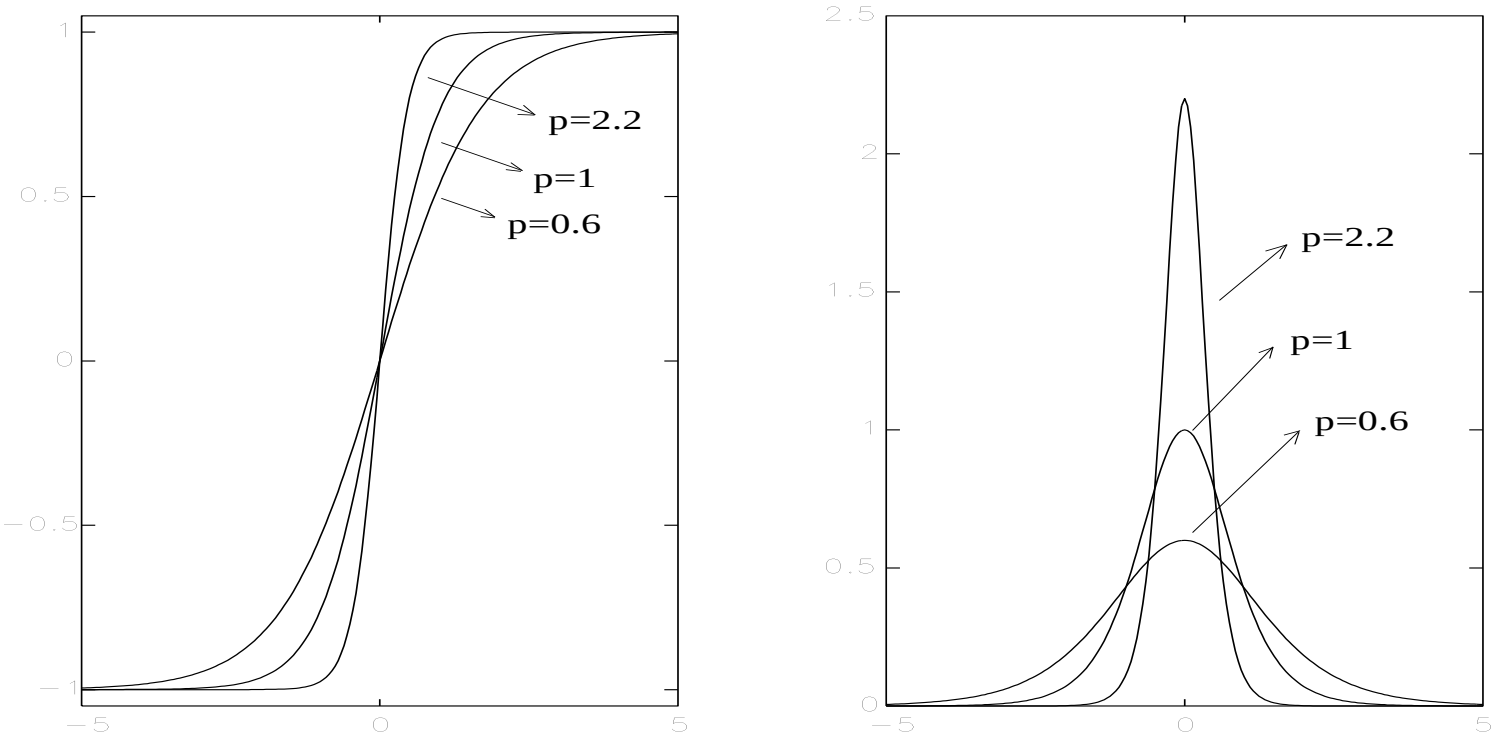
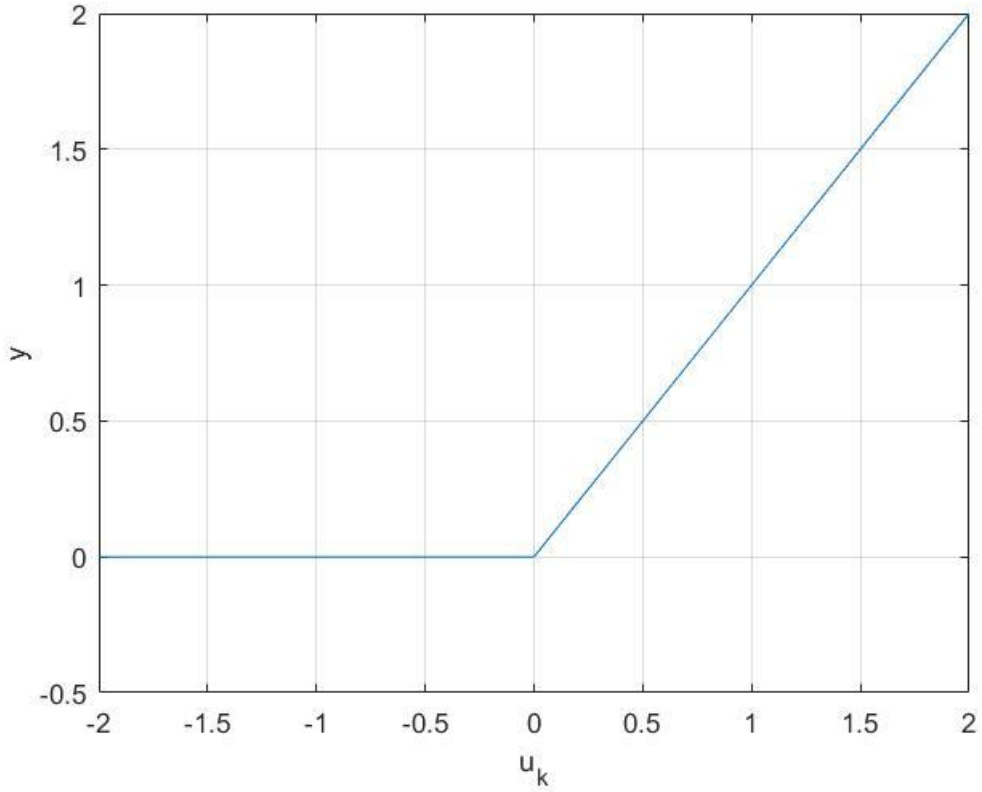


Figura 3 – Função tangente hiperbólica (a) e sua derivada em relação à entrada interna (b).

- Esta será a função de ativação a ser adotada nas implementações práticas do curso, com parâmetro $p = 1$ fixo.

- Outra função de ativação tem sido amplamente empregada no âmbito das redes neurais com aprendizado profundo (*deep learning*), a qual dá origem ao modelo de neurônio conhecido como ReLU (*rectified linear unit*):

$$y = f(u_k) = \max(0, u_k).$$



8. Produto interno e projeção

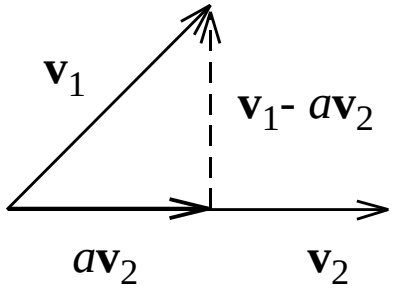


Figura 4 – Projeção realizada pelo produto interno no $\Re^2$.

- Sejam $\mathbf{v}_1, \mathbf{v}_2 \in \Re^2$ elementos não-nulos. Considere um escalar a tal que $a\mathbf{v}_2$ corresponda à projeção de $\mathbf{v}_1$ na direção de $\mathbf{v}_2$. Então, pode-se afirmar que

$$a\mathbf{v}_2 \perp \mathbf{v}_1 - a\mathbf{v}_2 ,$$

conduzindo a

$$\left\langle a\mathbf{v}_2, \mathbf{v}_1 - a\mathbf{v}_2 \right\rangle = 0 .$$

- Logo,

$$a\langle \mathbf{v}_2, \mathbf{v}_1 \rangle - a^2 \langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 0,$$

permitindo obter *a* na forma

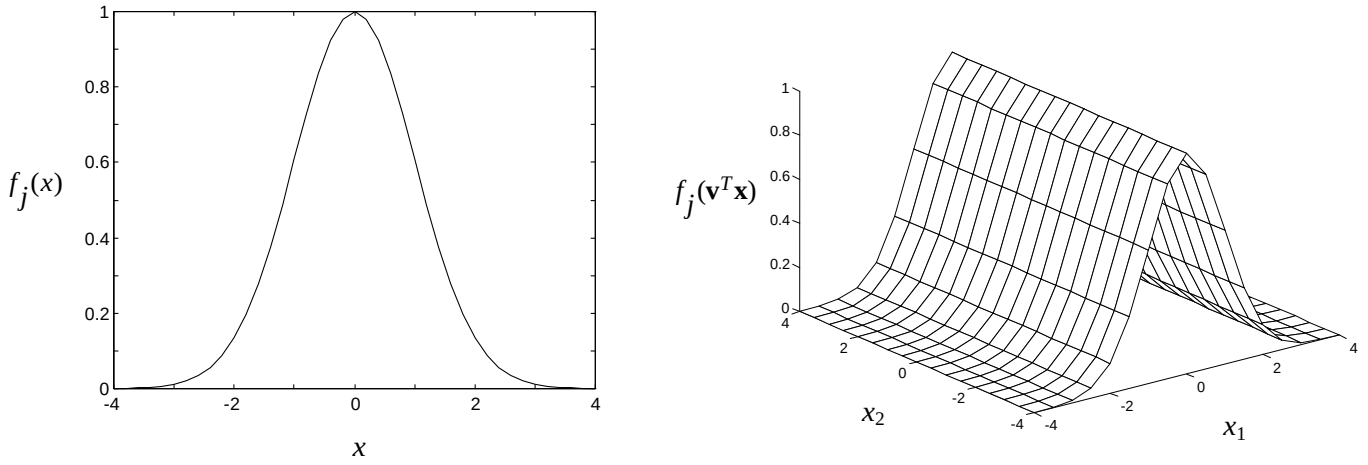
$$a = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle}.$$

- Isto significa que a projeção de $\mathbf{v}_1$ na direção de $\mathbf{v}_2$ ($\mathbf{v}_2 \neq \mathbf{0}$) assume a forma:

$$\text{proj}_{\mathbf{v}_2}(\mathbf{v}_1) = \frac{\langle \mathbf{v}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle} \mathbf{v}_2$$

- Mantendo constante o módulo de $\mathbf{v}_1$, a sua projeção na direção de $\mathbf{v}_2$ aumenta quanto mais colineares forem esses dois vetores.
- O produto interno, portanto, permite o estabelecimento de uma associação bem definida entre o vetor de estímulos de entrada de um neurônio e o seu vetor de pesos sinápticos, de modo que o neurônio é mais fortemente ativado quanto mais colineares forem esses dois vetores, supondo norma constante para esses vetores.

9. Função de expansão ortogonal



(a) $f_j(x) = e^{-0,5x^2}$ (b) $f_j(\mathbf{v}^T\mathbf{x}) = e^{-0,5\left([1\ 0]\begin{bmatrix}x_1\\x_2\end{bmatrix}\right)^2}$

Figura 5 – Função de expansão ortogonal em que $\mathbf{v} = [1\ 0]^T$ e $\mathbf{x} = [x_1\ \ x_2]^T$.

- A função de expansão ortogonal é conhecida na literatura em língua inglesa como *ridge function.*

10. Redes neurais e perceptron com uma camada intermediária

- O processo de conexão entre neurônios artificiais leva à geração de sinapses e à construção de redes neurais artificiais.

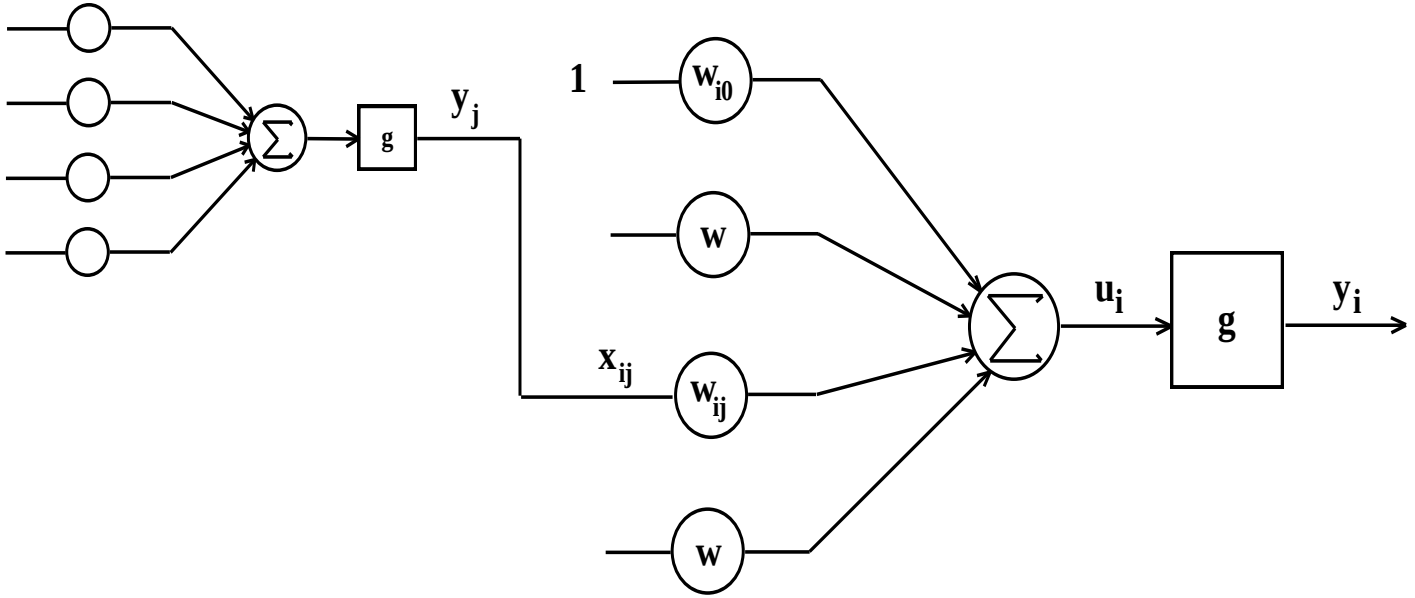


Figura 6 – Estabelecimento de conexão entre dois neurônios artificiais.

- As estruturas mais conhecidas são organizadas em camadas, nas quais a saída de cada neurônio de uma camada precedente serve de entrada para todos os neurônios da camada seguinte.

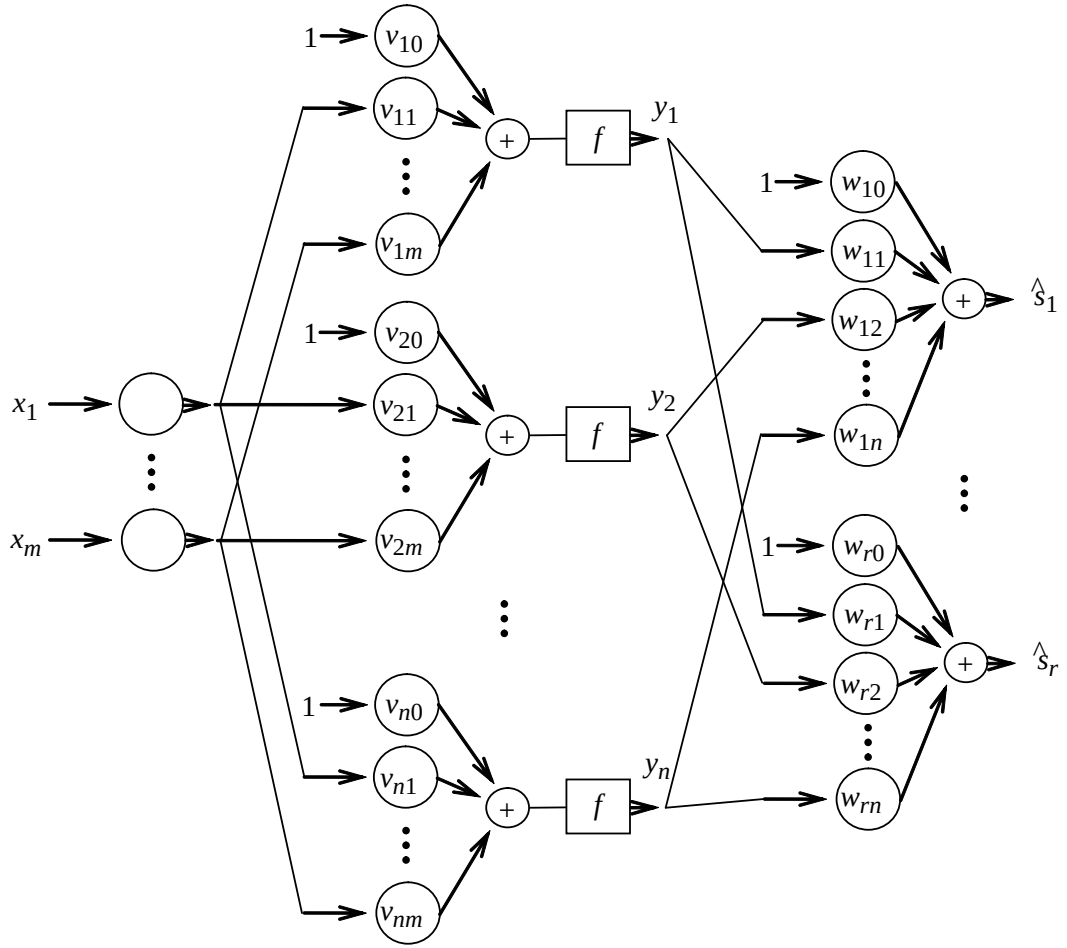


Figura 7 – Rede neural perceptron com uma camada intermediária (*Multilayer Perceptron*, MLP).

$$\hat{s}_k = \sum_{j=0}^{n} w_{kj} f\left(\sum_{i=0}^{m} v_{ji} x_i\right) = \sum_{j=0}^{n} w_{kj} f\left(\mathbf{v}_j^T \mathbf{x}\right) = \hat{g}_k(\mathbf{x}, \theta),\ k = 1,\ldots,r$$

11. Contribuição de cada neurônio em uma rede MLP

- Classicamente, o mapeamento não-linear realizado por uma rede neural do tipo perceptron de uma camada intermediária (MLP) é uma **combinação linear de funções de expansão ortogonal**, ou seja, funções que têm a forma de tangente hiperbólica em uma direção e são constantes nas demais direções ortogonais a esta única direção em que a forma da função se manifesta.
- Como um exemplo, vamos tomar amostras de um mapeamento do $\Re^2$ para o $\Re^1$, e utilizar uma rede neural com cinco neurônios na camada intermediária para buscar aproximar este mapeamento, o qual pode ser visualizado no $\Re^3$.
- Os pesos sinápticos resultantes do processo de treinamento estão apresentados na sequência, sendo que a rede neural tem ao todo $3 \times 5 + 6 \times 1 = 21$ pesos ajustáveis. São $m = 2$ entradas, $n = 5$ neurônios na camada intermediária e $r = 1$ saída, mais as entradas constantes (entradas de polarização) de todos os $(n+r) = 6$ neurônios da rede neural.

- Pesos sinápticos da camada intermediária (cada coluna representa os pesos de um neurônio):

-0.20008939714462 -0.70051908010040 0.39699221844113 -0.10003863267278 0.69606262467282

0.70018168528932 0.10015860417667 0.19860028823484 -0.29996195303800 0.29869112235480

-0.30006398146599 0.80022209855791 0.49372400421686 0.50005427222963 0.89515012131364

- Pesos sinápticos da camada de saída:

0.99989340388393

0.79971888341317

0.90007841696146

0.38564988369799

0.79996881679466

0.71442550587375

- OBS: A polarização está associada ao primeiro elemento do vetor de pesos de cada neurônio.

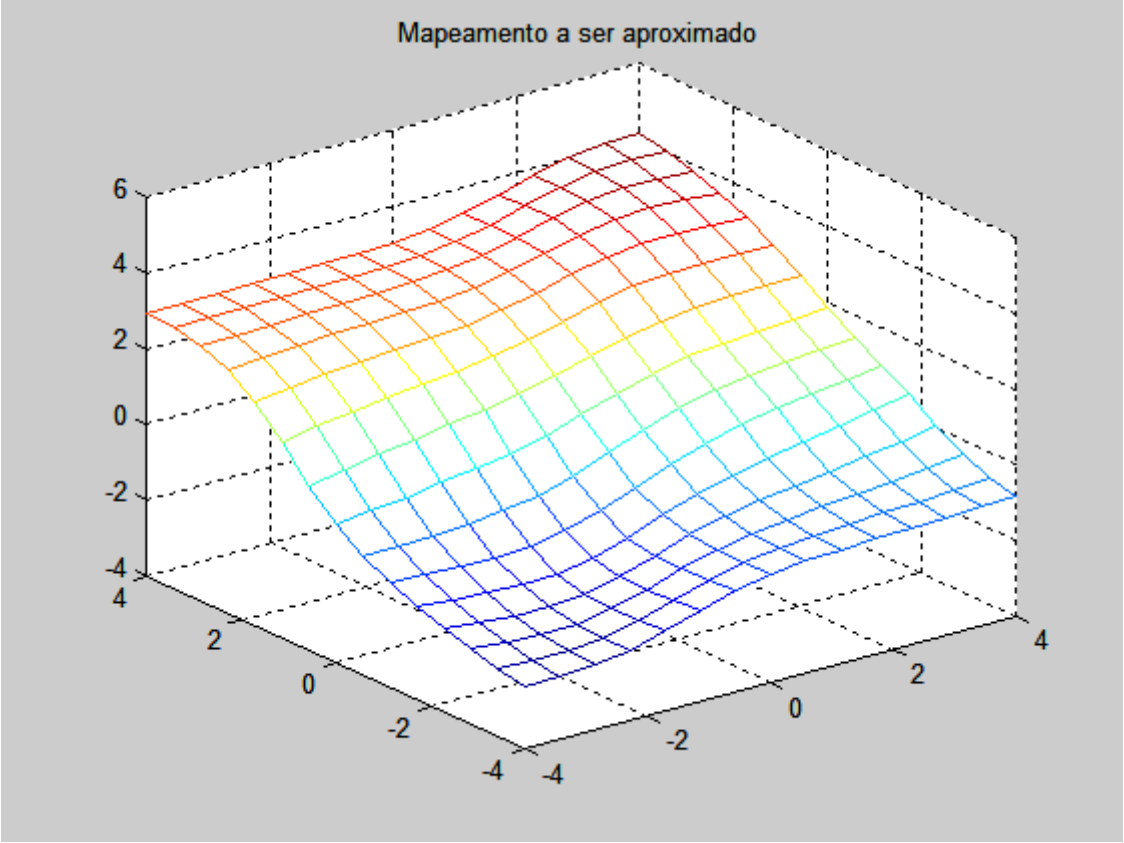


Figura 8 – Mapeamento a ser aproximado.

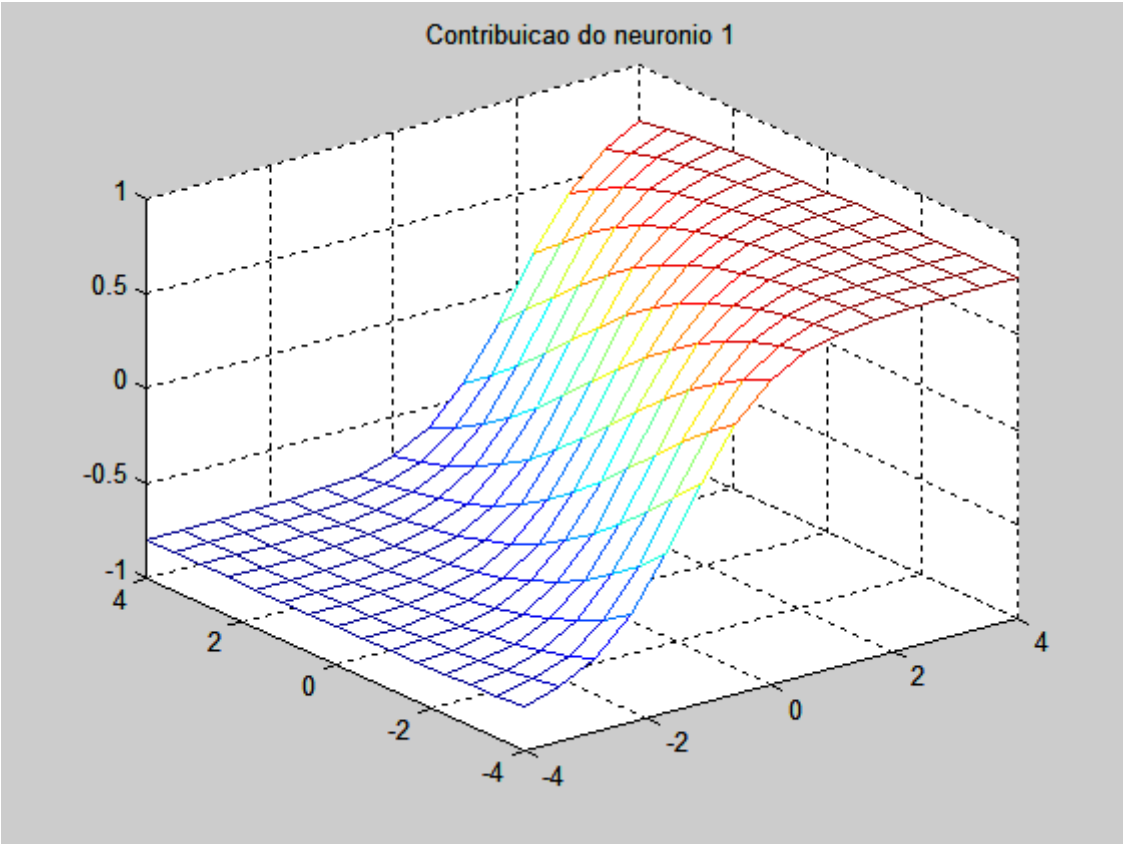


Figura 9 – Contribuição do neurônio 1.

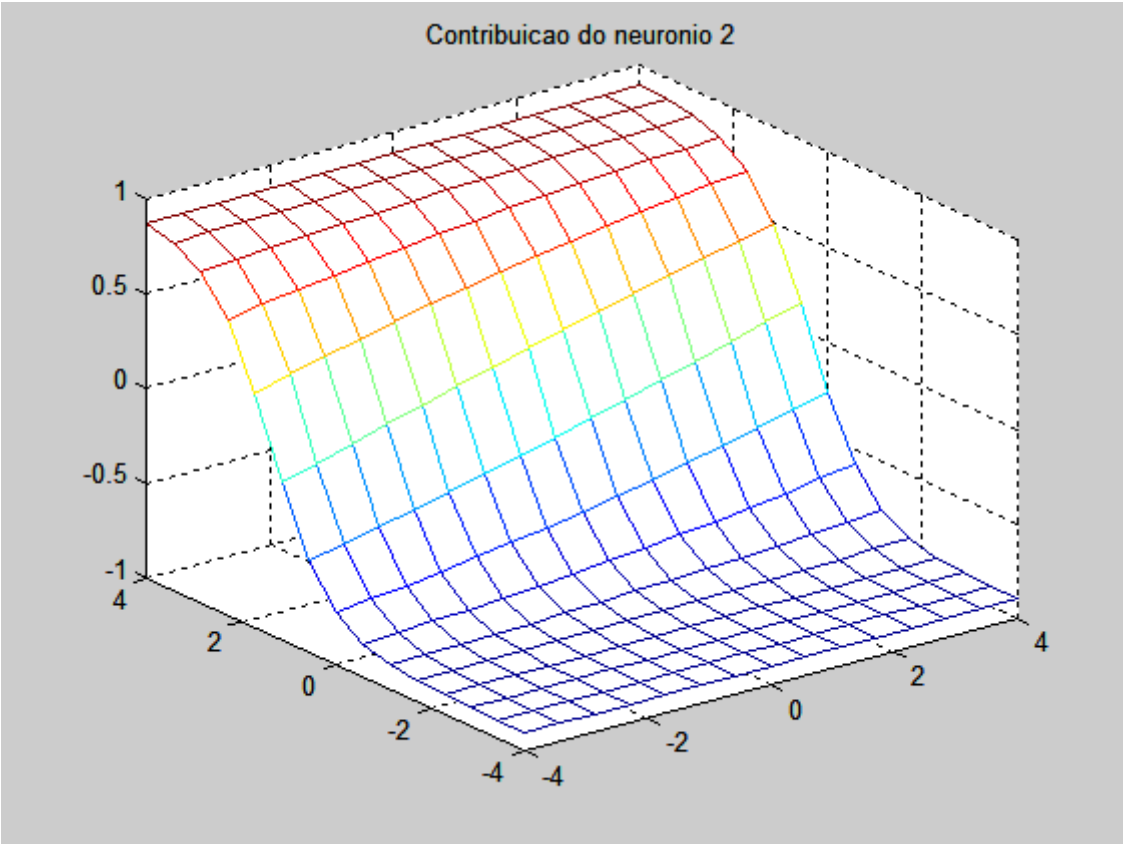


Figura 10 – Contribuição do neurônio 2.

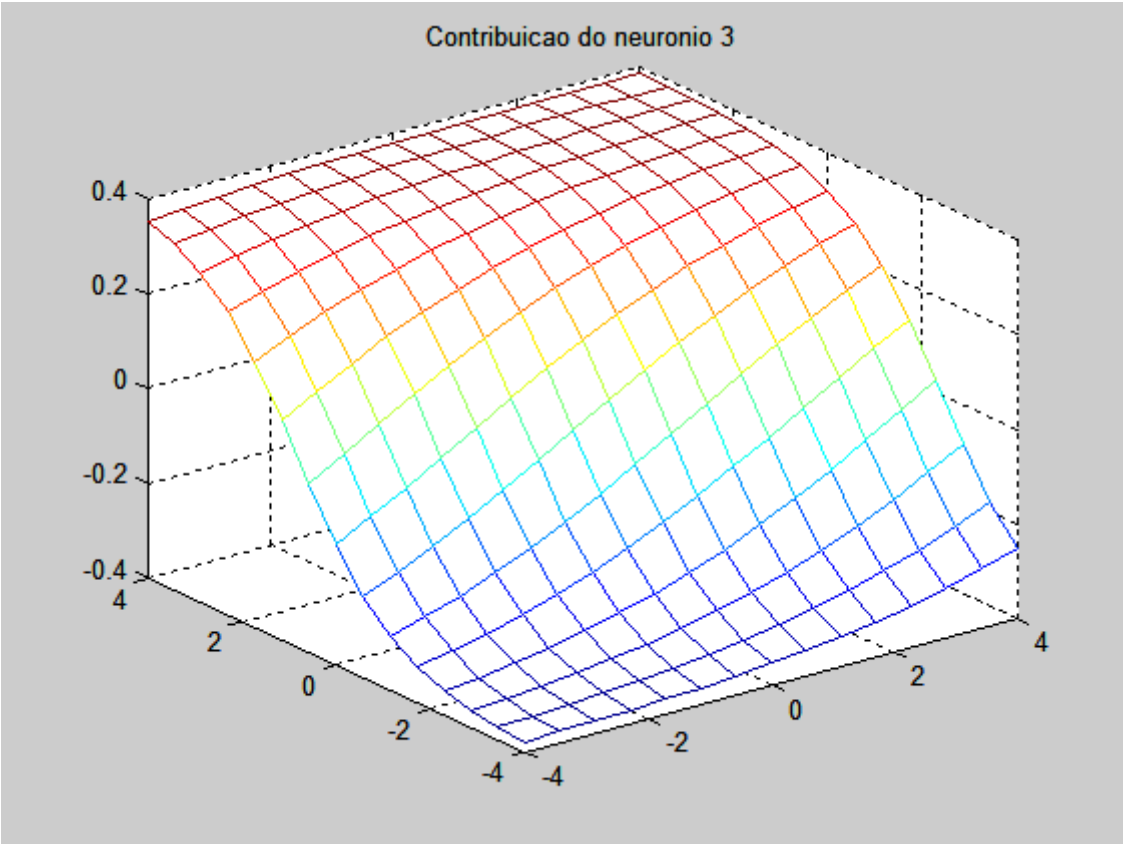


Figura 11 – Contribuição do neurônio 3.

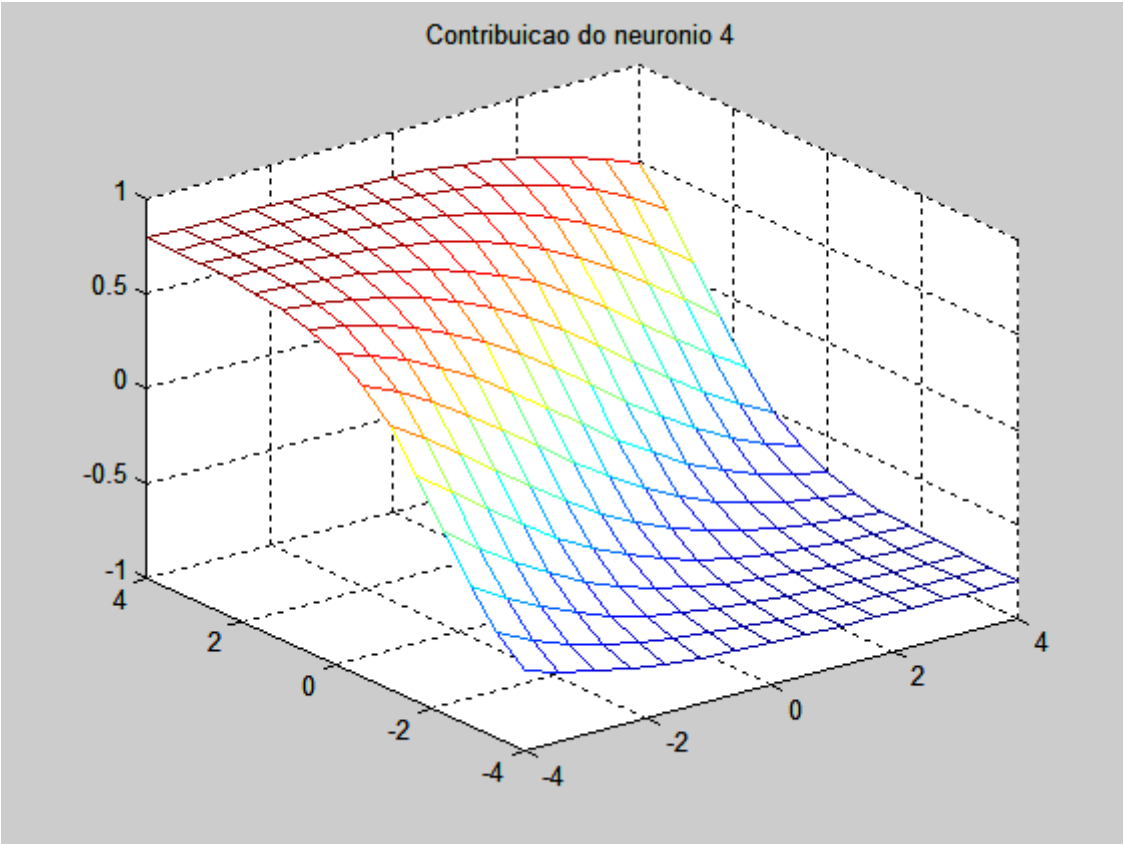


Figura 12 – Contribuição do neurônio 4.

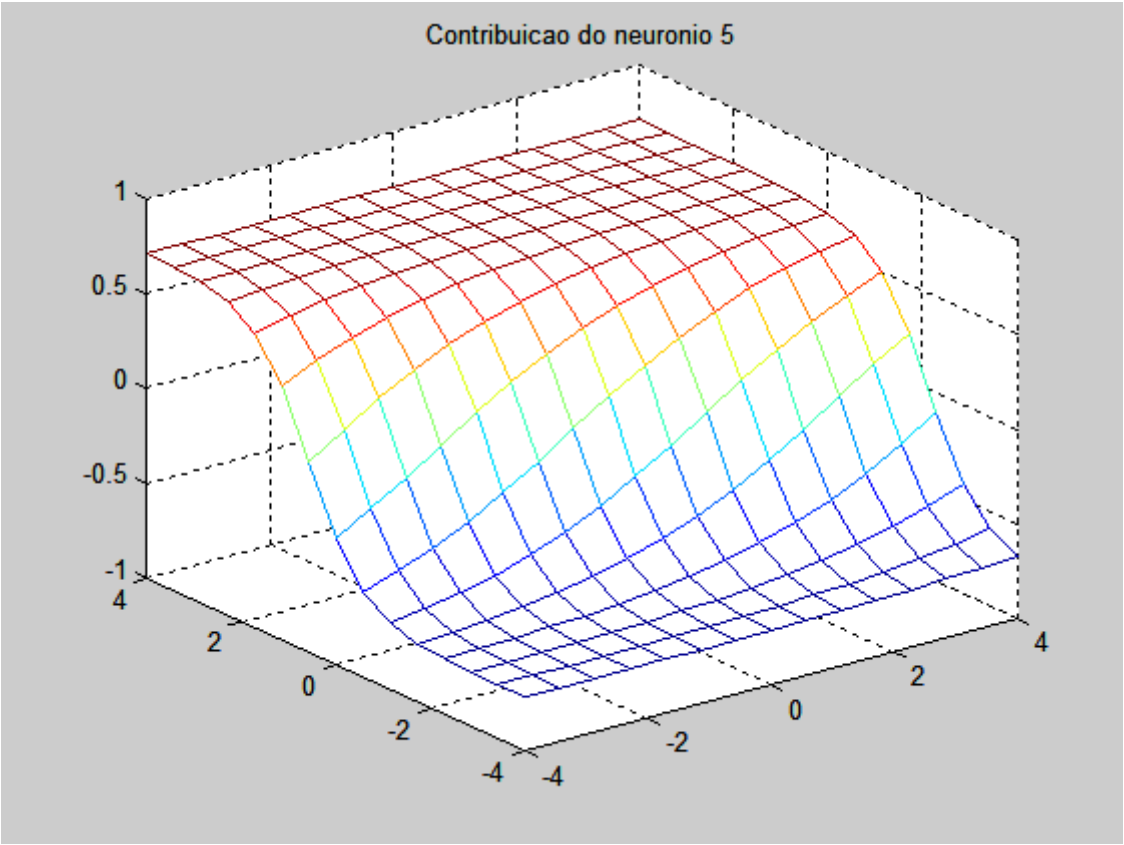
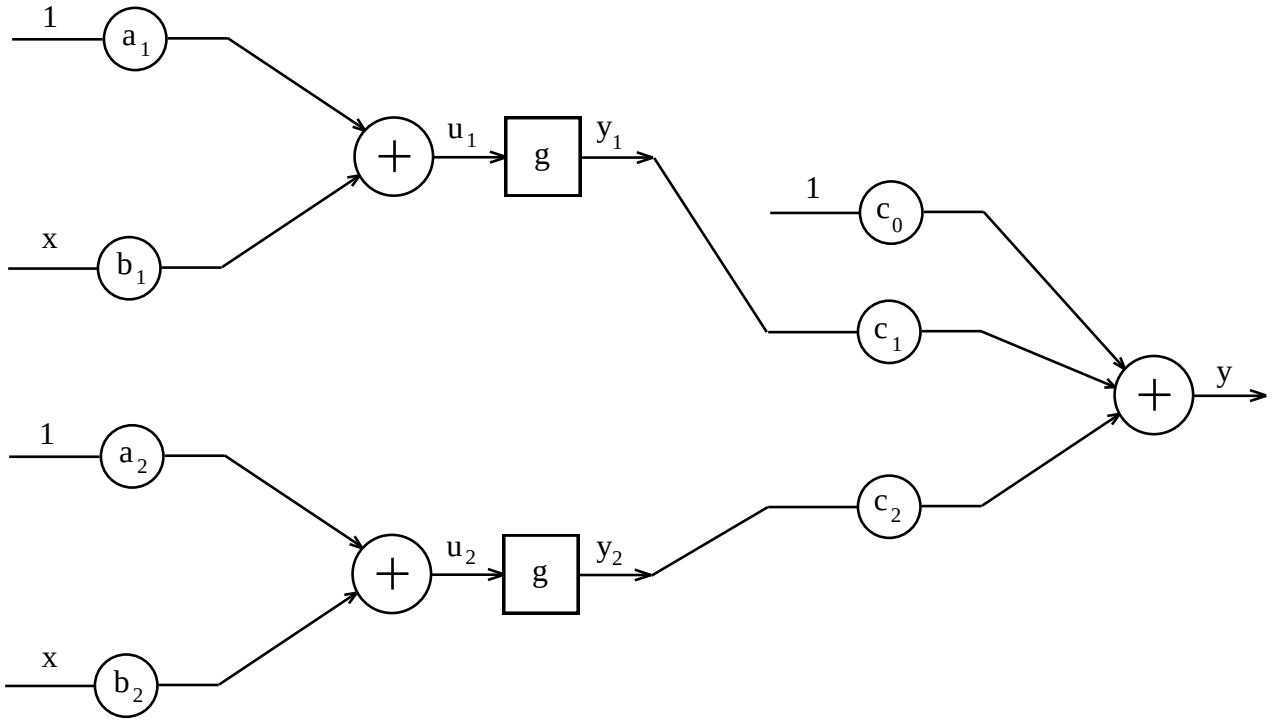


Figura 13 – Contribuição do neurônio 5.

12. O papel dos pesos sinápticos



$$y = c_0 + c_1 g(b_1 x + a_1) + c_2 g(b_2 x + a_2) \Rightarrow \begin{cases} \text{a : deslocamento no eixo x} \\ \text{b : inclinação da sigmóide} \\ \text{c : amplitude da sigmóide} \end{cases}$$

13. Superfície de erro

- Seja X uma região compacta do $\Re^m$ e seja g: $X \subset \Re^m \rightarrow \Re$ a função a ser aproximada (formulação para uma única saída, $r = 1$).
- O conjunto de dados de aproximação $\{(\mathbf{x}_l, s_l) \in \Re^m \times \Re\}_{l=1}^N$ é gerado considerando-se que os vetores de entrada $\mathbf{x}_l$ estão distribuídos na região compacta $X \subset \Re^m$ de acordo com uma função densidade de probabilidade fixa d_P: $X \subset \Re^m \rightarrow [0,1]$ e que os vetores de saída s_l são produzidos pelo mapeamento definido pela função g na forma:

$$s_l = g(\mathbf{x}_l) + \varepsilon_l, \quad l = 1,\ldots,N,$$

  onde $\varepsilon_l \in \Re$ é uma variável aleatória de média zero e variância fixa.
- A função g que associa a cada vetor de entrada $\mathbf{x} \in X$ uma saída escalar $s \in \Re$ pode ser aproximada com base no conjunto de dados de aproximação $\{(\mathbf{x}_l, s_l) \in \Re^m \times \Re\}_{l=1}^N$ por uma composição aditiva de funções de expansão ortogonal na forma:

$$\hat{s}_l = \hat{g}(\mathbf{x}_l, \theta) = \sum_{j=0}^{n} w_j \, f\left(\sum_{i=0}^{m} v_{ji} x_{li} \right) = \sum_{j=0}^{n} w_j \, f\left(\mathbf{v}_j^T \mathbf{x}_l \right)$$

onde θ é o vetor contendo todos os pesos sinápticos da rede neural.

- Logo, o erro quadrático médio produzido na saída da rede neural, considerando as N amostras, assume a forma:

$$J(\theta) = \frac{1}{N} \sum_{l=1}^{N} (\hat{s}_l - s_l)^2 = \frac{1}{N} \sum_{l=1}^{N} (\hat{g}(\mathbf{x}_l, \theta) - s_l)^2 =$$

$$= \frac{1}{N} \sum_{l=1}^{N} \left(\sum_{j=0}^{n} w_j \, f\left(\sum_{i=0}^{m} v_{ji} x_{li} \right) - s_l \right)^2 = \frac{1}{N} \sum_{l=1}^{N} \left(\sum_{j=0}^{n} w_j \, f\left(\mathbf{v}_j^T \mathbf{x}_l \right) - s_l \right)^2$$

- Sendo P a dimensão do vetor θ, então tem-se que: $J : \Re^P \rightarrow \Re^1$.
- A superfície de erro definida por $J(\theta)$ reside no espaço $\Re^{P+1}$, sendo que deve-se buscar em $\Re^P$ um ponto que minimiza $J(\theta)$, supondo que se queira minimizar o erro entre a saída produzida pelo rede neural e a saída desejada.

14. Aprendizado a partir de dados amostrados

- O processo de aprendizado supervisionado de uma rede neural pode ser visto como um problema de otimização não-linear sem restrições sobre os valores dos pesos sinápticos.
- A função objetivo (i.e., o critério de desempenho a ser otimizado, no caso, erro quadrático médio a ser minimizado) em função dos parâmetros ajustáveis:

$$\min_{\theta \in \Re^P} J(\theta)$$

- Formalização matemática do que se quer otimizar + método de solução.
- Solução na forma fechada × Busca iterativa.
- Os dados de entrada/saída e a questão dos 3 mapeamentos envolvidos no processo:
    1. O mapeamento a ser aproximado (do qual se conhece apenas um conjunto finito de dados amostrados);

2. O mapeamento resultante do processo de aproximação, associado a um único vetor de pesos sinápticos – é o mapeamento que a rede efetivamente produz;
3. O mapeamento entre cada vetor de pesos e o erro: superfície de erro.

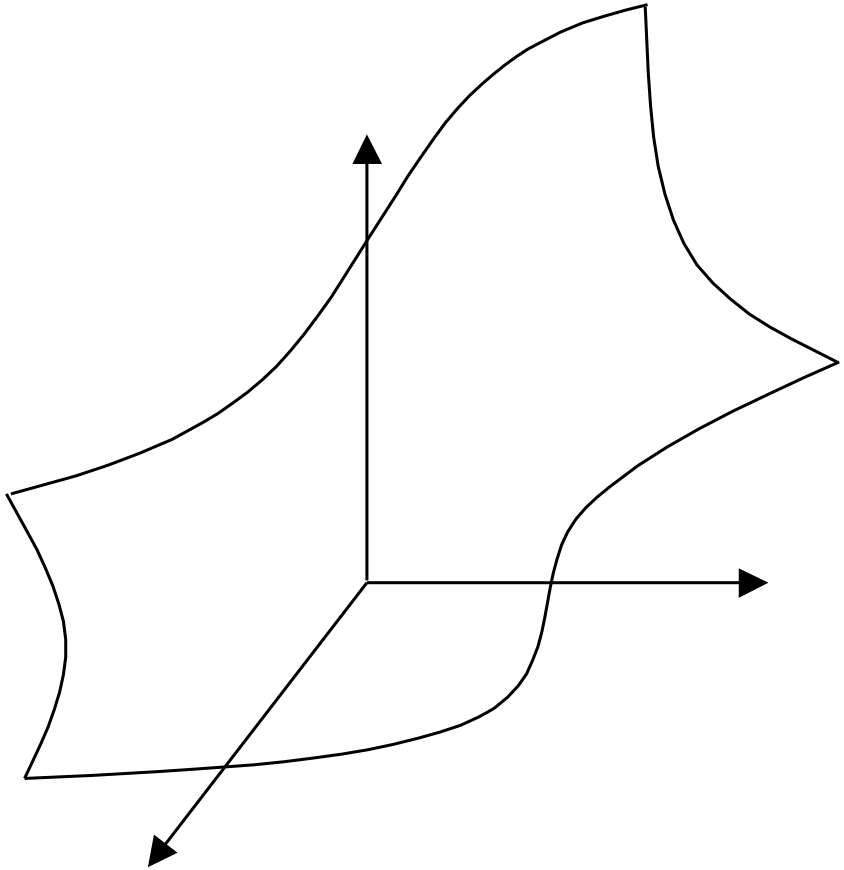


Figura 14– Mapeamento desconhecido a ser aproximado.

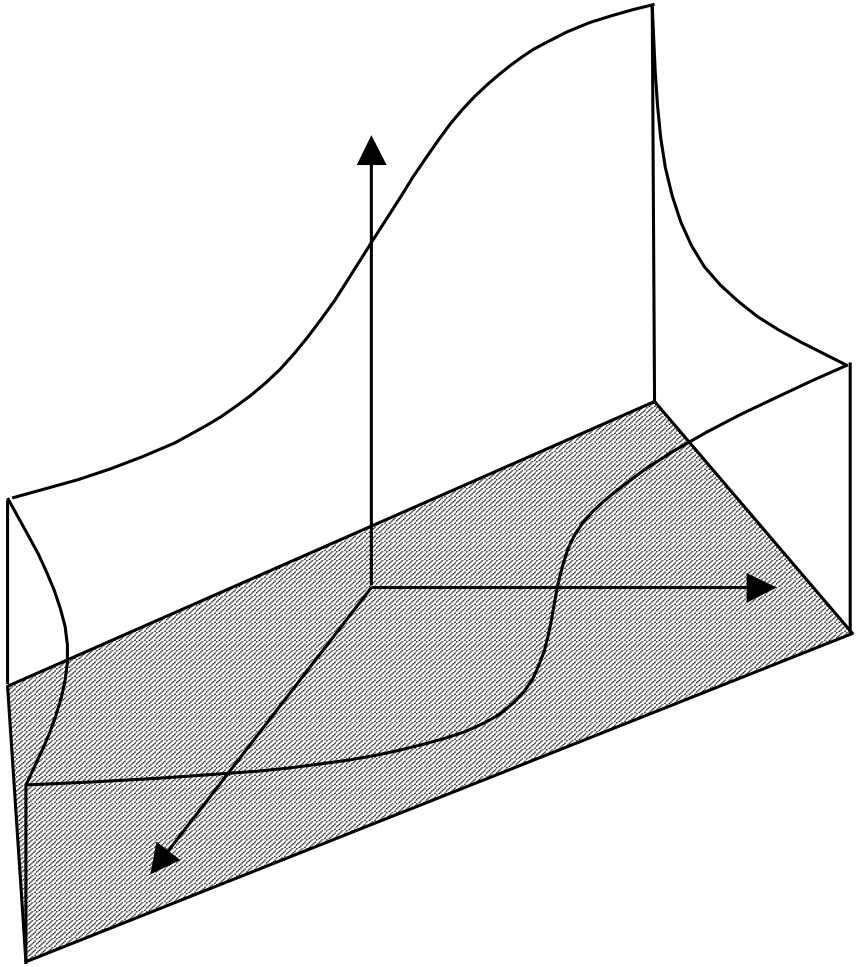


Figura 15 – Exemplo de região de operação. É uma região compacta (fechada e limitada).

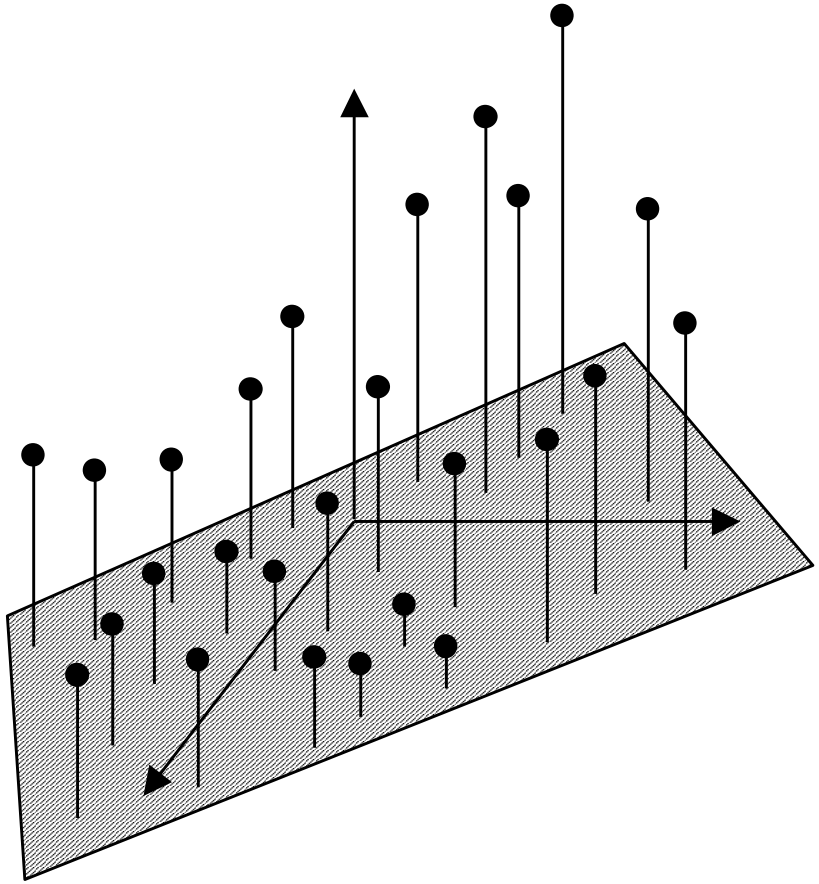


Figura 16 – Amostras expressando o comportamento da função para pontos específicos da região de operação. Essas amostras farão parte dos conjuntos de treinamento e validação (sendo que os dois conjuntos são independentes entre si).

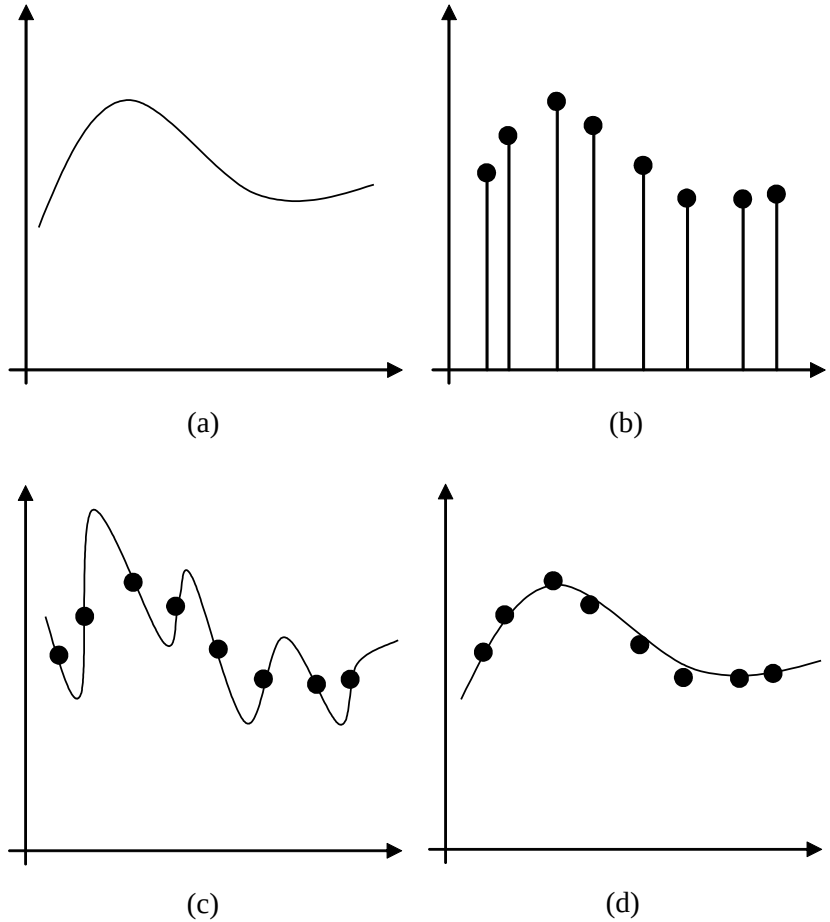


Figura 17 – (a) Função a ser aproximada (agora considerando apenas uma entrada); (b) Amostras disponíveis; (c) Resultado de um processo de aproximação com sobretreinamento (*overfitting*); (d) Resultado de um processo de aproximação sem sobretreinamento.

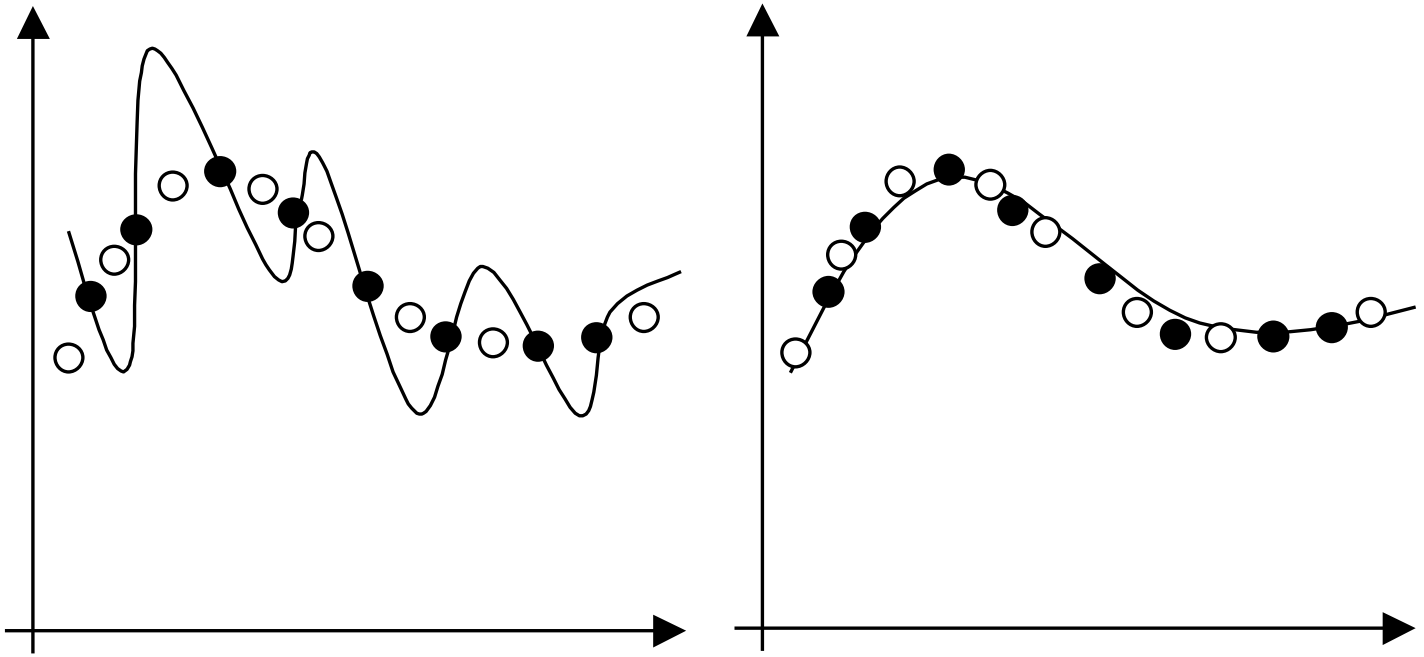


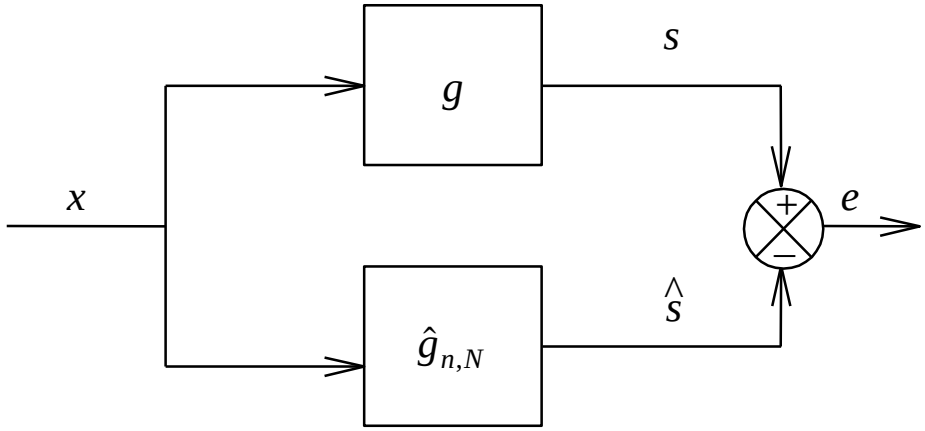
Figura 18 – Comparação de desempenho para dados de treinamento e validação, de modo a medir a capacidade de generalização dos mapeamentos produzidos.

- O mapeamento da esquerda apresenta um erro de treinamento muito baixo, mas um erro de validação bastante elevado, quando comparado ao mapeamento da direita.

15. Os Três Erros do Processo de Aproximação

- Durante o processo de aproximação da função $g(.)$ pela função $\hat{g}(.,\theta)$ – produzida pela rede neural –, devem ser considerados três tipos de erros (VAN DER SMAGT, 1994): o erro de *representação*, o erro de *generalização* e o erro de *computação*.
- *Erro de Representação*: Primeiro consideremos o caso em que todo o conjunto amostral está disponível $\{(\mathbf{x}_l, \mathbf{s}_l)\}_{l=1}^{\infty}$. Considere também que, dado $\{(\mathbf{x}_l, \mathbf{s}_l)\}_{l=1}^{\infty}$, é possível encontrar um conjunto de parâmetros ótimo θ^*. Neste caso, o erro vai depender da adequação e do nível de flexibilidade do modelo de aproximação $\hat{g}(.,\theta)$. Este erro é também conhecido como erro de aproximação, ou *efeito bias*.
- *Erro de Generalização*: Em aplicações de mundo real, somente um número finito de amostras está disponível ou pode ser usado simultaneamente. Além disso, os dados podem conter ruído. Os valores de $g(.)$ para os quais nenhuma amostra está disponível devem ser interpolados. Devido a estes fatores, pode ocorrer um erro de generalização, também conhecido como erro de estimação, ou *variância*.

- *Erro de Computação*: Nem sempre é possível explorar devidamente o espaço de hipóteses. Três razões: mínimos locais, limitação dos recursos computacionais para a busca e uso de representação numérica de precisão finita, todas associadas ao problema de otimização vinculado. Também conhecido como erro de otimização.



- $T_N = \{x_l, s_l\}_{l=1}^{N}$: conjunto de dados amostrados, sendo N o número de amostras;
- $X \subset \Re^m$: espaço de entrada, $x \in X$;
- $S \subset \Re^r$: espaço de saída, $s \in S$;
- C: classe de funções sendo modelada, $g \in C$;
- H_n: conjunto de funções realizáveis por uma dada classe de modelos de tamanho n, $\hat{g}_{n,N} \in H_n$ (também denominado de espaço de hipóteses).

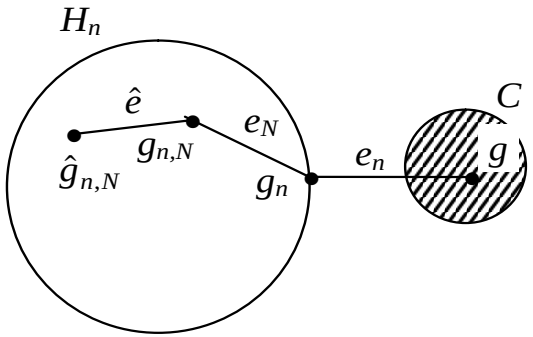
Composição dos três erros de aproximação

$$\hat{e}_{n,N} = g - \hat{g}_{n,N}$$

$$\hat{e}_{n,N} = \underbrace{g - g_n}_{e_n} + \underbrace{g_n - g_{n,N}}_{e_N} + \underbrace{g_{n,N} - \hat{g}_{n,N}}_{\hat{e}}$$

- e_n: Erro de representação
- e_N: Erro de generalização
- $\hat{e}$: Erro de computação

Usando a desigualdade triangular: $\left\|\hat{e}_{n,N}\right\| \leq \left\|e_n\right\| + \left\|e_N\right\| + \left\|\hat{e}\right\|$



Nota: As figuras desta seção 3.3 estão baseadas em Hush (1997).

16. Otimização não-linear irrestrita e capacidade de generalização

- Diversos tipos de parâmetros da rede neural poderiam ser submetidos a processos de ajuste durante o treinamento, como (*i*) pesos sinápticos; (*ii*) parâmetros da função de ativação de cada neurônio; (*iii*) número de neurônios na camada intermediária; (*iv*) número de camadas intermediárias.
- Iremos nos restringir aqui ao ajuste dos pesos sinápticos. Neste caso, o processo de treinamento supervisionado de redes neurais artificiais multicamadas é equivalente a um problema de otimização não-linear irrestrita, em que a superfície de erro reside em um espaço contínuo, que aponta o erro quadrático médio para cada vetor de pesos no $\Re^P$, e é minimizada a partir do ajuste dos pesos sinápticos.
- Iremos nos restringir também a redes MLP com uma única camada intermediária, visto que com apenas uma camada intermediária a rede neural já apresenta **capacidade de aproximação universal** (CYBENKO, 1989; HORNIK *et al.*, 1989; HORNIK *et al.*, 1990; HORNIK *et al.*, 1994).

- Um problema comum a todos os modelos de aproximação de funções que possuem capacidade de aproximação universal, não apenas redes neurais artificiais do tipo MLP, é a necessidade de controlar adequadamente o seu grau de flexibilidade.
- Como o conjunto de amostras disponíveis para treinamento supervisionado é finito, infinitos mapeamentos podem produzir o mesmo desempenho de aproximação, independente do critério de desempenho adotado. Esses mapeamentos alternativos vão diferir justamente nas regiões em que não há amostras disponíveis para diferenciá-los.
- Visando maximizar a **capacidade de generalização** do modelo de aproximação (no caso, uma rede neural MLP), ou seja, buscando encontrar o grau de flexibilidade adequado para o modelo de aproximação (dada a demanda da aplicação), um procedimento recomendado é dividir o conjunto de amostras disponível para treinamento em dois: um conjunto que será efetivamente empregado no ajuste dos pesos (conjunto de treinamento) e um conjunto que será empregado para definir o momento de interromper o treinamento (conjunto de validação).

- Deve-se assegurar que ambos os conjuntos sejam suficientemente representativos do mapeamento que se pretende aproximar. Assim, minimizar o erro junto ao conjunto de validação implica em maximizar a capacidade de generalização. Logo, espera-se que a rede neural que minimiza o erro junto ao conjunto de validação (não usado para o ajuste dos pesos) tenha o melhor desempenho possível junto a novas amostras.
- A figura alto/esquerda a seguir mostra um mapeamento unidimensional a ser aproximado (desconhecido pela rede neural) e amostras sujeitas a ruído de média zero (única informação disponível para o treinamento da rede neural). A figura alto/direita mostra o resultado da aproximação produzida por uma rede neural com poucos neurônios, a qual foi incapaz de realizar a aproximação (tem baixa flexibilidade).
- Já as figuras baixo/esquerda e baixo/direita mostram o resultado de uma mesma rede neural (com número suficiente de neurônios), mas à esquerda ocorreu sobreajuste, enquanto que à direita o treinamento foi interrompido quando o erro junto a dados de validação (não apresentados) foi minimizado.

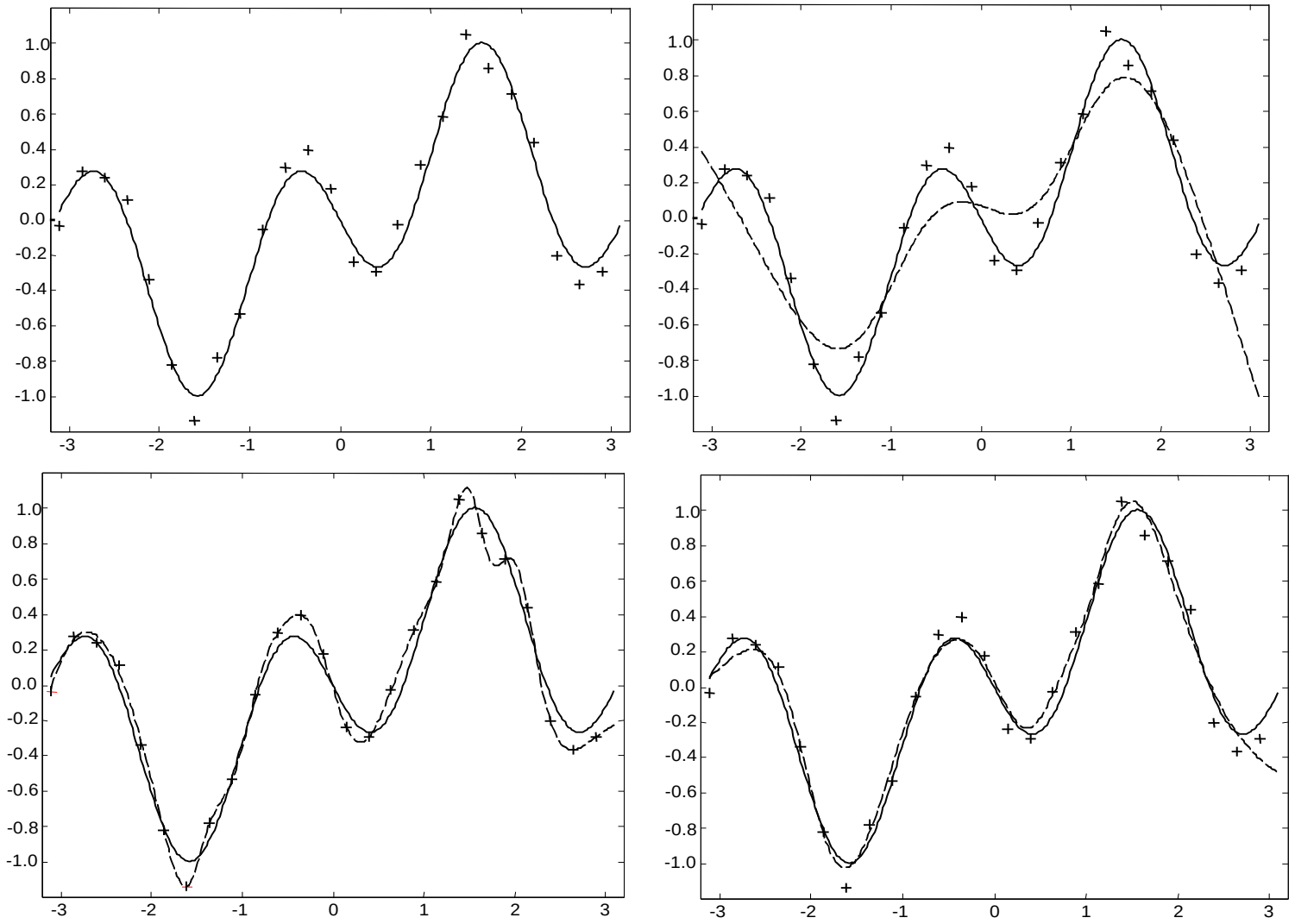


Figura 19 – Dados amostrados, função a ser aproximada e modelos de aproximação com diferentes capacidades de generalização

- Curvas típicas de erro de treinamento e validação são apresentadas a seguir.

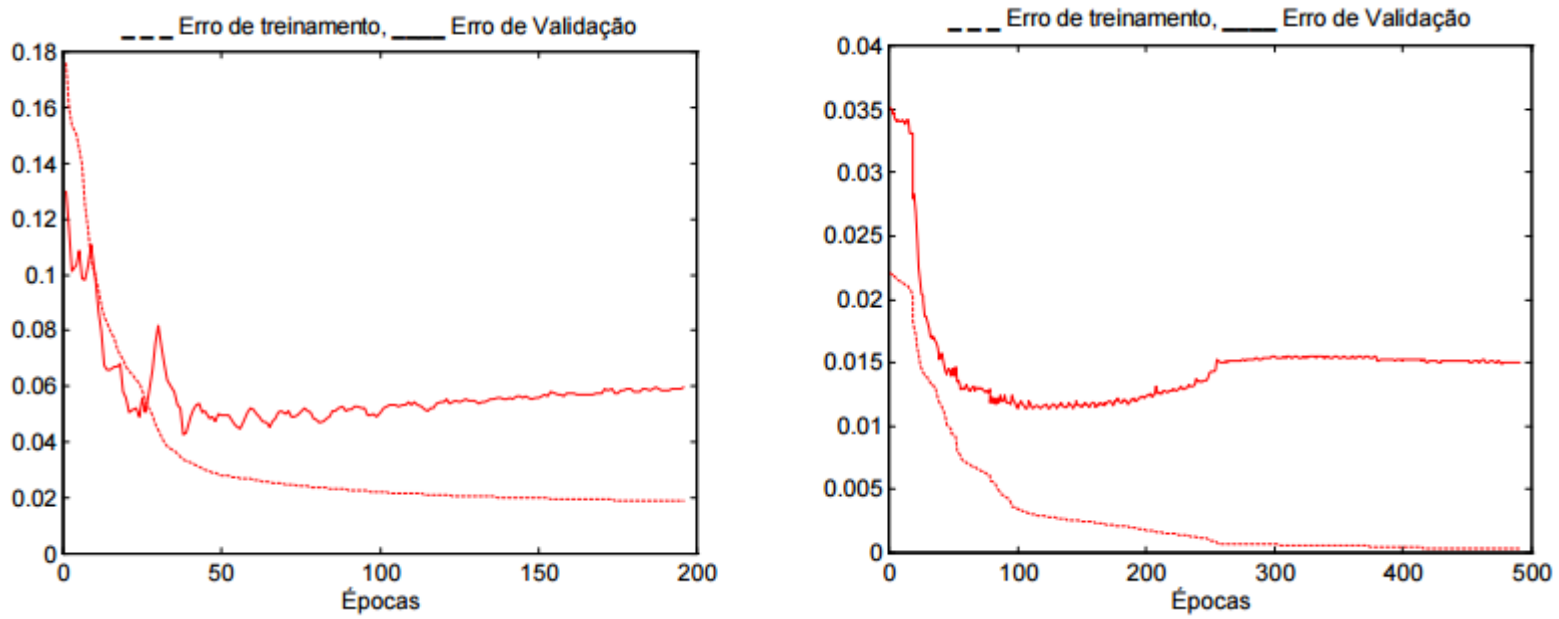


Figura 20 – Ilustrações típicas da evolução, ao longo das épocas de treinamento, dos erros de treinamento e de validação em treinamento supervisionado de redes MLP.

- No entanto, nem sempre o erro de validação apresenta este comportamento, e cada caso deve ser analisado isoladamente. Como a curva do erro de validação oscila bastante e esboça um comportamento pouco previsível, não é indicado desenvolver detectores automáticos de mínimos e encerrar o treinamento ali. O mais indicado é

permitir que o treinamento prossiga (i.e., sobretreinar a rede) e armazenar os pesos associados ao mínimo do erro de validação.

- Não existe um consenso sobre como separar as amostras do conjunto de dados, ou seja, sobre como dividi-lo de forma que possamos encontrar uma rede com a melhor capacidade de generalização em todos os casos. Uma sugestão de partida pode ser destinar 80% das amostras para treinamento e 20% para validação.
- Quando as amostras correspondem a dados rotulados em problemas de classificação de padrões, procure respeitar a distribuição junto a cada classe.
- Este mecanismo de regularização do treinamento de redes neurais é denominado *holdout*, por manter de fora do treinamento um subconjunto das amostras. Uma alternativa mais robusta é conhecida como *k-fold cross-validation*.

16.1 Gradiente, hessiana e algoritmos de otimização

- Considere uma função contínua e diferenciável até 2ª ordem em todos os pontos do domínio de interesse, tal que: $f: \Re^n \rightarrow \Re$ e $\mathbf{x} \in \Re^n$.

- Expansão em série de Taylor em torno do ponto $\mathbf{x}^* \in \Re^n$:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^T (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \nabla^2 f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) + O(3)$$

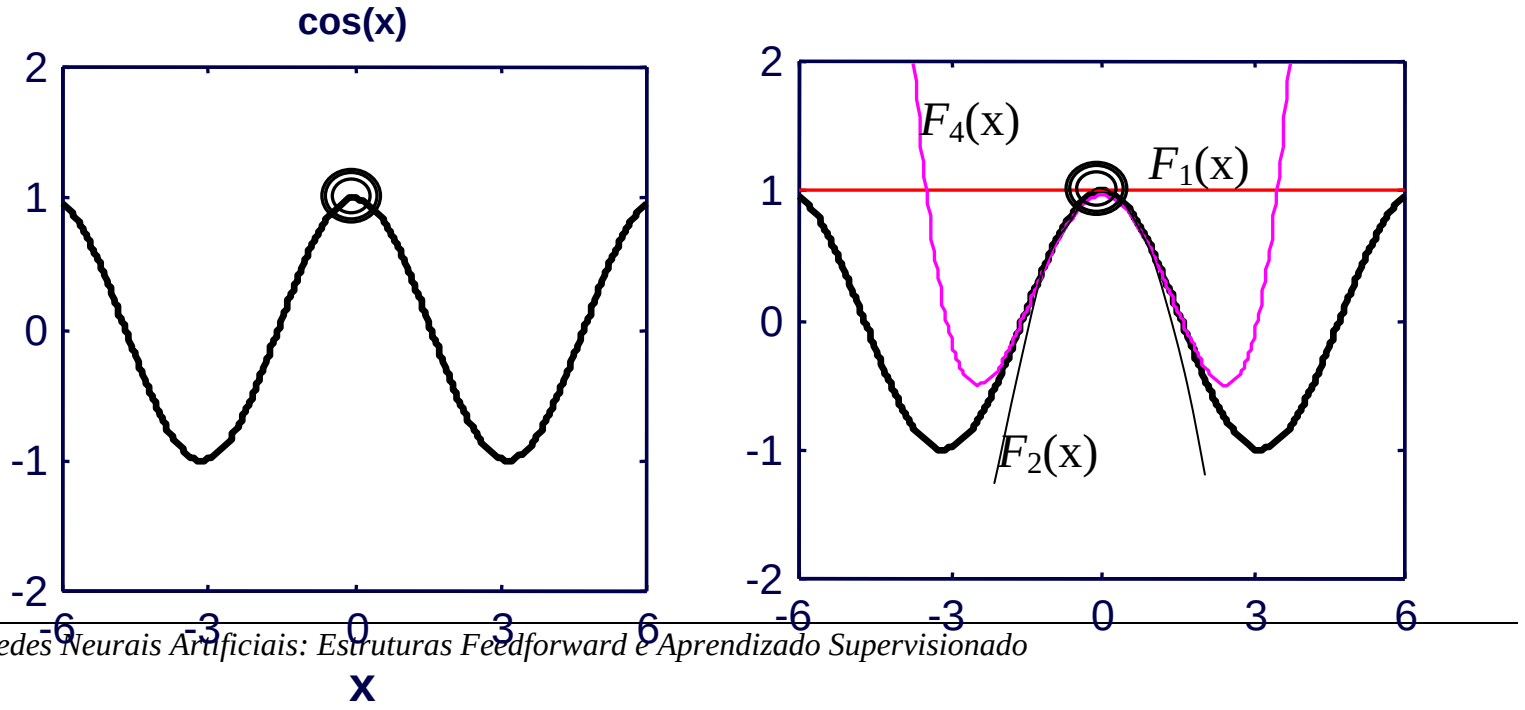
$$\nabla f(x^*) = \begin{bmatrix} \dfrac{\partial f(\mathbf{x}^*)}{\partial x_1} \\ \dfrac{\partial f(\mathbf{x}^*)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(\mathbf{x}^*)}{\partial x_n} \end{bmatrix} \qquad \nabla^2 f(\mathbf{x}^*) = \begin{bmatrix} \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_1^2} & \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_2^2} & & \vdots \\ & & \ddots & \\ \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_n \partial x_1} & \cdots & & \dfrac{\partial^2 f(\mathbf{x}^*)}{\partial x_n^2} \end{bmatrix}$$

Vetor gradiente **Matriz hessiana**

- O algoritmo de retropropagação do erro (do inglês, *backpropagation*) é empregado para obter o vetor gradiente, onde cada elemento do vetor gradiente está associado a

um peso da rede neural e indica o quanto a saída é influenciada por uma variação incremental neste peso sináptico.

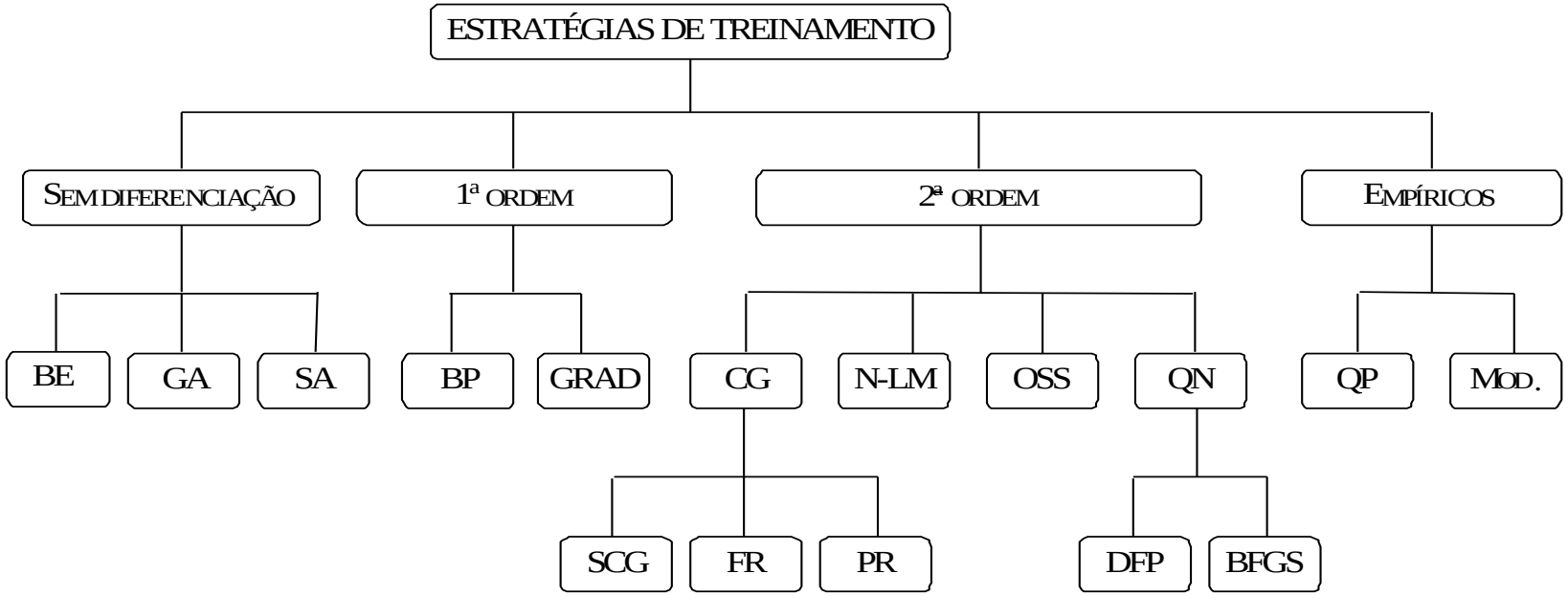
- Existem várias técnicas para obter exatamente ou aproximadamente a informação de 2ª ordem junto à superfície de erro produzida por redes neurais MLP (BATTITI, 1992; BISHOP, 1992).
- Na função cos(*x*) a seguir, observam-se as aproximações de primeira, segunda e quarta ordens em torno do ponto $x = 0$.



- O processo de otimização não-linear envolvido no ajuste de pesos de uma rede neural vai realizar aproximações locais de primeira ordem, ou de primeira e segunda ordem, junto à superfície de erro e realizar ajustes incrementais e recursivos na forma:

$$\theta_{k+1} = \theta_k + \text{passo}_k \times \text{direção}_k .$$

- Partindo de uma condição inicial θ_0, aplica-se iterativamente a fórmula acima, sendo que a direção depende da informação local de primeira e segunda ordem. Cada proposta de algoritmo de otimização vai diferir justamente na forma de computar o *passo* e a *direção* de ajuste, a cada iteração *k*.
- A figura a seguir apresenta uma classificação dos principais algoritmos empregados para o treinamento supervisionado de redes neurais artificiais.



- Figura 21 – Taxonomia de algoritmos de otimização para treinamento supervisionado de redes neurais MLP.

16.2 Mínimos locais

- Como o processo de ajuste é iterativo e baseado apenas em informações locais, os algoritmos de otimização geralmente convergem para o mínimo local mais próximo de

onde se encontra a busca, que pode representar uma solução inadequada (com nível de erro acima do aceitável).

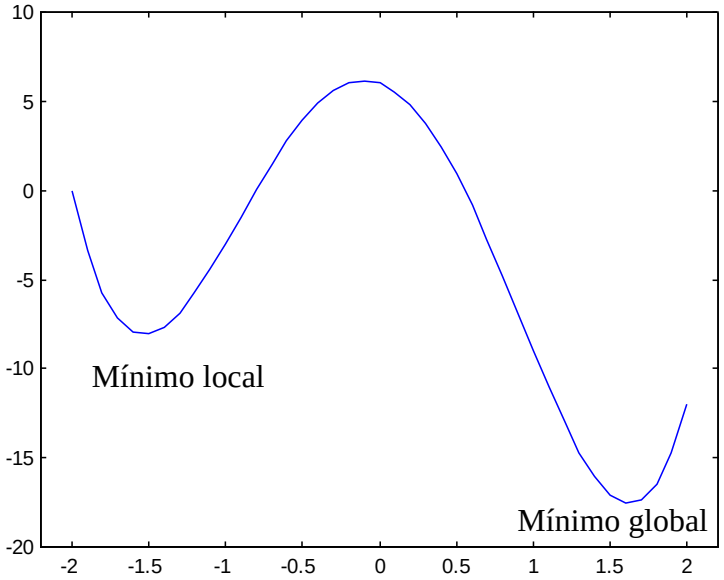


Figura 22 – Exemplo ilustrativo de mínimos local e global (considerando uma única variável).

- É importante destacar que algoritmos de 2ª ordem tendem a convergir mais rápido para os mínimos locais, mas não se pode afirmar que eles convergem para mínimos de melhor qualidade que aqueles produzidos pelos algoritmos de primeira ordem.

- Um conceito relevante aqui é o de **bacia de atração**. Todo mínimo local tem uma bacia de atração e o mínimo local a ser fornecido pelo algoritmo de otimização tende a ser aquele em cuja bacia de atração encontra-se a condição inicial (ponto de partida da busca iterativa).
- Isso é bem provável no caso de buscas iterativas que dão passos sempre minimizantes, mas podem ocorrer passos capazes de deslocar a busca para bacias de atração vizinhas, associadas a outros mínimos locais.
- A trajetória da condição inicial até o ótimo local resultante será certamente distinta para algoritmos de 1ª e 2ª ordem, pois eles diferem a cada iteração da busca, mas o resultado final tende a ser o mesmo, a menos que haja deslocamentos casuais para outras bacias de atração vizinhas.

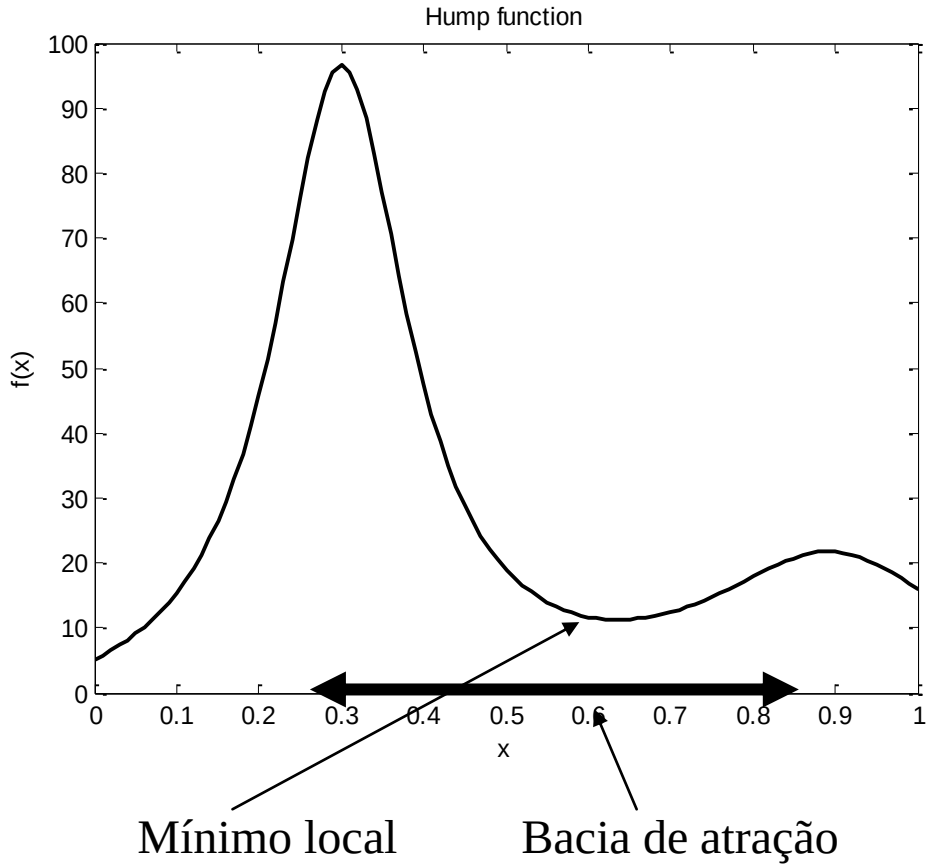


Figura 23 – Exemplo de bacia de atração de um mínimo local.

16.3 Condição inicial para os pesos da rede neural

- Embora existam técnicas mais elaboradas, os pesos da rede neural podem ser inicializados com valores pequenos e aleatoriamente distribuídos em torno de zero.
- Esta inicialização provê as seguintes propriedades à rede neural inicial:
  - O mapeamento realizado pela MLP tende a se aproximar de um hiperplano, não apresentando, assim, nenhuma tendência definida, em termos de comportamento não-linear;
  - A ativação de todos os neurônios vai se encontrar fora da região de saturação, facilitando o processo de ajuste de pesos a ser iniciado.
- Em termos práticos, pode-se afirmar que, com este procedimento de inicialização, o mapeamento produzido pela rede neural começa sem nenhuma contorção expressiva, mas com máxima capacidade de se contorcer de acordo com a demanda da aplicação.

16.4 Critério de parada

- O critério de parada mais empregado é o número máximo de épocas de treinamento, onde uma época é dada pela apresentação de todas as amostras do conjunto de treinamento à rede neural.
- No entanto, existem outros critérios que podem ser considerados conjuntamente ou em substituição ao número de épocas:
    - Módulo do vetor gradiente abaixo de um limiar;
    - Progresso do erro de treinamento abaixo de um limiar;
    - Grau de ajuste dos pesos sinápticos abaixo de um limiar.
- Esses limiares mencionados nos três itens acima devem ser definidos pelo usuário, havendo a necessidade de realização de alguns testes junto a cada aplicação pretendida.

17. Processo Iterativo para MLP – Método Padrão-a-Padrão

Obs1: Está sendo considerada a aplicação de um método de 1ª ordem para ajuste dos pesos.

Obs2: Este método e variações dele são conhecidas como *stochastic gradient descent* (DUCHI et al. 2011).

- Defina uma condição inicial para o vetor de pesos **w** e um passo α pequeno;
- Faça $k = 0$, $t = 0$ e calcule $J(\mathbf{w}(t))$;
- Enquanto o critério de parada não for atendido, faça:
    - ♦ Ordene aleatoriamente os padrões de entrada-saída;
    - ♦ Para l variando de 1 até N, faça:

        Apresente o padrão l de entrada à rede;

        Calcule $J_l(\mathbf{w}(t))$ e $\nabla J_l(\mathbf{w}(t))$;

        $\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha \nabla J_l(\mathbf{w}(t))$; $t = t + 1$;

    - ♦ $k = k + 1$;
    - ♦ Calcule $J(\mathbf{w}(t))$;

18. Processo Iterativo para MLP – Método em Lote ou Batelada

Obs: Está sendo considerada a aplicação de um método de 1ª ordem para ajuste dos pesos.

- Defina uma condição inicial para o vetor de pesos **w** e um passo α pequeno;
- Faça $k = 0$ e calcule $J(\mathbf{w}(k))$;
- Enquanto o critério de parada não for atendido, faça:
  - ♦ Para l variando de 1 até N, faça:

    Apresente o padrão l de entrada à rede;

    Calcule $J_l(\mathbf{w}(k))$ e $\nabla J_l(\mathbf{w}(k))$;
  - ♦ $\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\alpha}{N}\sum_{l=1}^{N}\nabla J_l(\mathbf{w}(k))$;
  - ♦ $k = k + 1$;
  - ♦ Calcule $J(\mathbf{w}(k))$;

19. O problema do OU-exclusivo em MLP

- Considere os pontos (0,0),(0,1),(1,0) e (1,1) no plano $\Re^2$, conforme apresentado na Figura 24. O objetivo é determinar uma rede com duas entradas $\mathbf{x}_i \in \{0,1\}$ (i=1,2), e uma saída $\mathbf{y} \in \{0,1\}$ de maneira que: $\begin{cases} (x_1,x_2) = (0,0) \text{ ou } (1,1) \Rightarrow y = 0 \\ (x_1,x_2) = (1,0) \text{ ou } (0,1) \Rightarrow y = 1 \end{cases}$

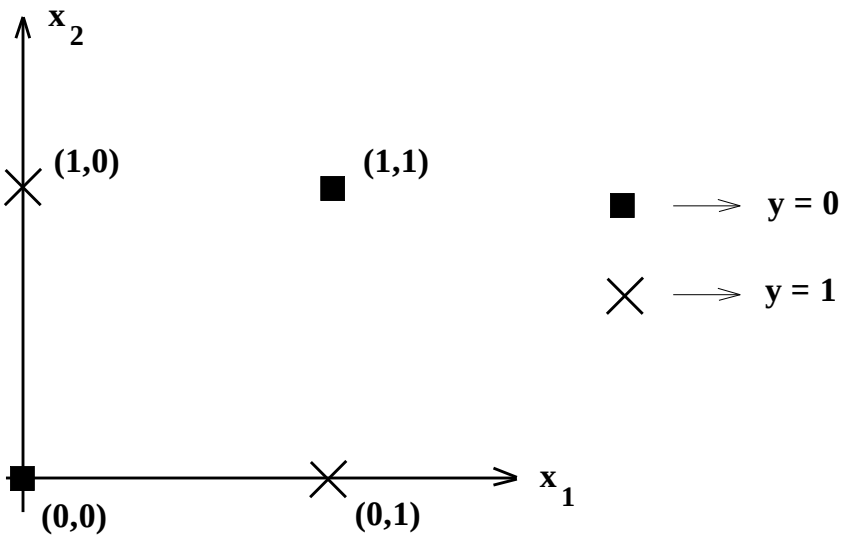


Figura 24 – O problema do OU-exclusivo.

- Inicialmente será analisado o comportamento de um único neurônio do tipo *perceptron* (veja Figura 25) no processo de solução do problema exposto acima. A saída *y* pode ser representada na forma:

$$y = \mathrm{g}(w_1 x_1 + w_2 x_2 + w_0) \quad \text{onde} \quad \begin{cases} \mathrm{g}(u) = 1 \ \text{se} \ u \geq 0 \\ \mathrm{g}(u) = 0 \ \text{se} \ u < 0 \end{cases}$$

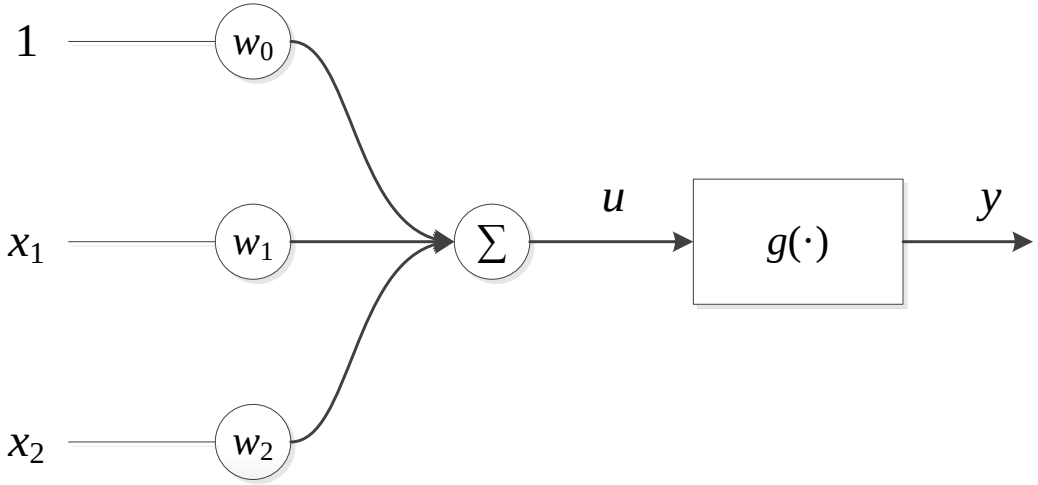


Figura 25 – Neurônio tipo *perceptron*, com duas entradas (mais a polarização).

- Para qualquer valor dos parâmetros w_0 , w_1 e w_2, a função $g(u)$ separa o espaço de entradas em duas regiões, sendo que a curva de separação é uma linha reta.

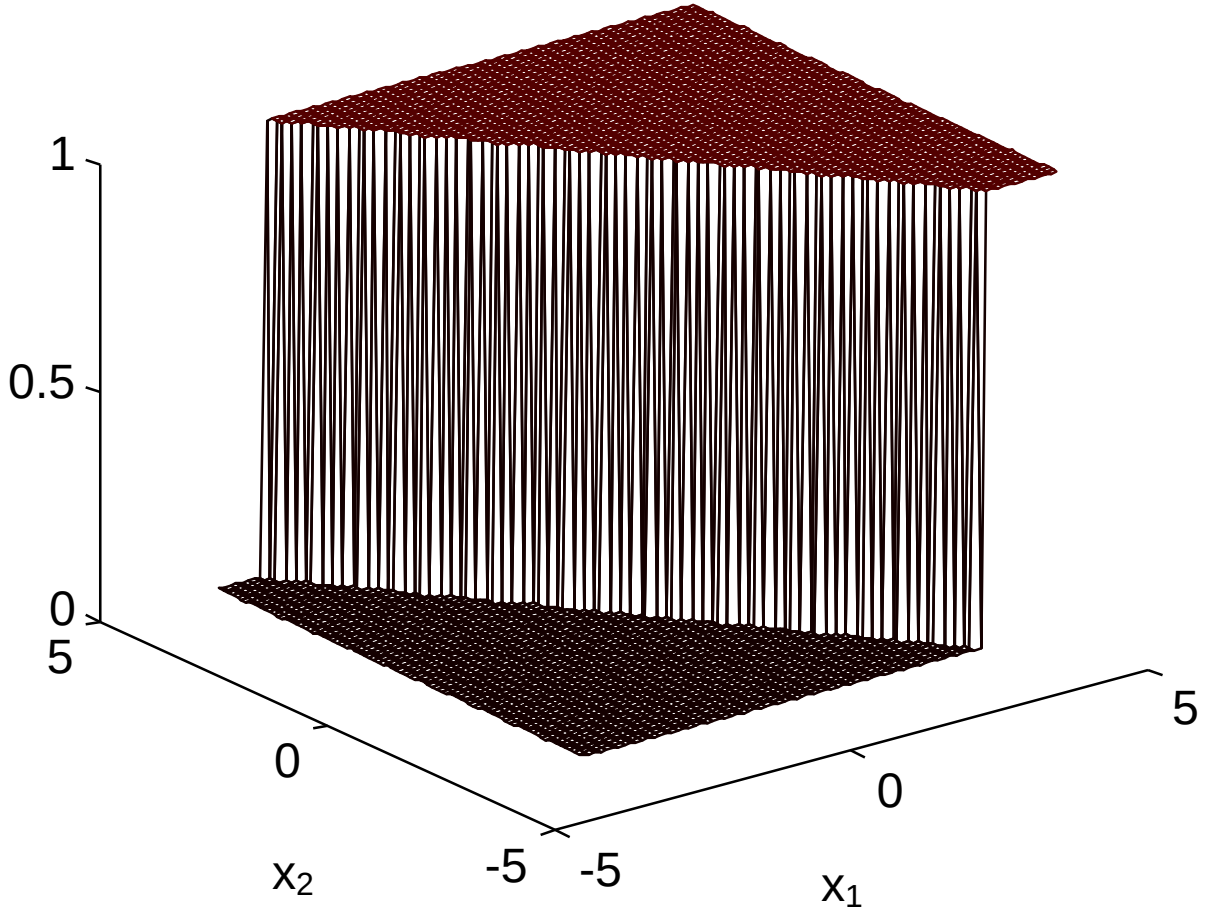


Figura 26 – Mapeamento de entrada-saída para o *perceptron* da Figura 25, com $w_0 = -6$, $w_1 = 4$ e $w_2 = 3$.

- Aqui tomou-se a função $g(\cdot)$ como sendo a função sinal, pois as saídas são binárias.

- No problema do OU-exclusivo (Figura 24), pode-se constatar que não existe uma única reta divisória de forma que os pontos (0,0) e (1,1) se posicionem de um lado enquanto que (0,1) e (1,0) permaneçam do outro lado da linha.
- Logo, pode-se imediatamente concluir que um neurônio tipo *perceptron* não apresenta grau de liberdade suficiente para resolver o problema proposto, o que foi corretamente constatado por Minsky & Papert, em 1969.
- No entanto, esses autores também acreditavam que não havia razão para supor que redes multicamadas pudessem conduzir a uma solução para o problema proposto. Esta hipótese só foi definitivamente rejeitada com a aplicação do algoritmo de retropropagação (*backpropagation*), já nos anos 80, o qual permite o ajuste automático de pesos para redes neurais multicamadas, arquitetura necessária para a realização de mapeamentos não-lineares.
- Considere o problema de mapeamento de uma rede neural tipo *perceptron*, com uma camada intermediária (Figura 27), aplicada ao problema do OU-exclusivo.

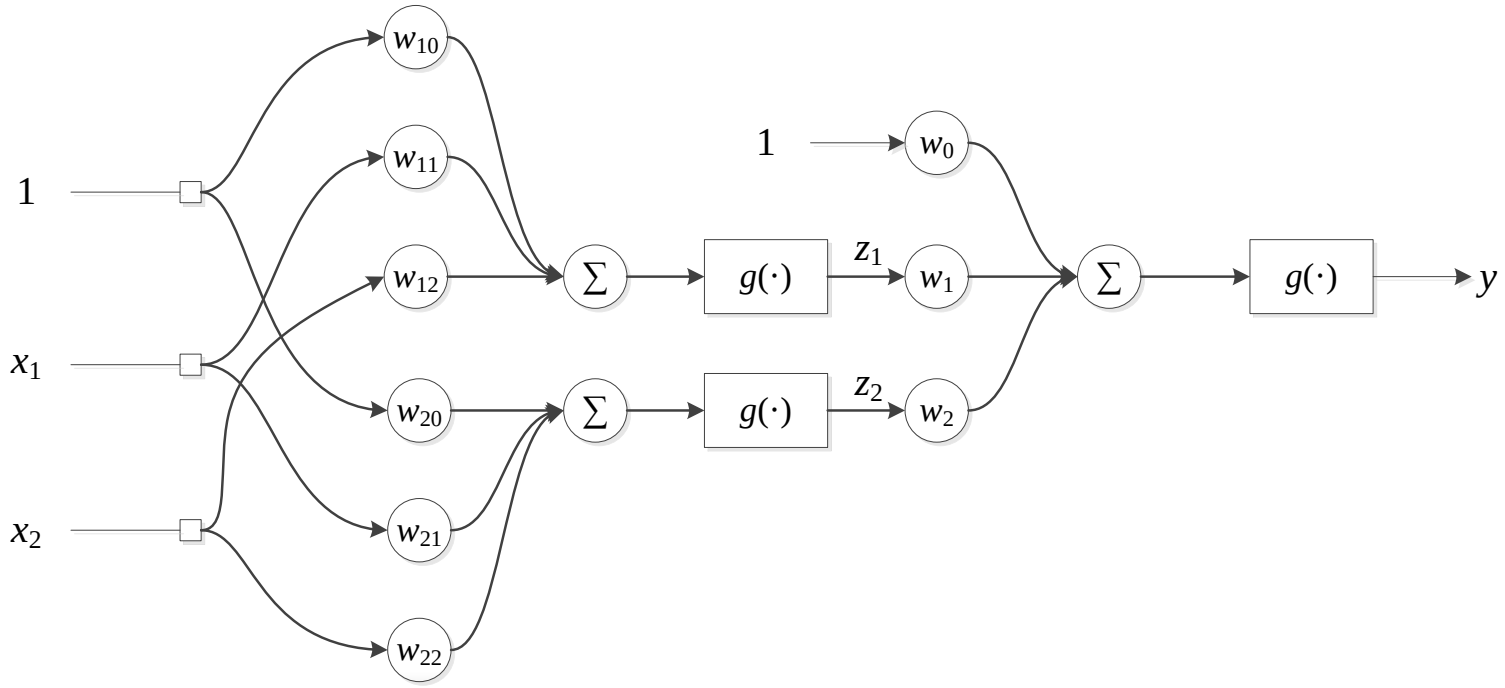


Figura 27 – *Perceptron* de três camadas (uma camada intermediária).

- A camada de entrada fornece um vetor de entrada (x_1,x_2) para a camada intermediária, enquanto que a camada intermediária produz duas saídas $z_1=\text{sgn}(w_{10}+w_{11}x_1+w_{12}x_2)$ e $z_2=\text{sgn}(w_{20}+w_{21}x_1+w_{22}x_2)$. Na camada de saída, o sinal de saída da rede neural é dado por $y=\text{sgn}(w_0+w_1z_1+w_2z_2)$.

- Surge uma questão: existem parâmetros w_{ij} (i=1,2; j=0,1,2) e w_k (k = 0,1,2) tais que y = 0 para as entradas (0,0) e (1,1) e y = 1 para as entradas (1,0) e (0,1)?
- As saídas da primeira camada (z_1 e z_2) podem ser consideradas como variáveis intermediárias utilizadas na geração da saída y. As variáveis z_1 e z_2 formam o que será denominado mais à frente no curso de espaço de características.
- Do que já foi visto a respeito de um neurônio do tipo *perceptron*, sabe-se que existem pesos w_{1j} (j=0,1,2) tais que (veja curva de separação $\mathbf{L_1}$ na Figura 28(a)):

$$(0,1) \text{ produza } z_1 = 1$$

$$(0,0),(1,0),(1,1) \text{ produza } z_1 = 0.$$

- De forma similar, existem pesos w_{2j} (j=0,1,2) tais que (veja curva de separação $\mathbf{L_2}$ na Figura 28(a)):

$$(0,1),(0,0),(1,1) \text{ produza } z_2 = 1$$

$$(1,0) \text{ produza } z_2 = 0$$

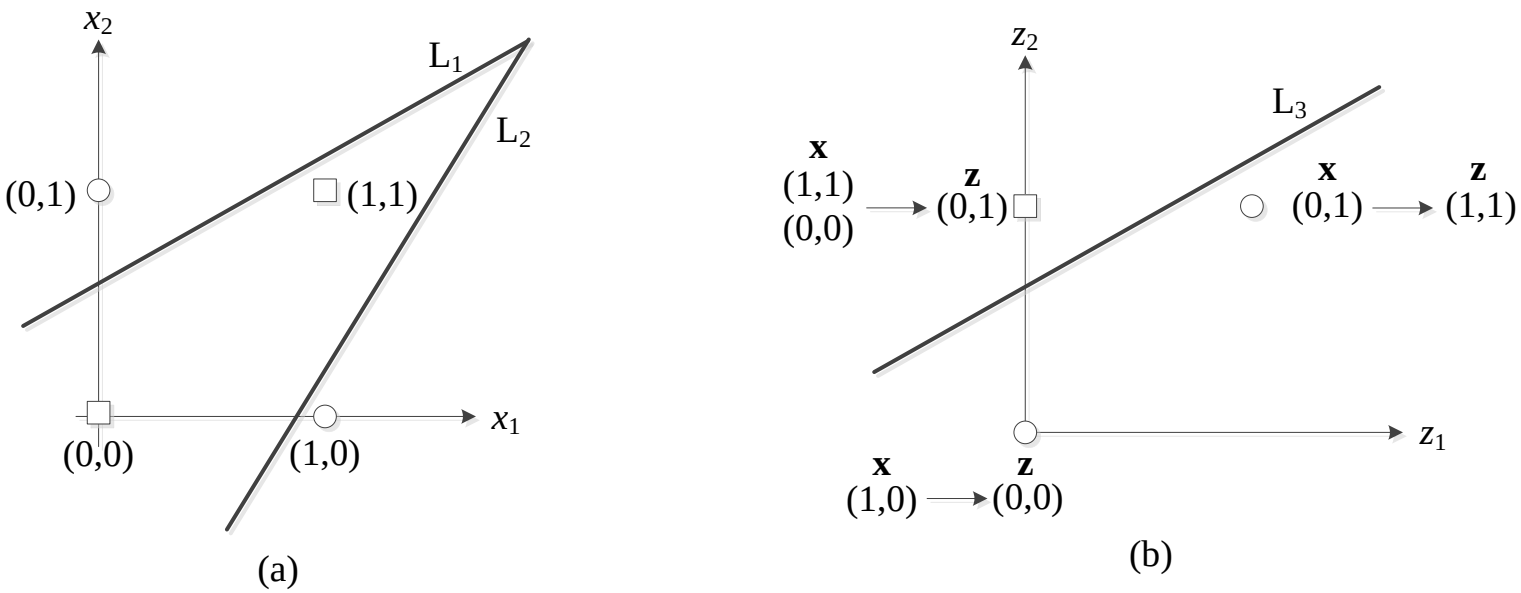


Figura 28 – Realização da função OU-exclusivo

A discussão acima mostra que existem pesos w_{ij} (i=1,2; j=0,1,2) de maneira que a entrada (0,1) resulte em $z_1 = 1$, $z_2 = 1$, e a entrada (1,0) resulte em $z_1 = 0$, $z_2 = 0$, enquanto que (0,0) e (1,1) produzam $z_1 = 0$, $z_2 = 1$. Já que (0,0) e (1,1) podem ser separados linearmente de (0,1), como mostrado na Figura 28(b) pela curva de separação $\mathbf{L_3}$, pode-se concluir que a função booleana desejada pode ser obtida utilizando-se *perceptrons* em cascata, ou seja, 3 neurônios do tipo *perceptron*.

20. Rede neural com função de ativação de base radial

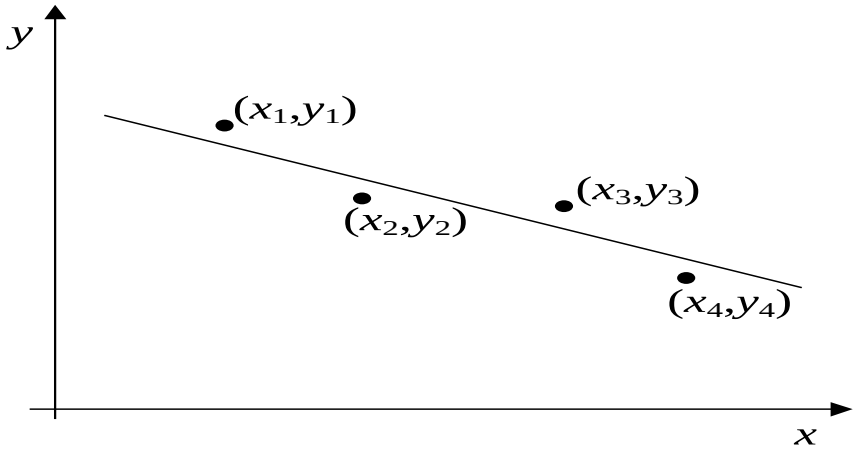
20.1 Regressão paramétrica e não-paramétrica

- Em problemas paramétricos, os parâmetros livres, bem como as variáveis dependentes e independentes, geralmente têm uma interpretação física.
- Exemplo: ajuste de uma reta a uma distribuição de pontos

$f(x) = y = ax + b$

a,b desconhecidos

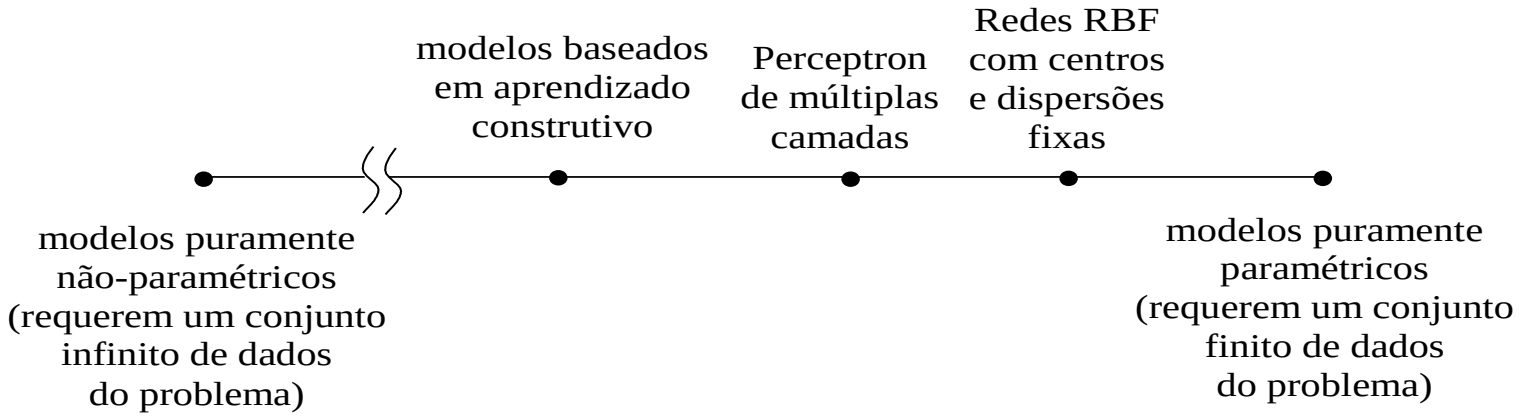
y: sujeito a ruído



- Regressão não-paramétrica: sua característica distintiva é a ausência (completa ou quase completa) de conhecimento a priori a respeito da forma da função que está sendo estimada. Sendo assim, mesmo que a função continue a ser estimada a partir do

ajuste de parâmetros livres, o conjunto de "formas" que a função pode assumir (classe de funções que o modelo do estimador pode prever) é muito amplo.

- Como consequência, vai existir um número elevado de parâmetros (por exemplo, quando comparado ao número de dados de entrada-saída para treinamento), os quais não mais admitem uma interpretação física isolada.



- Com base no exposto acima, fica evidente que redes neurais artificiais para treinamento supervisionado pertencem à classe de modelos de regressão não-paramétricos. Sendo assim, os pesos não apresentam um significado físico particular

em relação ao problema de aplicação. No entanto, certamente existem modelos de redes neurais "mais não-paramétricos" que outros.

- Além disso, estimar os parâmetros de um modelo não-paramétrico (por exemplo, pesos de uma rede neural artificial) não é o objetivo primário do aprendizado supervisionado. O objetivo primário é estimar a "forma" da função em uma região compacta do espaço de aproximação (ou ao menos a saída para certos valores desejados de entrada).
- Por outro lado, em regressão paramétrica, o objetivo primário é estimar o valor dos parâmetros, por dois motivos:
    1. A "forma" da função já é conhecida;
    2. Os parâmetros admitem uma interpretação física.

20.2 Funções de ativação de base radial

- Uma função de ativação de base radial é caracterizada por apresentar uma resposta que decresce (ou cresce) monotonicamente com a distância em relação a um ponto central.

- O centro e a taxa de decrescimento (ou crescimento) em cada direção são alguns dos parâmetros a serem definidos. Estes parâmetros devem ser constantes (i.e., pré-definidos e mantidos fixos) caso o modelo de regressão seja tomado como linear nos parâmetros ajustáveis.
- Uma função de base radial monotonicamente decrescente típica é a função gaussiana, dada na forma:
  - $h_j(x) = \exp\left(-\frac{(x-c_j)^2}{r_j^2}\right)$, para o caso escalar (veja Figura 29(a));
- Uma função de base radial monotonicamente crescente típica é a função multiquádrica dada na forma:
  - $h_j(x) = \frac{\sqrt{r_j^2 + (x-c_j)^2}}{r_j}$, para o caso escalar (veja Figura 29(b));

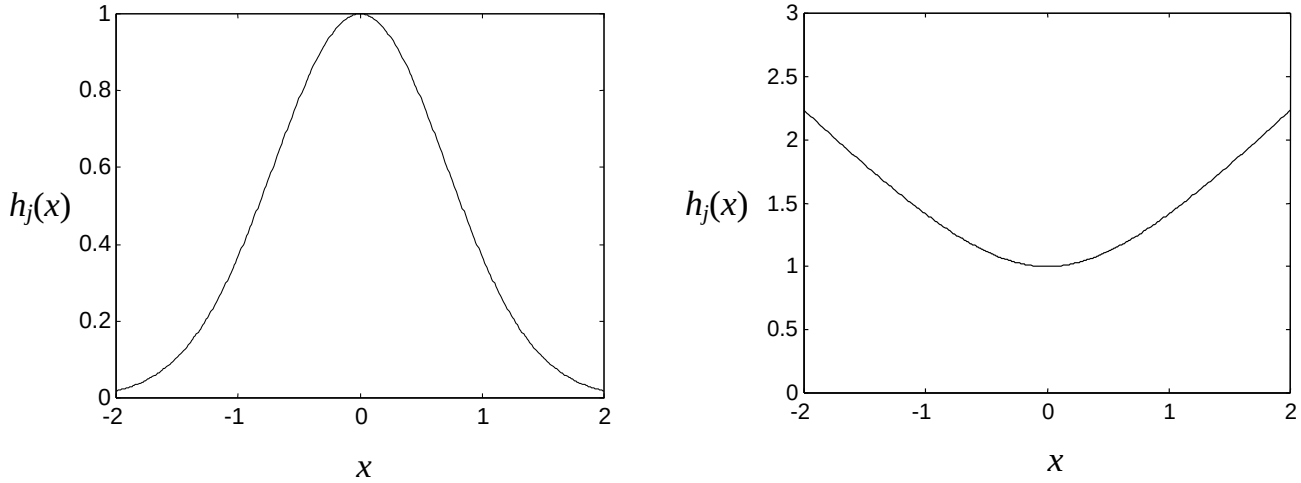
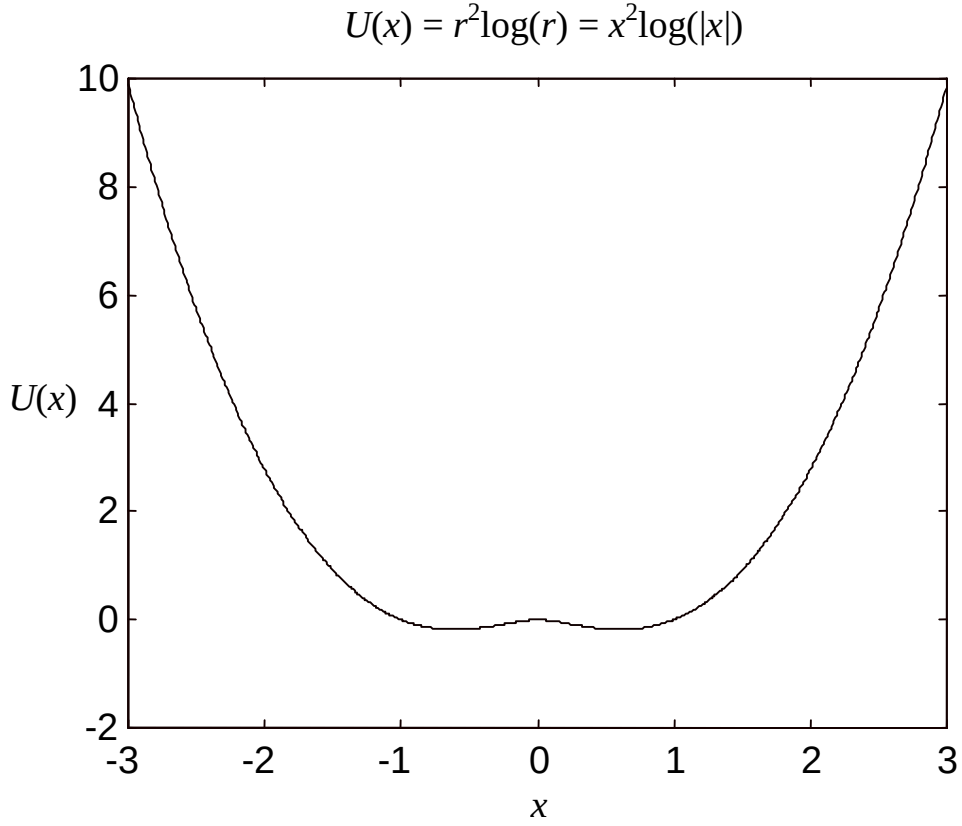


Figura 29 – Exemplos de funções de base radial monovariáveis, com $c_j = 0$ e $r_j = 1$.

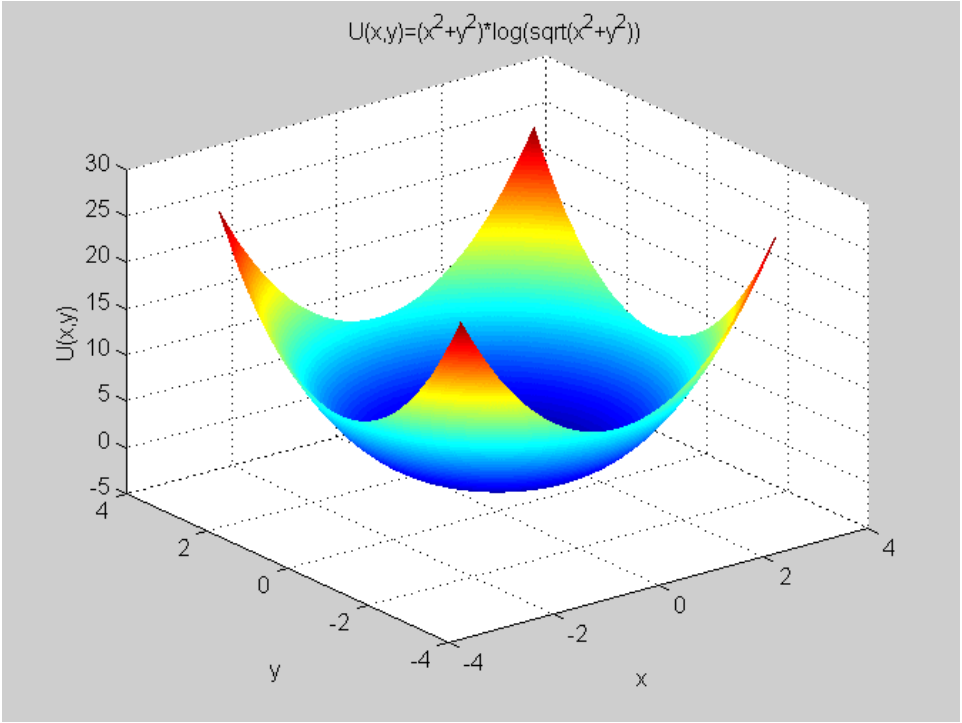
- Também apresenta propriedades interessantes a função *thin plate spline*. Considerando uma e duas variáveis, ela assume a forma:

$$h_j(x) = U(x) = (x - c_j)^2 \log(|x - c_j|)$$



Thin plate spline considerando $c_j = 0$.

$$h_j(x,y) = \left((x-x_j)^2 + (y-y_j)^2\right)\log\left(\sqrt{(x-x_j)^2 + (y-y_j)^2}\right)$$



Thin plate spline considerando $(x_j, y_j) = (0,0)$.

- No caso multidimensional e tomando a função gaussiana, $h_j(\mathbf{x})$ assume a forma:

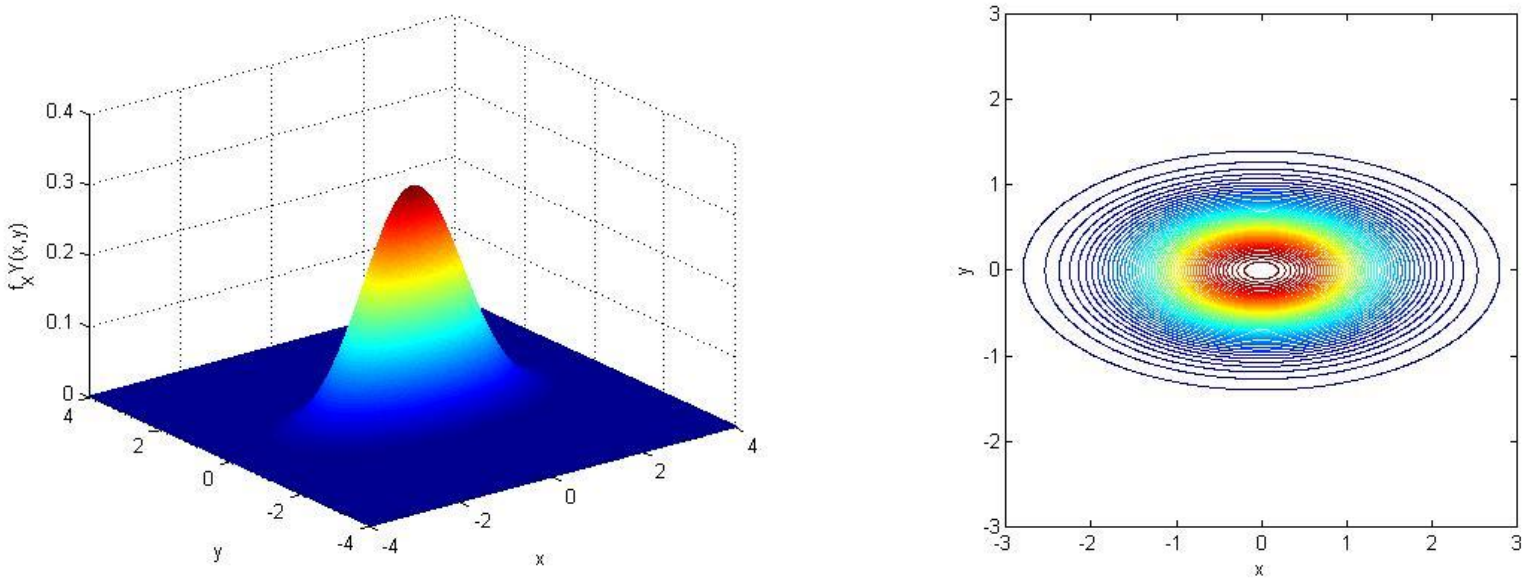
$$h_j(\mathbf{x}) = \exp\left(-\left(\mathbf{x}-\mathbf{c}_j\right)^T \Sigma_j^{-1}\left(\mathbf{x}-\mathbf{c}_j\right)\right) \qquad (1)$$

onde $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$ é o vetor de entradas, $\mathbf{c}_j = \begin{bmatrix} c_{j1} & c_{j2} & \cdots & c_{jn} \end{bmatrix}^T$ é o vetor que define o centro da função de base radial e a matriz Σ_j é definida positiva e diagonal, dada por:

$$\Sigma_j = \begin{bmatrix} \sigma_{j1} & 0 & \cdots & 0 \\ 0 & \sigma_{j2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{jn} \end{bmatrix},$$

de modo que $h_j(\mathbf{x})$ pode ser expandida na forma:

$$h_j(\mathbf{x}) = \exp\left(-\frac{\left(x_1-c_{j1}\right)^2}{\sigma_{j1}} - \frac{\left(x_2-c_{j2}\right)^2}{\sigma_{j2}} - \ldots - \frac{\left(x_n-c_{jn}\right)^2}{\sigma_{jn}}\right). \qquad (2)$$



Função gaussiana multivariada: $\mathbf{c}_j = [0\ 0]^T$ e $\boldsymbol{\sigma}_j = [1\ 0{,}5]^T$.

- Neste caso, os elementos do vetor $\boldsymbol{\sigma}_j = [\sigma_{j1} \cdots \sigma_{jn}]^T$ são responsáveis pela taxa de decrescimento da gaussiana junto a cada coordenada do espaço de entrada, e o argumento da função exponencial é uma norma ponderada da diferença entre o vetor de entrada ($\mathbf{x}$) e o centro da função de base radial ($\mathbf{c}_j$).

- A matriz Σ_j pode ser não-diagonal, mas ainda simétrica e definida positiva. No entanto, por envolver muitos parâmetros, esta opção geralmente não é utilizada.

20.3 Rede Neural RBF (*Radial Basis Function Neural Network*)

- As funções de base radial (são funções não-lineares) podem ser utilizadas como funções-base em qualquer tipo de modelo de regressão não-linear (linear ou não-linear nos parâmetros) e, particularmente, como função de ativação de qualquer tipo de rede multicamada.
- O fato do modelo de regressão resultante ser linear ou não-linear nos parâmetros se deve à possibilidade ou não de se ajustar os centros e as dispersões das funções.
- As redes neurais com função de ativação de base radial (em inglês, *radial-basis function*, RBF) apresentam **três** diferenças principais em relação às redes MLP:
    1) Elas sempre apresentam uma única camada intermediária;
    2) Os neurônios de saída são sempre lineares;

3) Os neurônios da camada intermediária têm uma função de base radial como função de ativação, ao invés de uma função sigmoidal ou outras.

- Como exposto acima, se apenas os pesos da camada de saída formarem o conjunto de parâmetros ajustáveis, então a rede neural será linear nos parâmetros. Caso contrário, ou seja, quando os centros $\mathbf{c}_j$ e as matrizes Σ_j, $j = 1,...,n$, também são ajustáveis, a rede neural se torna não-linear nos parâmetros, admitindo o próprio algoritmo de retropropagação do erro para o processo de ajuste via treinamento supervisionado, como feito no caso das MLPs.
- Contudo, no caso de redes RBFs, a questão dos mínimos locais tem uma influência bem maior e sugere-se evitar este mecanismo de ajuste.
- A arquitetura da rede é apresentada na Figura 30, para o caso de uma única saída, resultando no seguinte mapeamento de entrada-saída:

$$y = \sum_{j=1}^{m} w_j h_j(\mathbf{x})$$

- Caso $\mathbf{c}_j$ e Σ_j, $j = 1,...,n$, sejam ajustáveis, a saída assume a forma:

$$y = \sum_{j=1}^{m} w_j h_j(\mathbf{c}_j, \Sigma_j, \mathbf{x}).$$

- Substituindo as formas compactas e expandidas de $h_j(\mathbf{x})$, dadas respectivamente pelas equações (1) e (2), resultam:

$$y = \sum_{j=1}^{m} w_j \exp\left(-\left(\mathbf{x}-\mathbf{c}_j\right)^T \Sigma_j^{-1}\left(\mathbf{x}-\mathbf{c}_j\right)\right)$$

e

$$y = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left(x_1 - c_{j1}\right)^2}{\sigma_{j1}} - \frac{\left(x_2 - c_{j2}\right)^2}{\sigma_{j2}} - \ldots - \frac{\left(x_n - c_{jn}\right)^2}{\sigma_{jn}}\right)$$

- Uma versão para múltiplas saídas é apresentada na Figura 31.

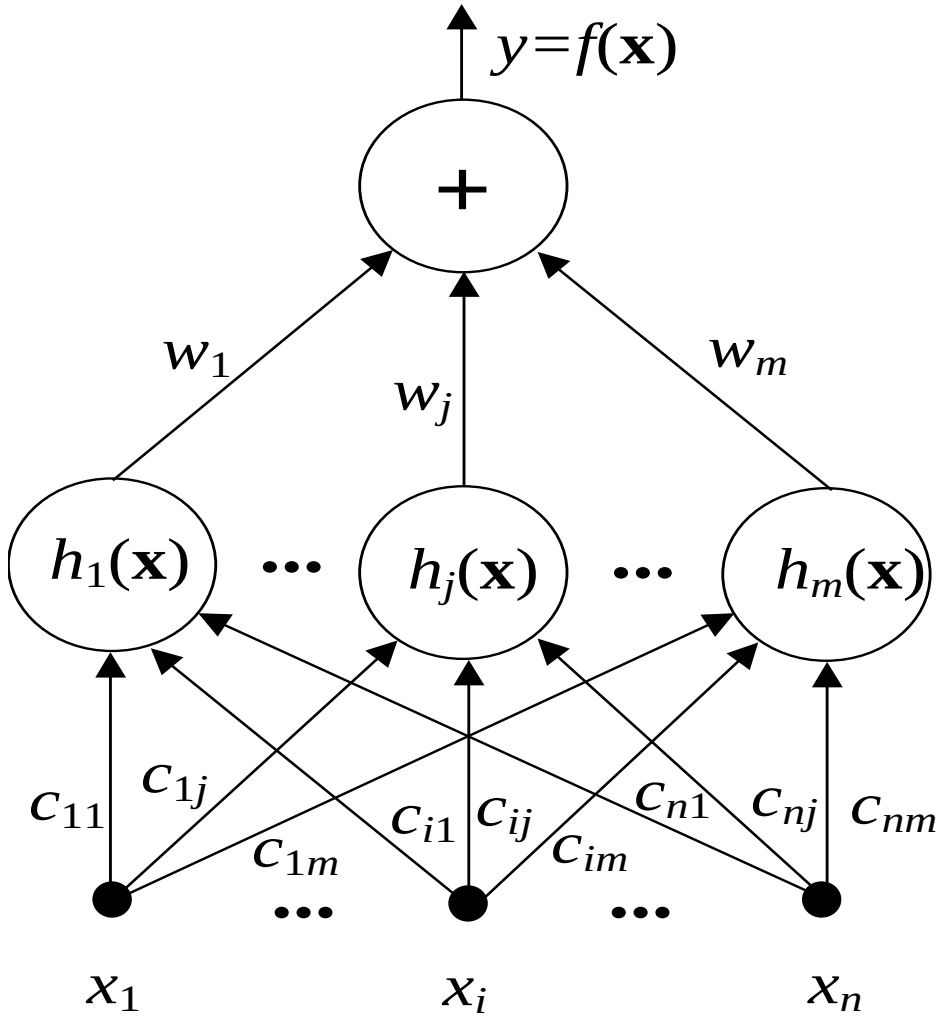


Figura 30 – Rede neural de base radial (BROOMHEAD & LOWE, 1988).

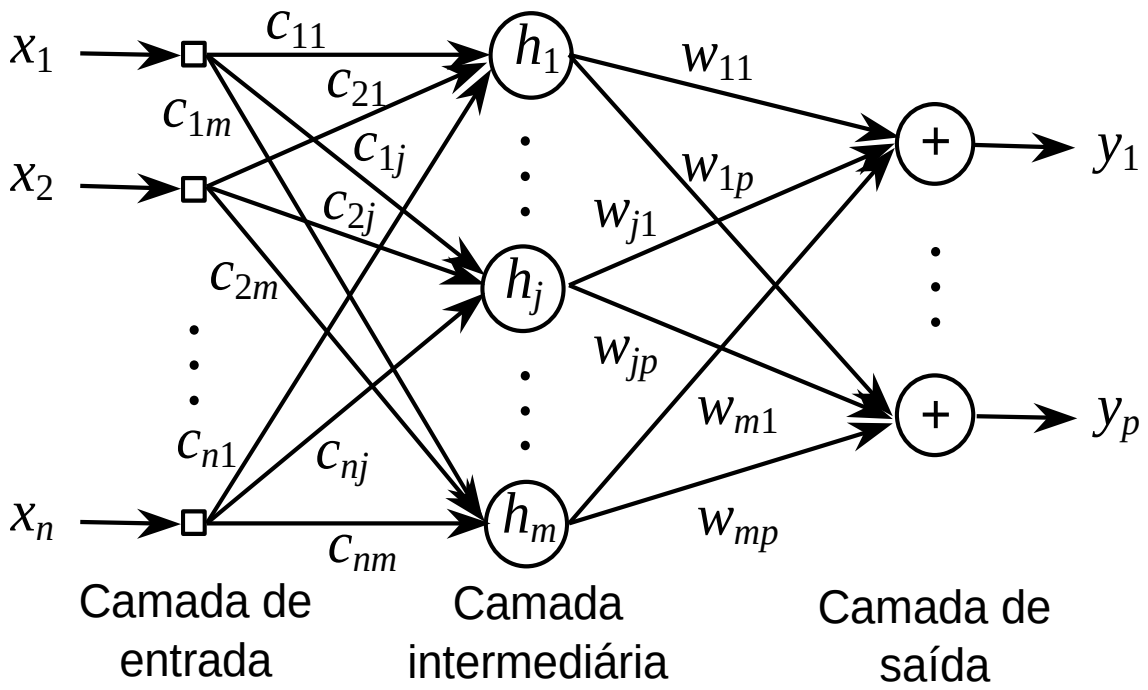


Figura 31 – Rede RBF com múltiplas saídas.

- Ao invés da ativação interna de cada neurônio da camada intermediária se dar pelo emprego do produto escalar (produto interno) entre o vetor de entradas e o vetor de pesos, como no caso do *perceptron*, ela é obtida a partir de uma norma ponderada da diferença entre ambos os vetores.

- Dado um número suficiente de neurônios com função de base radial, qualquer função contínua definida numa região compacta pode ser devidamente aproximada usando uma rede RBF (PARK & SANDBERG, 1991).
- As redes RBF são redes de aprendizado local, de modo que é possível chegar a uma boa aproximação desde que um número suficiente de dados para treinamento seja fornecido na região de interesse.
- Em contraste, *perceptrons* multicamadas são redes de aprendizado "global" (em virtude da natureza das funções de ativação: são *ridge functions*) que fazem aproximações de efeito global, em regiões compactas do espaço de aproximação.
- Redes neurais com capacidade de aproximação local são muito eficientes quando a dimensão do vetor de entradas é reduzida. Entretanto, quando o número de entradas não é pequeno, as redes MLP apresentam uma maior capacidade de generalização (HAYKIN, 2008).

- Isto ocorre porque o número de funções de base radial deve aumentar exponencialmente com o aumento da dimensão da entrada para termos uma boa cobertura do espaço em que os dados se encontram.

20.4 O problema dos quadrados mínimos para aproximação de funções

- Considere o problema de encontrar os coeficientes de um polinômio de grau *n* que interpole *n*+1 pontos:

$$c_0 + c_1 x_j + \cdots + c_n x_j^n = y_j, \quad j = 0,1,\ldots,n,$$

- Estas *n*+1 equações com *n*+1 incógnitas podem ser colocadas na forma matricial:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \Rightarrow \mathbf{X} \cdot \mathbf{c} = \mathbf{y}$$

- Os coeficientes que compõem o vetor **c** são obtidos simplesmente fazendo $\mathbf{c} = \mathbf{X}^{-1}\mathbf{y}$. Para a existência da inversa, basta que todas as colunas de **X** sejam LI, ou seja, que **X** tenha posto completo.
- Um problema de interpolação equivalente é o de encontrar os n+1 coeficientes de uma composição aditiva de funções-base que interpole n+1 pontos:

$$\sum_{j=0}^{n} c_j f_j(x_i) = y_i, \quad i = 0,1,\ldots,n$$

$$\begin{bmatrix} f_0(x_0) & f_1(x_0) & \cdots & f_n(x_0) \\ f_0(x_1) & \ddots & & \vdots \\ \vdots & & & \\ f_0(x_n) & \cdots & & f_n(x_n) \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \Rightarrow \mathbf{F} \cdot \mathbf{c} = \mathbf{y} .$$

- Particularmente no caso de problemas de aproximação de funções em que os parâmetros ajustáveis influenciam linearmente na saída do modelo, como no caso de redes neurais RBF com ajuste apenas dos pesos da camada de saída, resultam

problemas parecidos com os dois problemas de interpolação acima, mas agora o número de amostras tende a ser bem maior que o número de parâmetros a determinar. Assim, o equivalente às matrizes **X** e **F** terá bem mais linhas que colunas.

- A razão para lidarmos agora com problemas de aproximação e não de interpolação é de duas naturezas:
  - Os dados disponíveis geralmente estão sujeitos a ruído;
  - Como a forma final do mapeamento não é conhecida, os dados disponíveis não contêm toda a informação necessária sobre o mapeamento.
- Assim, com um modelo de regressão linear na forma (considerando uma saída):

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j h_j(\mathbf{x})$$

e o conjunto de treinamento dado por $\{(\mathbf{x}_i, s_i)\}_{i=1}^{N}$, o método dos quadrados mínimos se ocupa em minimizar (em relação aos coeficientes da combinação linear) a soma dos quadrados dos erros produzidos a partir de cada um dos *N* padrões de entrada-saída.

$$\min_{\mathbf{w}} J(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N} \left(s_i - f(\mathbf{x}_i)\right)^2 = \min_{\mathbf{w}} \sum_{i=1}^{N} \left(s_i - \sum_{j=1}^{m} w_j h_j(\mathbf{x}_i) \right)^2$$

- Do cálculo elementar sabe-se que a aplicação da condição de otimalidade (restrições atendidas pelos pontos de máximo e mínimo de uma função diferenciável) permite obter a solução ótima do problema de otimização $\min_{\mathbf{w}} J(\mathbf{w})$, na forma:

    1. Diferencie a função em relação aos parâmetros ajustáveis;
    2. Iguale estas derivadas parciais a zero;
    3. Resolva o sistema de equações resultante.

- No caso em questão, os parâmetros livres são os coeficientes da combinação linear, dados na forma do vetor de pesos $\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_j & \cdots & w_m \end{bmatrix}^T$.

- O sistema de equações resultante é dado na forma:

$$\frac{\partial J}{\partial w_j} = -2 \sum_{i=1}^{N} \left(s_i - f(\mathbf{x}_i)\right) \frac{\partial f}{\partial w_j} = -2 \sum_{i=1}^{N} \left(s_i - f(\mathbf{x}_i)\right) h_j(\mathbf{x}_i) = 0, \quad j=1,...,m.$$

- Separando-se os termos que envolvem a incógnita $f(\cdot)$, resulta:

$$\sum_{i=1}^{N} f(\mathbf{x}_i)h_j(\mathbf{x}_i) = \sum_{i=1}^{N}\left[\sum_{r=1}^{m} w_r h_r(\mathbf{x}_i)\right]h_j(\mathbf{x}_i) = \sum_{i=1}^{N} s_i h_j(\mathbf{x}_i), \quad j=1,...,m.$$

- Portanto, existem m equações para obter as m incógnitas $\{w_r, r=1,...,m\}$. Exceto sob condições "patológicas", este sistema de equações vai apresentar uma solução única.
- Para encontrar esta solução única do sistema de equações lineares, é interessante recorrer à notação vetorial, fornecida pela álgebra linear, para obter:

$$\mathbf{h}_j^T\mathbf{f} = \mathbf{h}_j^T\mathbf{s}, \quad j=1,...,m,$$

onde

$$\mathbf{h}_j = \begin{bmatrix} h_j(\mathbf{x}_1) \\ \vdots \\ h_j(\mathbf{x}_N) \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \sum_{r=1}^{m} w_r h_r(\mathbf{x}_1) \\ \vdots \\ \sum_{r=1}^{m} w_r h_r(\mathbf{x}_N) \end{bmatrix} \quad \text{e} \quad \mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_N \end{bmatrix}.$$

- Como existem m equações, resulta:

$$\begin{bmatrix} \mathbf{h}_1^T\mathbf{f} \\ \vdots \\ \mathbf{h}_m^T\mathbf{f} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1^T\mathbf{s} \\ \vdots \\ \mathbf{h}_m^T\mathbf{s} \end{bmatrix}$$

- Definindo a matriz **H**, com sua *j*-ésima coluna dada por $\mathbf{h}_j$, temos:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_m \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \cdots & h_m(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \cdots & h_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & h_2(\mathbf{x}_N) & \cdots & h_m(\mathbf{x}_N) \end{bmatrix}$$

sendo possível reescrever o sistema de equações lineares como segue:

$$\mathbf{H}^T\mathbf{f} = \mathbf{H}^T\mathbf{s}$$

- O *i*-ésimo componente do vetor **f** pode ser apresentado na forma:

$$f_i = f(\mathbf{x}_i) = \sum_{r=1}^{m} w_r h_r(\mathbf{x}_i) = \begin{bmatrix} h_1(\mathbf{x}_i) & h_2(\mathbf{x}_i) & \cdots & h_m(\mathbf{x}_i) \end{bmatrix}\mathbf{w}$$

permitindo expressar **f** em função da matriz **H**, de modo que:

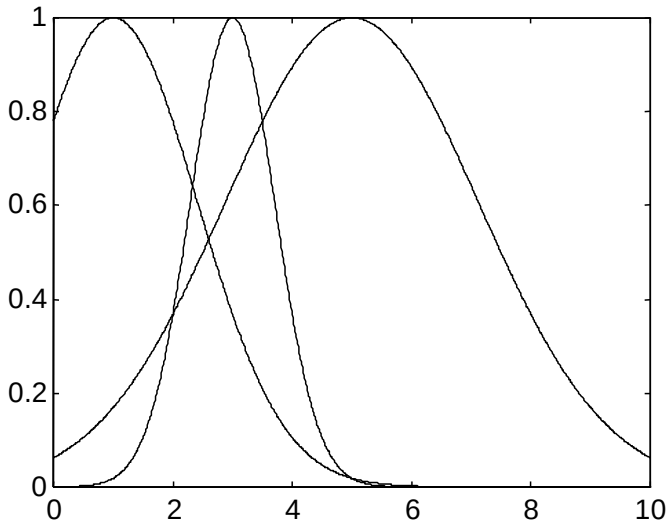
$$\mathbf{f} = \mathbf{H}\mathbf{w}$$

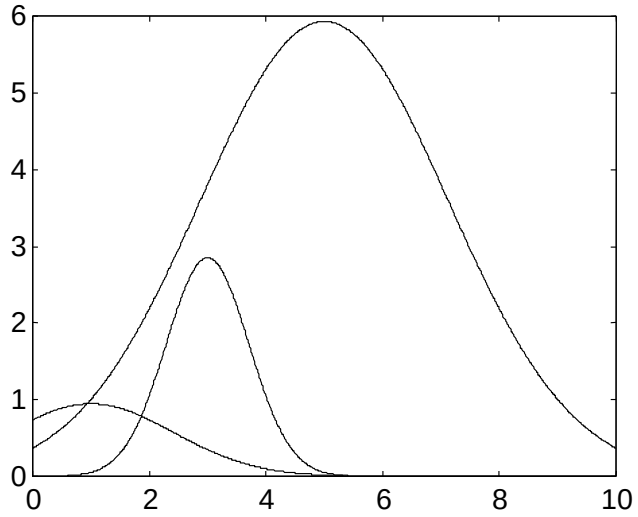
- Substituindo no sistema de equações lineares, resulta a solução ótima para o vetor de coeficientes da combinação linear (que correspondem aos pesos da camada de saída da rede neural de base radial):

$$\mathbf{H}^T\mathbf{H}\mathbf{w} = \mathbf{H}^T\mathbf{s} \Rightarrow \mathbf{w} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{s}$$

- Esta equação de solução do problema dos quadrados mínimos é conhecida como *equação normal*. Para que exista a inversa de $\mathbf{H}^T\mathbf{H}$, basta que a matriz $\mathbf{H}$ tenha posto completo, já que $m \leq N$.
- A denominação de equação normal se deve ao fato de que $\mathbf{e} = \mathbf{H}\mathbf{w} - \mathbf{s}$, que é o erro cuja norma deve ser minimizada, é um vetor ortogonal (ou normal) ao espaço gerado pelas colunas de $\mathbf{H}$, sendo que $\mathbf{H}\mathbf{w}$ representa qualquer vetor neste espaço.

20.5 Exemplo didático 1



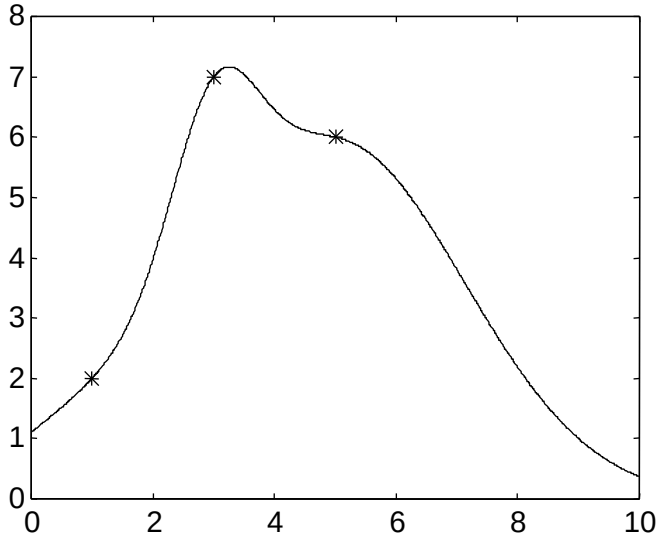


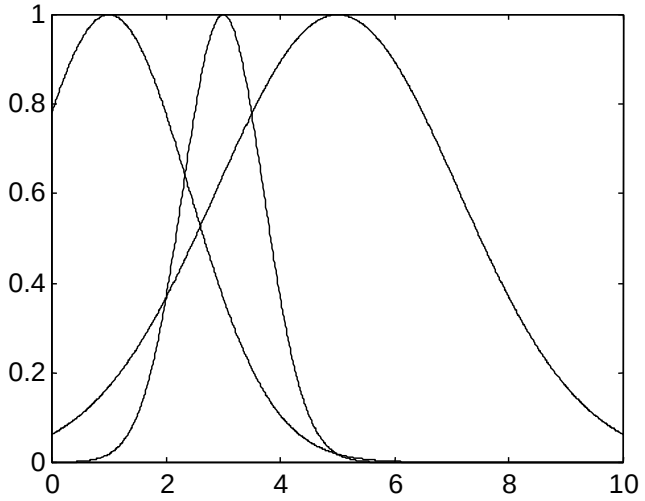
Caso 1: $m = N$

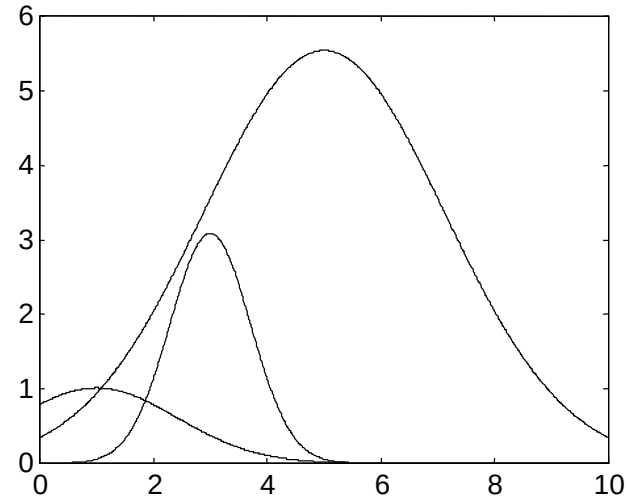
Pontos amostrados: (1,2); (3,7); (5,6)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 0.945 \\ 2.850 \\ 5.930 \end{bmatrix}$$

Obs: As funções de base radial têm centros nos valores de x e dispersões arbitrariamente definidas.





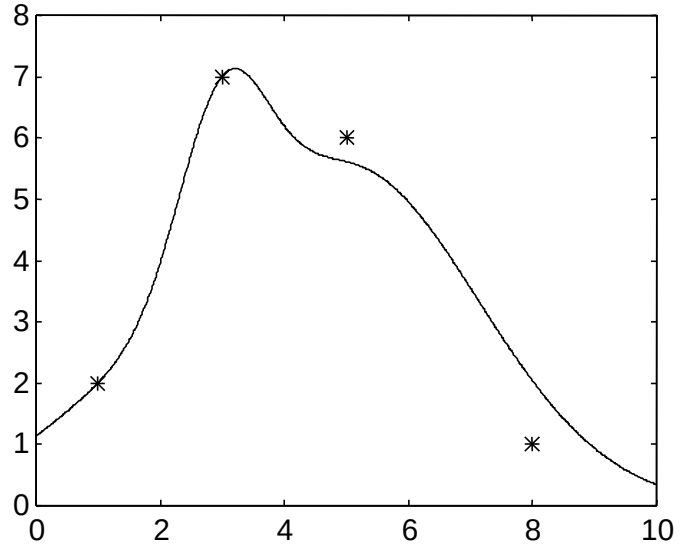


Caso 2: $m < N$

Pontos amostrados: (1,2); (3,7); (5,6); (8,1)

$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 1.012 \\ 3.084 \\ 5.538 \end{bmatrix}$$

Obs: As funções de base radial são as mesmas do Caso 1.



20.6 Definição da dispersão

- Quanto às dispersões das funções de base radial, embora elas possam ser distintas (e até ajustáveis) para cada centro, usualmente se adota uma única dispersão para todos os centros, na forma (HAYKIN, 2008):

$$\sigma = \frac{d_{\max}}{\sqrt{2m}},$$

  onde m é o número de centros, e $d_{\max}$ é a distância máxima entre os centros.

- Com este critério de dispersão, evita-se que as funções de base radial sejam excessivamente pontiagudas, ou então com uma base demasiadamente extensa.

20.7 Seleção dos centros por auto-organização

- Para auto-organizar os centros, é suficiente aplicar algum algoritmo capaz de refletir a distribuição dos dados de entrada.

- O algoritmo a ser apresentado a seguir corresponde ao método de clusterização denominado *k*-means (COVER & HART, 1967; MACQUEEN, 1967). Este algoritmo se

assemelha ao adotado para as redes de Kohonen, mas não leva em conta noções de vizinhança entre os centros. **Ele também está sujeito a mínimos locais**.

- Sejam $\{\mathbf{c}_j(t)\}_{j=1}^{m}$ os centros das funções de base radial na iteração *t*. O algoritmo *k*-means padrão-a-padrão pode ser descrito da seguinte forma:

  1. *Inicialização*: Escolha valores aleatórios distintos para os centros $\mathbf{c}_j(t)$.
  2. *Amostragem*: Tome aleatoriamente um vetor $\mathbf{x}_i$ do conjunto de padrões de entrada;
  3. *Matching*: Determine o índice *k* do centro que mais se aproxima deste padrão, na forma: $k(\mathbf{x}_i) = \arg\min_j \|\mathbf{x}_i(t) - \mathbf{c}_j(t)\|, \quad j = 1,...,m.$
  4. *Ajuste*: Ajuste os centros usando a seguinte regra:

$$\mathbf{c}_j(t+1) = \begin{cases} \mathbf{c}_j(t) + \gamma[\mathbf{x}_i(t) - \mathbf{c}_j(t)], & k = k(\mathbf{x}_i) \\ \mathbf{c}_j(t), & \text{alhures} \end{cases}$$

  onde $\gamma \in (0,1)$ é a taxa de ajuste.

5. *Ciclo*: Repita os Passos 2 a 5 para todos os N padrões de entrada e até que os centros não apresentem deslocamento significativo após cada apresentação completa dos N padrões.

- O algoritmo *k*-means em batelada pode ser descrito da seguinte forma:

1. *Inicialização*: Escolha valores aleatórios distintos para os centros $\mathbf{c}_j(t)$.
2. *Matching*: Determine o índice k do centro que mais se aproxima de cada padrão, na forma: $k(\mathbf{x}_i) = \arg\min_j \|\mathbf{x}_i(t) - \mathbf{c}_j(t)\|, \quad j = 1,...,m$.
3. *Ajuste*: Defina a nova posição de cada um dos m centros como a média dos padrões cujo índice corresponde àquele centro.
4. *Ciclo*: Repita os Passos 2 e 3 enquanto houver mudança de índice de centro para pelo menos um dos padrões.

20.8 Aplicação das propostas de determinação de centros e dispersão

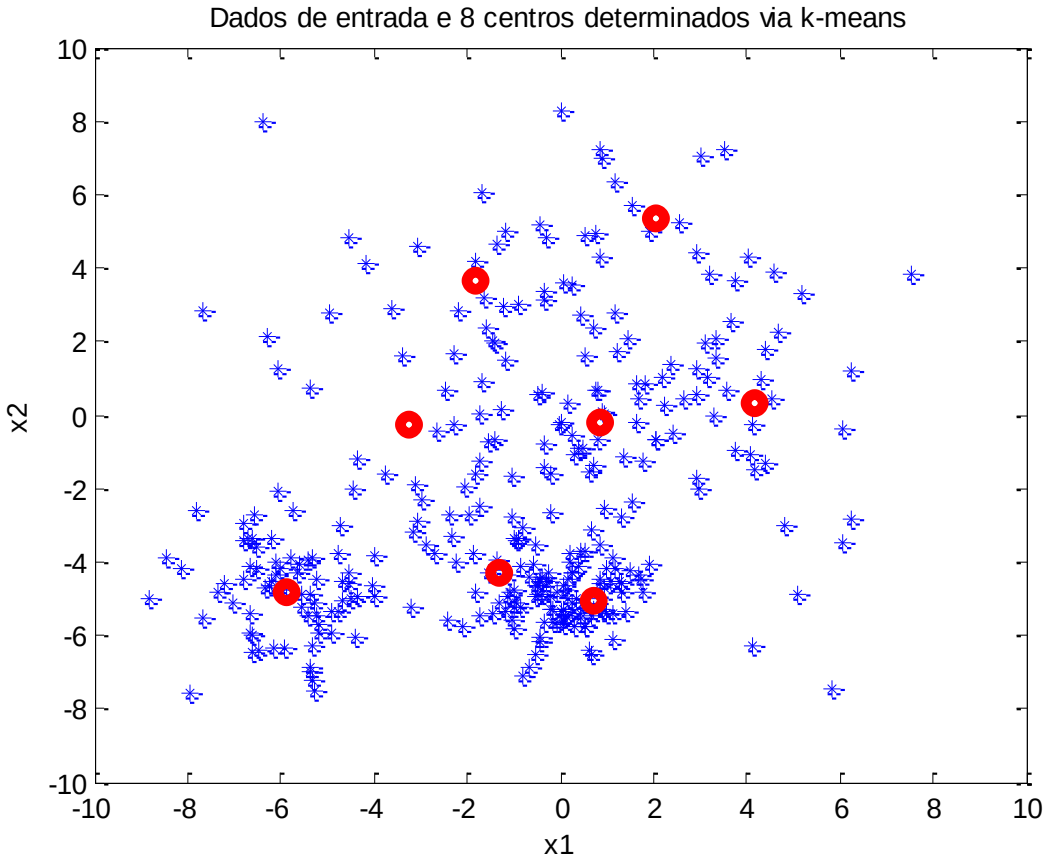


Figura 32 – Proposta de posicionamento dos centros das funções de base radial para uma rede neural RBF com 8 neurônios na camada intermediária.

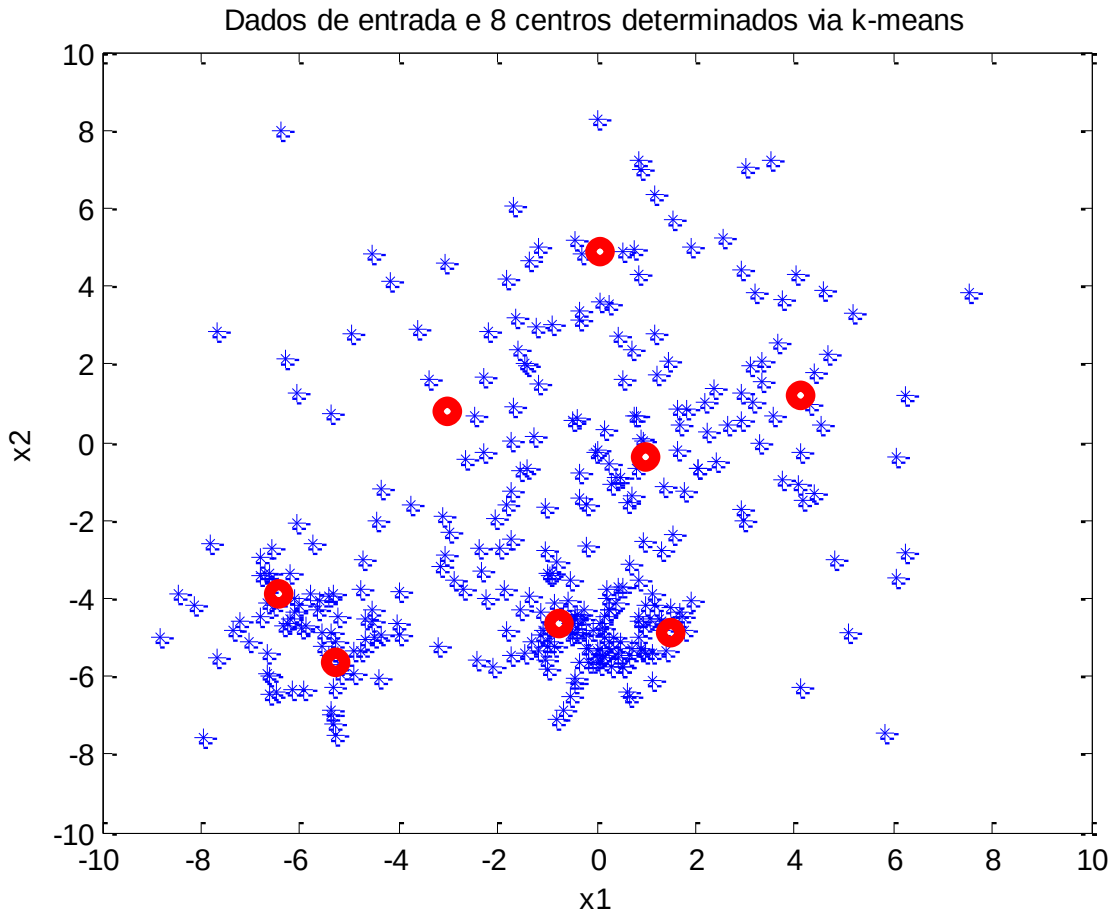


Figura 33 – Outra proposta de posicionamento dos centros para os mesmos dados, produzida por uma segunda execução do algoritmo *k*-means.

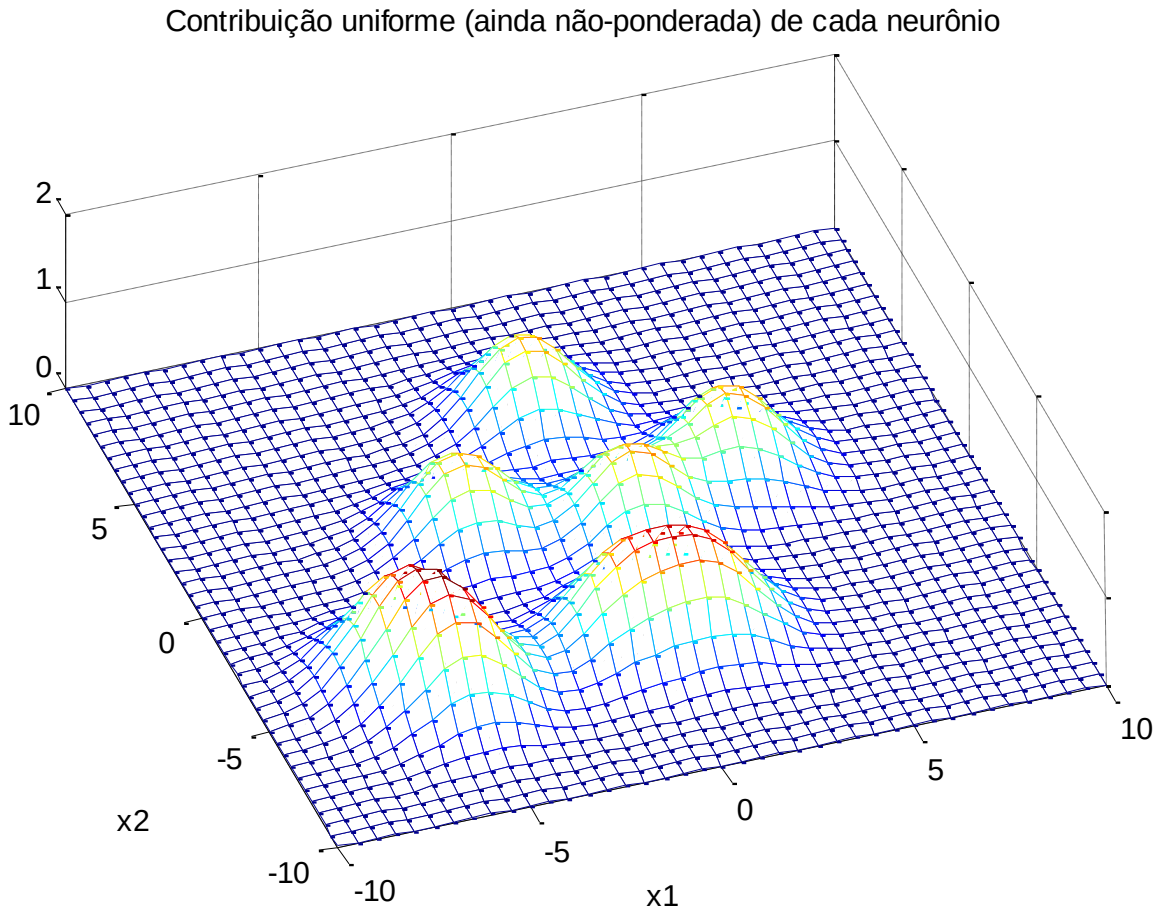


Figura 34 – Ativação dos neurônios da rede neural RBF com os centros da Figura 33, considerando todos os pesos de saída iguais a 1 e ausência de peso de bias. A dispersão é a mesma para todas as funções de ativação, dada pela fórmula da pág. 116.

21. Aprendizado supervisionado em RBF (revisão)

$$\{(\mathbf{x}_i, s_i)\}_{i=1}^{N}$$

$$f(\mathbf{w}, \mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j h_j(\mathbf{x}) = \sum_{j=0}^{n} w_j h_j(\mathbf{x}), \text{ com } h_0(\mathbf{x}) = 1.$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots & \mathbf{h}_n \end{bmatrix} = \begin{bmatrix} h_0(\mathbf{x}_1) & h_1(\mathbf{x}_1) & \cdots & h_n(\mathbf{x}_1) \\ h_0(\mathbf{x}_2) & h_1(\mathbf{x}_2) & \cdots & h_n(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_0(\mathbf{x}_N) & h_1(\mathbf{x}_N) & \cdots & h_n(\mathbf{x}_N) \end{bmatrix}$$

$$\text{Resolver } \mathbf{Hw} = \mathbf{s} \text{ com } \mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_n \end{bmatrix}^T \equiv \min_{\mathbf{w}} \|\mathbf{Hw} - \mathbf{s}\|_2^2$$

$$\mathbf{H}^T\mathbf{Hw} = \mathbf{H}^T\mathbf{s} \Rightarrow \mathbf{w} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{s}$$

21.1 Exemplo didático 2

- Dados *N* pontos no plano, na forma $\{x_i, y_i\}_{i=1}^{N}$, obtenha uma fórmula fechada para os coeficientes do polinômio $P_r(x) = a_r x^r + a_{r-1}x^{r-1} + \cdots + a_1 x + a_0$ que melhor se ajusta aos pontos, segundo o método dos quadrados mínimos e com $r \leq N-1$.

Solução:

Deve-se minimizar a seguinte função: $J(a_r, a_{r-1}, ..., a_1, a_0) = \sum_{i=1}^{N} (y_i - P_r(x_i))^2$

Esta função pode ser expandida na forma:

$$J(a_r, a_{r-1}, ..., a_1, a_0) = \sum_{i=1}^{N} {y_i}^2 - 2\sum_{i=1}^{N} P_r(x_i) y_i + \sum_{i=1}^{N} (P_r(x_i))^2 =$$

$$= \sum_{i=1}^{N} {y_i}^2 - 2\sum_{i=1}^{N} \left(\sum_{j=0}^{r} a_j {x_i}^j \right) y_i + \sum_{i=1}^{N} \left(\sum_{j=0}^{r} a_j {x_i}^j \right)^2 =$$

$$= \sum_{i=1}^{N} {y_i}^2 - 2\sum_{j=0}^{r} a_j \left(\sum_{i=1}^{N} y_i {x_i}^j \right) + \sum_{j=0}^{r} a_j \sum_{k=0}^{r} a_k \left(\sum_{i=1}^{N} {x_i}^{j+k} \right)$$

Condição necessária de otimalidade: $\frac{\partial J}{\partial a_j} = 0$ para j=0,1,...,r. Assim, para cada j resulta:

$$\frac{\partial J}{\partial a_j} = -2\sum_{i=1}^{N} y_i x_i^{\,j} + 2\sum_{k=0}^{r} a_k \sum_{i=1}^{N} x_i^{\,j+k} = 0 \text{ para } j=0,1,...,r.$$

$$\sum_{k=0}^{r} a_k \sum_{i=1}^{N} x_i^{\,j+k} = \sum_{i=1}^{N} y_i x_i^{\,j} \text{ para } j=0,1,...,r.$$

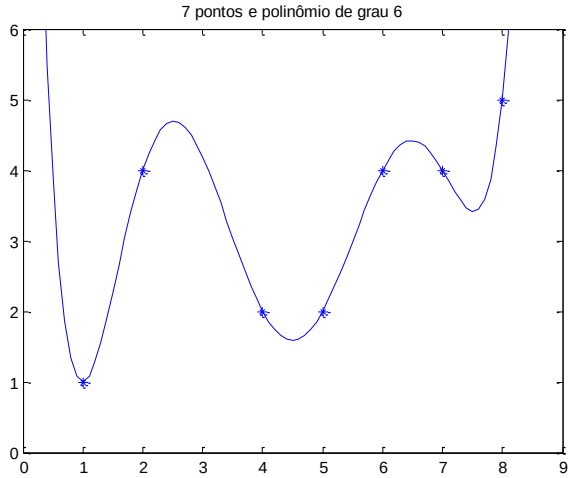
Expandindo para k e para j resulta:

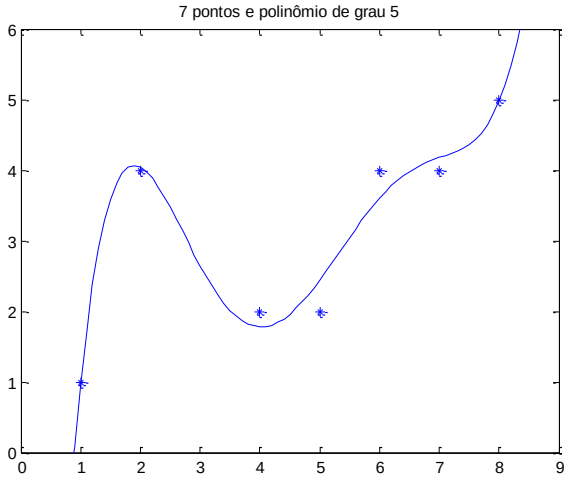
$$\begin{cases} a_0 \sum_{i=1}^{N} x_i^{\,0} + a_1 \sum_{i=1}^{N} x_i^{\,1} + a_2 \sum_{i=1}^{N} x_i^{\,2} + \cdots + a_r \sum_{i=1}^{N} x_i^{\,r} = \sum_{i=1}^{N} y_i x_i^{\,0} \\ a_0 \sum_{i=1}^{N} x_i^{\,1} + a_1 \sum_{i=1}^{N} x_i^{\,2} + a_2 \sum_{i=1}^{N} x_i^{\,3} + \cdots + a_r \sum_{i=1}^{N} x_i^{\,r+1} = \sum_{i=1}^{N} y_i x_i^{\,1} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_0 \sum_{i=1}^{N} x_i^{\,r} + a_1 \sum_{i=1}^{N} x_i^{\,r+1} + a_2 \sum_{i=1}^{N} x_i^{\,r+2} + \cdots + a_r \sum_{i=1}^{N} x_i^{\,2r} = \sum_{i=1}^{N} y_i x_i^{\,r} \end{cases}$$

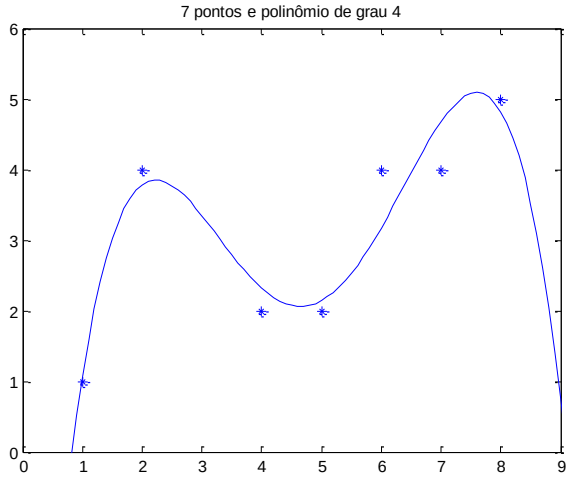
Em forma matricial, fica:

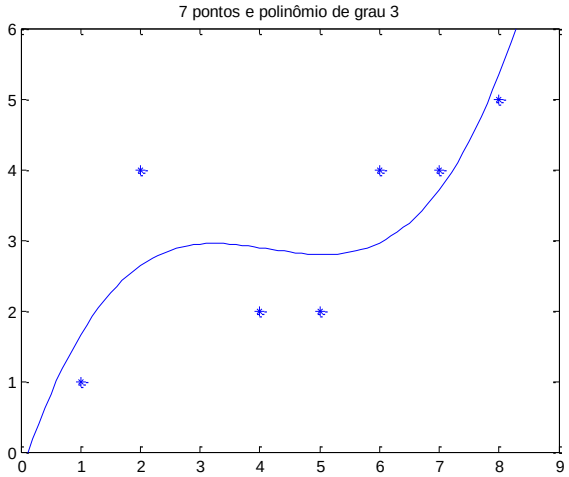
$$\begin{bmatrix} \sum_{i=1}^{N} x_i^{\,0} & \sum_{i=1}^{N} x_i^{\,1} & \sum_{i=1}^{N} x_i^{\,2} & \cdots & \sum_{i=1}^{N} x_i^{\,r} \\ \sum_{i=1}^{N} x_i^{\,1} & \sum_{i=1}^{N} x_i^{\,2} & \sum_{i=1}^{N} x_i^{\,3} & \cdots & \sum_{i=1}^{N} x_i^{\,r+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} x_i^{\,r} & \sum_{i=1}^{N} x_i^{\,r+1} & \sum_{i=1}^{N} x_i^{\,r+2} & \cdots & \sum_{i=1}^{N} x_i^{\,2r} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_r \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} y_i x_i^{\,0} \\ \sum_{i=1}^{N} y_i x_i^{\,1} \\ \vdots \\ \sum_{i=1}^{N} y_i x_i^{\,r} \end{bmatrix} \Rightarrow Za = q$$

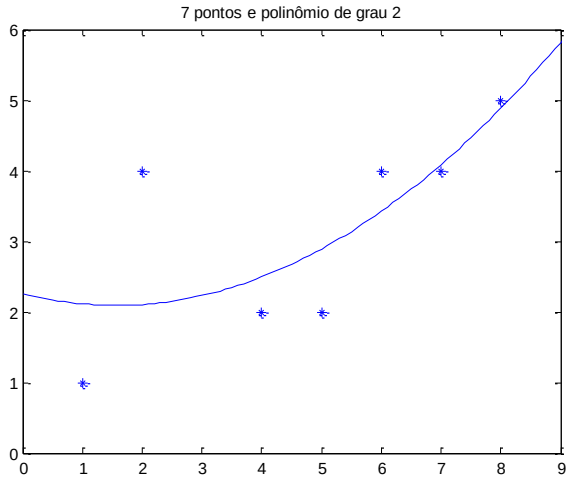
Se os x_i's forem todos distintos, então a matriz $Z \in \Re^{(r+1)\times(r+1)}$ é inversível e a solução assume a forma: $a = Z^{-1}q$. Exemplos:

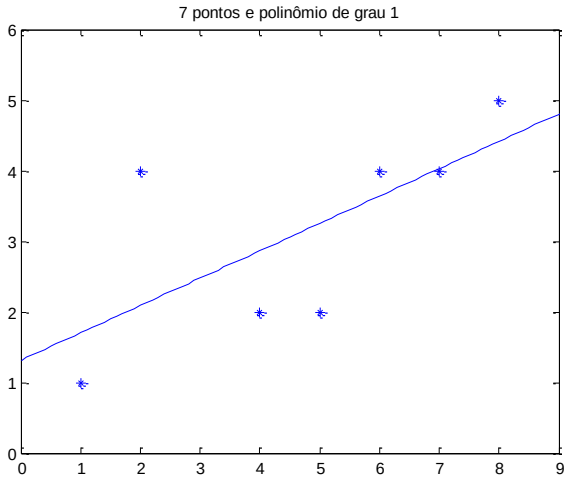





7 pontos e polinômio de grau 4


7 pontos e polinômio de grau 3


7 pontos e polinômio de grau 2


7 pontos e polinômio de grau 1

Programa em Matlab para se chegar aos resultados acima:

```
% Programa para aproximação polinomial de pontos no plano
clear all;
ptos = [1 1;2 4;4 2;5 2;6 4;7 4;8 5];
N = length(ptos(:,1));
x = ptos(:,1);
y = ptos(:,2);
r = 6; % Grau do polinômio
% Cálculo da matriz Z e do vetor q
Z = zeros(r+1,r+1);
q = zeros(r+1,1);
for j=0:r,
    for k=0:r,
        for i=1:N,
            Z(j+1,k+1) = Z(j+1,k+1)+power(x(i),j+k);
        end
    end
    for i=1:N,
        q(j+1) = q(j+1)+y(i)*power(x(i),j);
    end
end
% Obtenção dos coeficientes do polinômio
a = inv(Z)*q;
% Produção dos resultados gráficos
t = [0:0.1:10];
ind = 1;
f = zeros(length(t),1);
for i=1:length(t),
    for j=0:r,
        f(i,1) = f(i,1)+a(j+1)*power(t(i),j);
    end
end
figure(1);
% Plotagem dos pontos fornecidos
for i=1:N,
    plot(x(i),y(i),'*');
    hold on;
end
% Plotagem da função polinomial
plot(t,f);
axis([0 9 0 6]);
title(sprintf('%d pontos e polinômio de grau %d',N,r));
hold off;
```

22. Extreme Learning Machines (ELM)

- Correspondem a abordagens que definem arbitrariamente os pesos dos neurônios da camada intermediária de redes MLP, os quais não são passíveis de ajuste.
- Os pesos ajustáveis correspondem somente àqueles da camada de saída. Aplica-se, então, o método dos quadrados mínimos regularizados, que permitem obter soluções que generalizam bem. Já há extensões para outros tipos de funções de ativação.

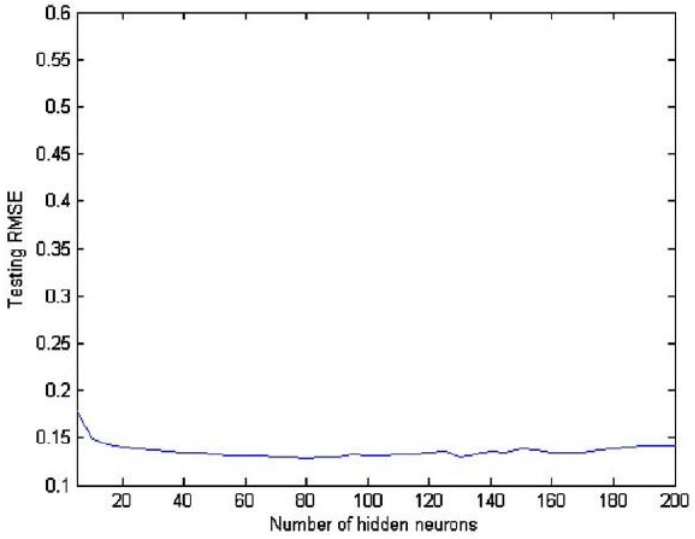


Fig. 3. The generalization performance of ELM is stable on a wide range of number of hidden nodes.

HUANG, G.B., ZHOU, H., DING, X. & ZHANG, R. (2012) "Extreme Learning Machine for Regression and Multiclass Classification", IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, vol. 42, no. 2, pp. 513-529.

23. Treinamento das ELMs

- Treinar uma ELM é equivalente a resolver o seguinte problema de otimização para cada uma das saídas da rede neural:

$$\mathbf{w}_k^* = \arg \min_{\mathbf{w}_k \in \Re^{n+1}} J(\mathbf{w}_k) + C_k \times \|\mathbf{w}_k\|_2^2$$

onde

- ✓ k é o índice da saída;
- ✓ n é o número de neurônios na camada intermediária;
- ✓ $\|\cdot\|_2$ é a norma euclidiana;
- ✓ C_k é um coeficiente ou fator de ponderação, a ser determinado, por exemplo, por métodos de busca unidimensional;
- ✓ $J(\mathbf{w}_k) = \frac{1}{2}\sum_{l=1}^{N}\left[\sum_{j=1}^{n} w_{kj} f(\mathbf{v}_j, b_j, \mathbf{x}_l) + w_{k0} - s_{kl}\right]^2$ – erro quadrático;
- ✓ N é o número de amostras disponíveis para treinamento.

- Este problema de otimização é conhecido na literatura como *ridge regression* (HASTIE, TIBSHIRANI & FRIEDMAN, 2009; HOERL & KENNARD, 1970).

23.1 Como encontrar os pesos sinápticos

- Uma vez fornecido o coeficiente de ponderação C_k, para a k-ésima saída da rede neural, o vetor de pesos sinápticos é obtido como segue:

1. Monta-se a matriz H_{inicial} de dimensão $N \times n$, com as ativações de todos os neurônios para todos os padrões de entrada, produzindo:

$$H_{\text{inicial}} = \begin{bmatrix} f(\mathbf{v}_1, b_1, \mathbf{x}_1) & f(\mathbf{v}_2, b_2, \mathbf{x}_1) & \cdots & f(\mathbf{v}_n, b_n, \mathbf{x}_1) \\ f(\mathbf{v}_1, b_1, \mathbf{x}_2) & \ddots & & \vdots \\ \vdots & & & \\ f(\mathbf{v}_1, b_1, \mathbf{x}_N) & \cdots & & f(\mathbf{v}_n, b_n, \mathbf{x}_N) \end{bmatrix}$$

2. Acrescenta-se uma coluna de 1's à matriz H_{inicial}, produzindo a matriz H:

$$H = \begin{bmatrix} f(\mathbf{v}_1, b_1, \mathbf{x}_1) & f(\mathbf{v}_2, b_2, \mathbf{x}_1) & \cdots & f(\mathbf{v}_n, b_n, \mathbf{x}_1) & 1 \\ f(\mathbf{v}_1, b_1, \mathbf{x}_2) & \ddots & & \vdots & 1 \\ \vdots & & & & \vdots \\ f(\mathbf{v}_1, b_1, \mathbf{x}_N) & \cdots & & f(\mathbf{v}_n, b_n, \mathbf{x}_N) & 1 \end{bmatrix}$$

3. Monta-se o vetor $\mathbf{s}_k$, contendo todos os padrões de saída, na forma:

$$\mathbf{s}_k = \begin{bmatrix} s_{k1} & s_{k2} & \cdots & s_{kN} \end{bmatrix}^T$$

4. Considerando que a matriz H tenha posto completo, o vetor w_k é obtido a partir da solução de quadrados mínimos:

   4.1. Se $n \leq N$, $\mathbf{w}_k = \left(H^T H + C_k I\right)^{-1} H^T \mathbf{s}_k$;

   4.2. Se $n > N$, $\mathbf{w}_k = H^T \left(HH^T + C_k I\right)^{-1} \mathbf{s}_k$.

23.2 Como encontrar o coeficiente de ponderação

- A maximização da capacidade de generalização requer a definição de um valor adequado para o coeficiente de ponderação C_k, associado à saída k.
- Sugere-se aqui o uso de uma busca unidimensional (por exemplo, via seção áurea), empregando um conjunto de validação. O valor "ótimo" de C_k é aquele que minimiza o erro junto ao conjunto de validação.

24. Regularização para funções unidimensionais

- Há outras formas de impor suavidade quando aproximando funções a partir de dados amostrados, além de *ridge regression*.

24.1 LASSO

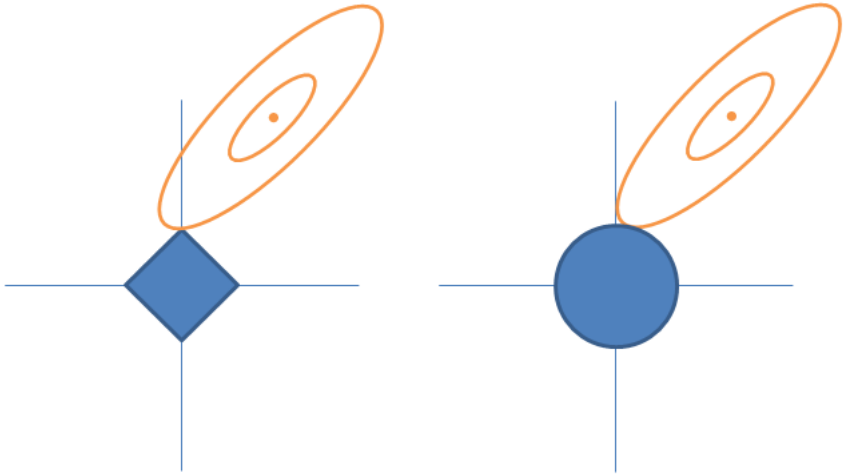
- Uma formulação alternativa considera a penalização com a norma l_1, produzindo:

$$\min_{\theta} \; J(\theta) + c\|\theta\|_1$$

  onde $\|\theta\|_1 = \sum_{i=1}^{P}|\theta_i|$ e $c \geq 0$.

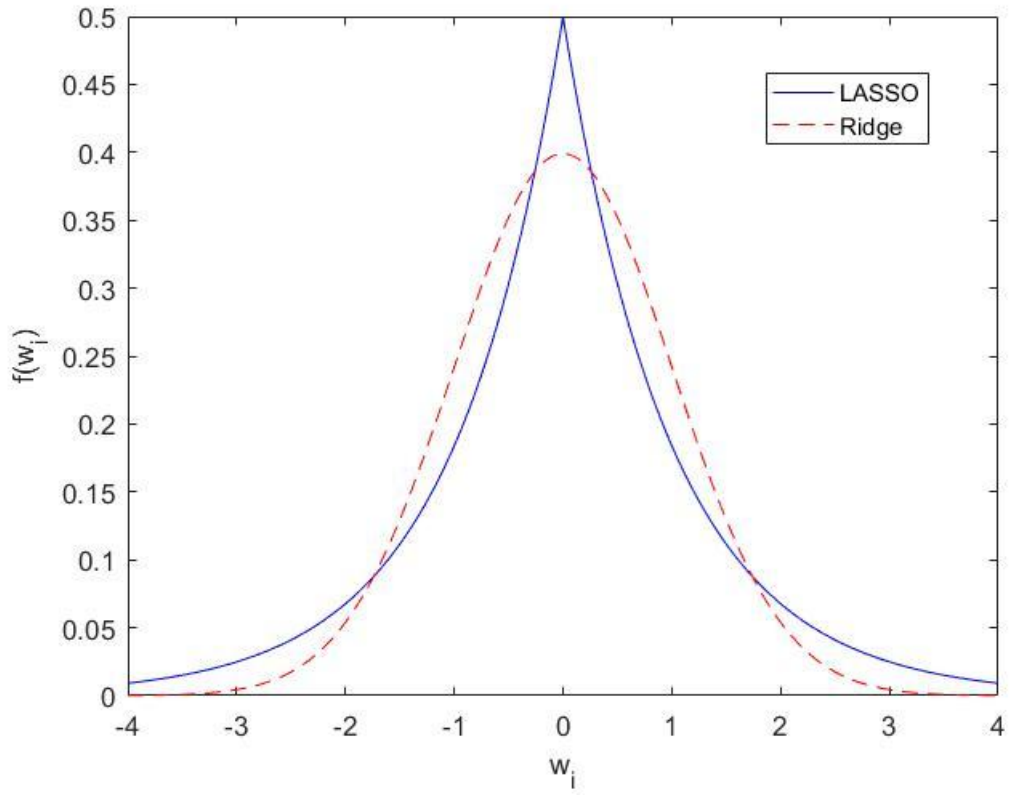
- Esta estratégia de regularização é conhecida como LASSO, do inglês *Least Absolute Shrinkage and Selection Operator*, e foi proposta por Tibshirani em 1996: *Tibshirani, R. "Regression shrinkage and selection via the LASSO", Journal of the Royal Statistical Society B, vol. 58, pp. 267–288, 1996*.

- A vantagem do LASSO está associada ao fato de que múltiplos termos do vetor θ acabam sendo anulados, o que indica a emergência de um processo de seleção de variáveis e leva a modelos de aproximação mais parcimoniosos.
- A motivação geométrica que permite justificar a anulação de um subconjunto de parâmetros ajustáveis é apresentada na figura a seguir. Nesta figura, apresentam-se os lugares geométricos de valores constantes para $J(\theta)$ e $\|\theta\|_1$, à esquerda, e os lugares geométricos de valores constantes para $J(\theta)$ e $\|\theta\|_2$, à direita.



Inspirado na interpretação geométrica presente em HASTIE, TIBSHIRANI & FRIEDMAN (2009).

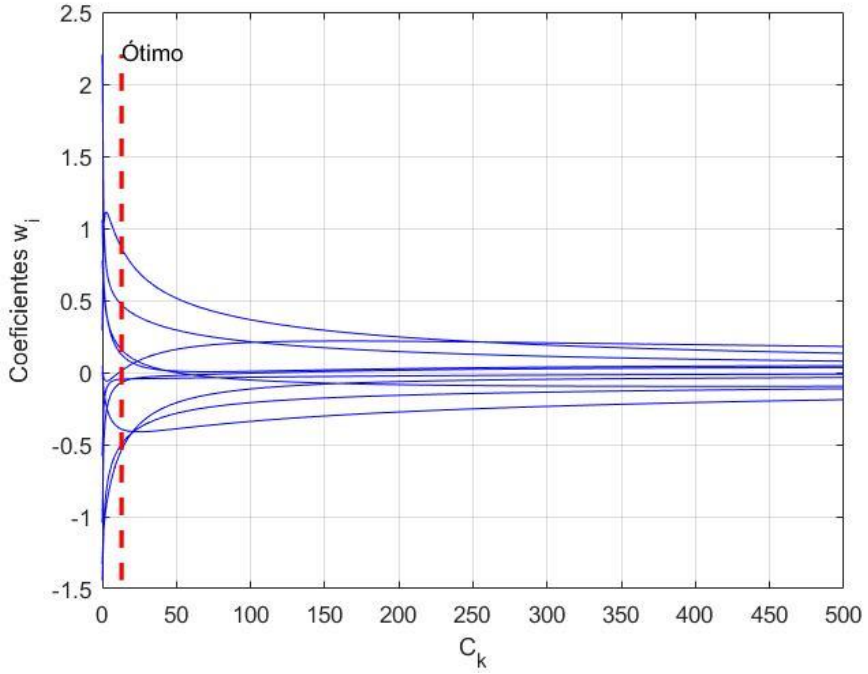
- Do ponto de vista estatístico, a penalização proporcional à norma l_1 do vetor de coeficientes define uma função densidade de probabilidade (PDF) a priori para os coeficientes distinta daquela associada à *ridge regression*:



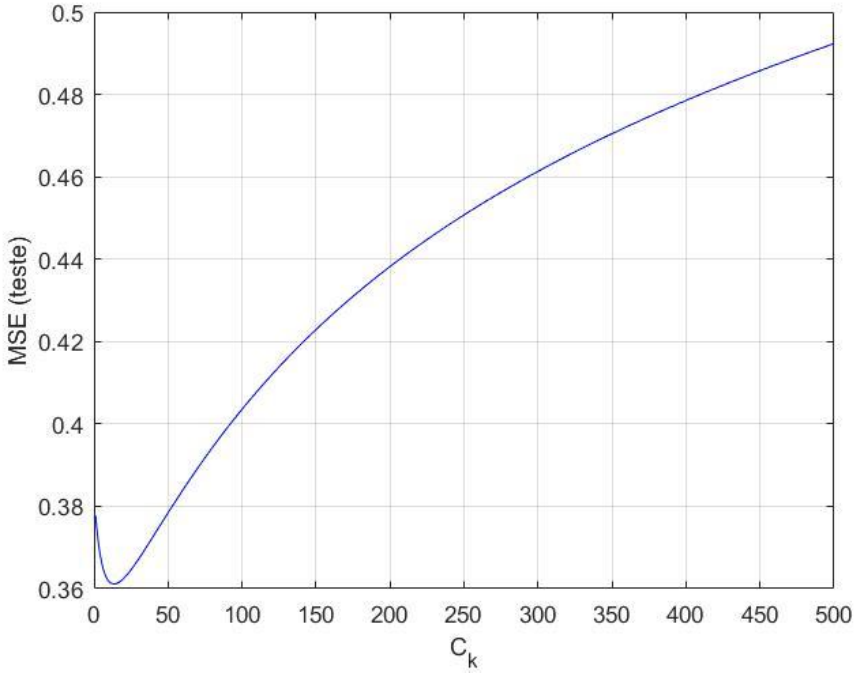
24.2 Exemplo ilustrativo

- ELM: 10 neurônios na camada intermediária.

Ridge regression

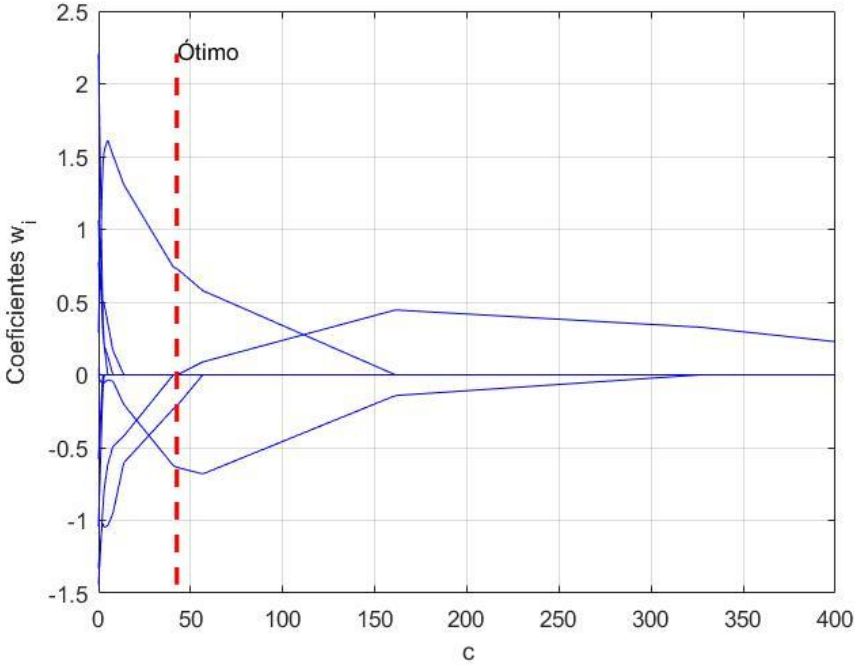


Curvas de evolução dos pesos da camada de saída da ELM em função do coeficiente de ponderação.

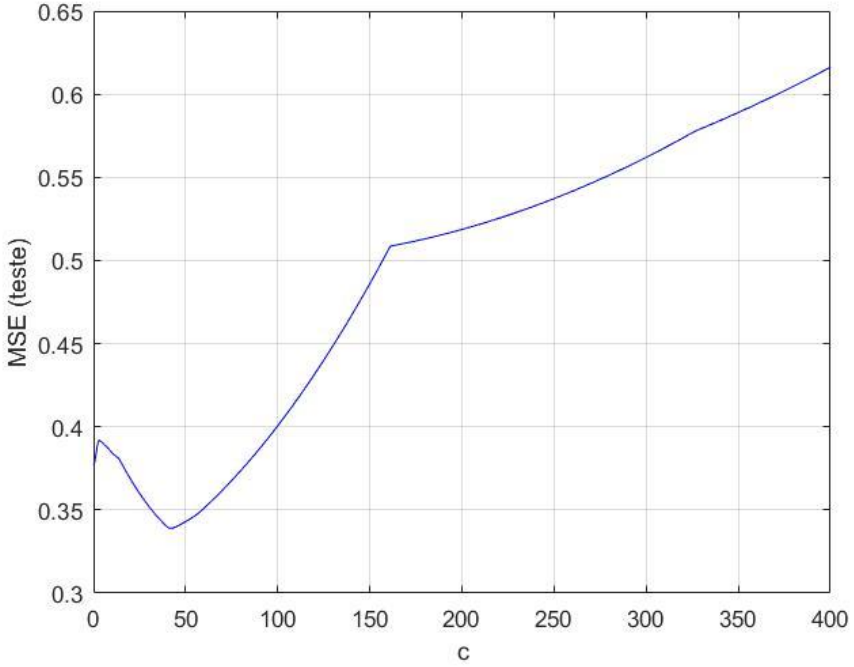


Erro quadrático médio junto ao conjunto de teste.

LASSO



Curvas de evolução dos pesos da camada de saída da ELM em função do coeficiente de ponderação.



Erro quadrático médio junto ao conjunto de teste.

- **Comparação dos resultados:**

	MSE Irrestrito	***Ridge Regression***	**LASSO**
w_1	0,0067	-0,3935	-0,6354
w_2	2,2067	0,4691	0
w_3	-1,3302	-0,4967	-0,2130
w_4	-1,4410	-0,5372	0
w_5	0,0440	0,0124	0
w_6	-0,5787	-0,0471	0
w_7	-1,0418	-0,0685	0
w_8	0,7768	0,1576	0
w_9	1,0620	0,1226	0
w_{10}	0,2905	0,8625	0,7293
Número de coefs. nulos	0	0	7
Coeficiente de regularização	0	6,508	42,553
MSE (Teste)	0,37706	0,36115	0,33896

24.3 Elastic Net

- Uma solução intermediária entre *ridge regression* e LASSO é a Elastic Net, proposta em: *Zou, H., Hastie, T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B, vol. 67, pp. 301–320, 2005*.
- No caso da Elastic Net, o problema de otimização regularizado assume a forma:

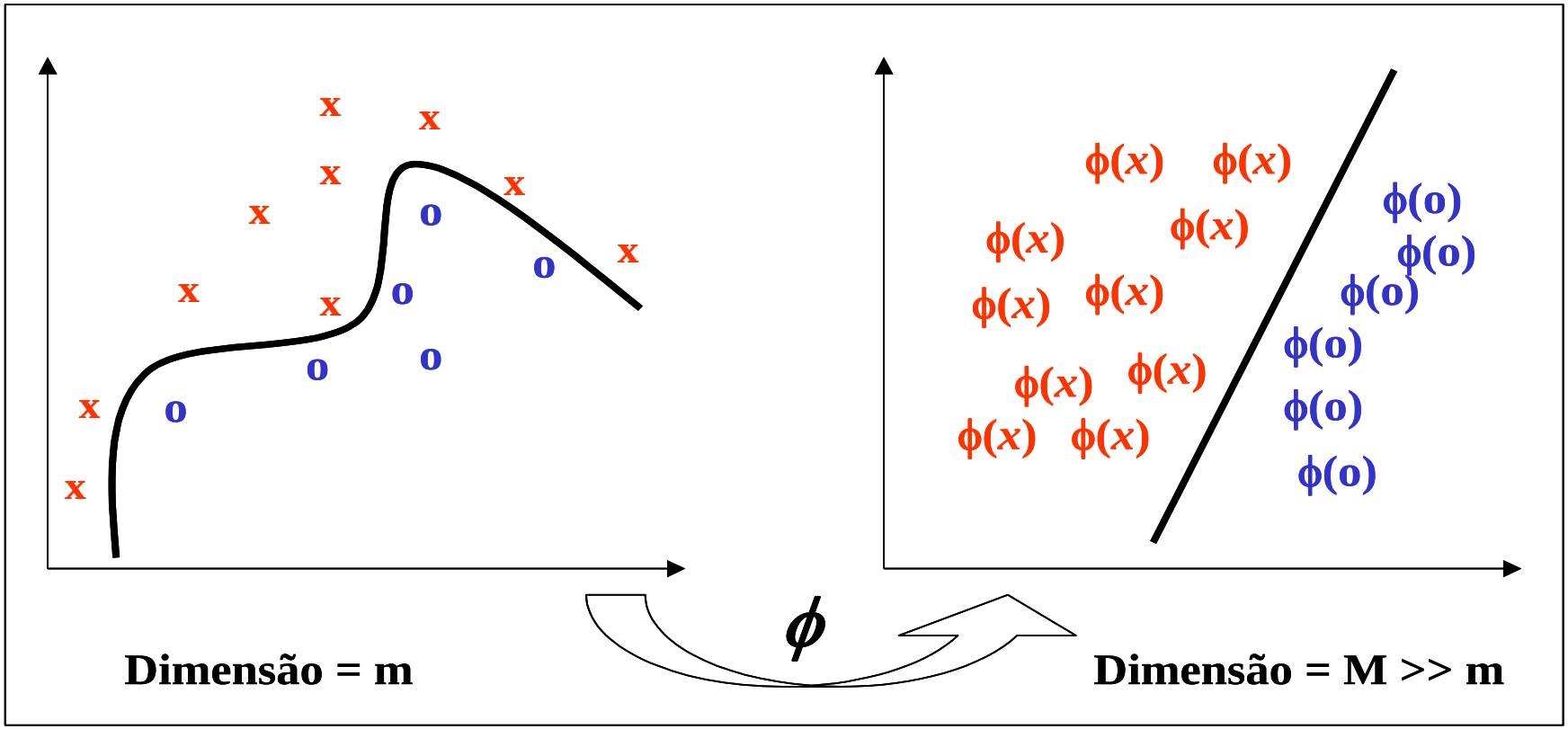
$$\min_{\theta} \; J(\theta) + c[\alpha\|\theta\|_1 + (1-\alpha)\|\theta\|_2^2]$$

  onde $\alpha \in [0, +1]$.
- O cálculo dos coeficientes c e α requer uma busca visando minimizar o erro junto a um conjunto de dados de validação.
- Repare que a Elastic Net contempla todas as demais propostas, para valores específicos de c e α:

  - ✓ Quadrados mínimos irrestrito: $c = 0$;
  - ✓ *Ridge Regression*: $c > 0$ e $\alpha = 0$;
  - ✓ LASSO: $c > 0$ e $\alpha = 1$;

Elastic net: $c > 0$ e $0 < \alpha < 1$.

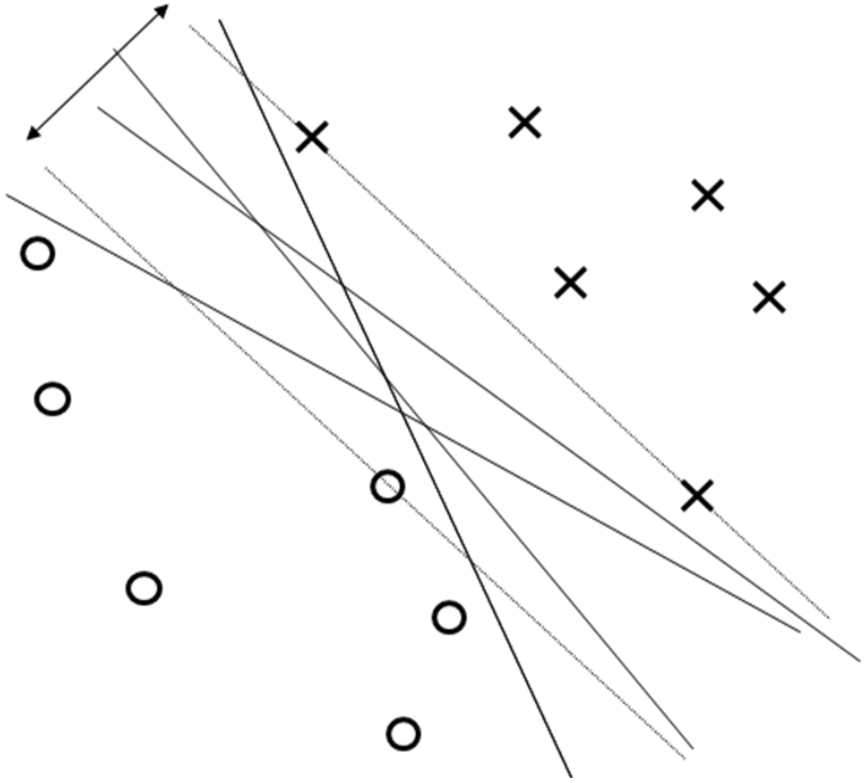
25. Máquinas de Vetores-Suporte (Support Vector Machines – SVM)

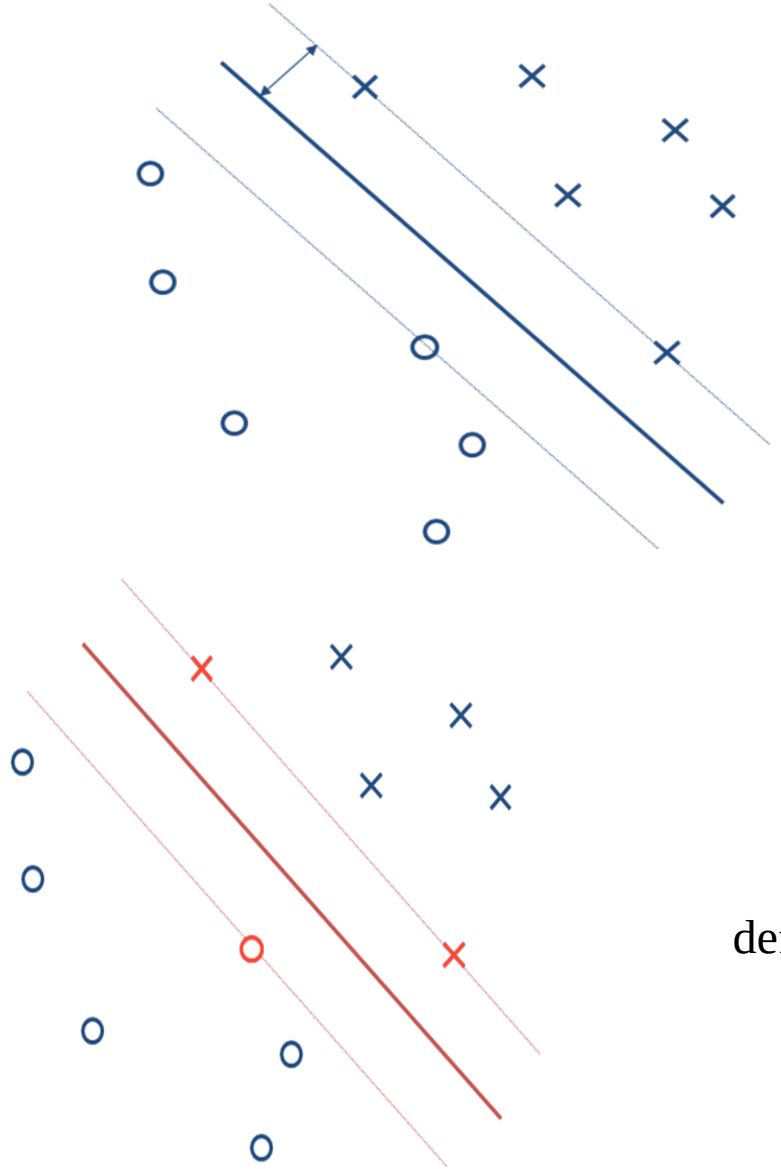


- Supera dois problemas:
  1. Como controlar a complexidade do espaço de hipóteses?
  2. Como evitar o problema de mínimos locais no ajuste de parâmetros?

25.1 Hiperplano de máxima margem de separação

- Há infinitos hiperplanos que separam os dados no espaço de características (espaço expandido). Qual deles escolher?





$$y = \text{sign}(\mathbf{w} \cdot \mathbf{X} + b)$$

$$\min_{\mathbf{w}} : \|\mathbf{w}\|$$

Os vetores-suporte são os pontos que definem o hiperplano de máxima margem.

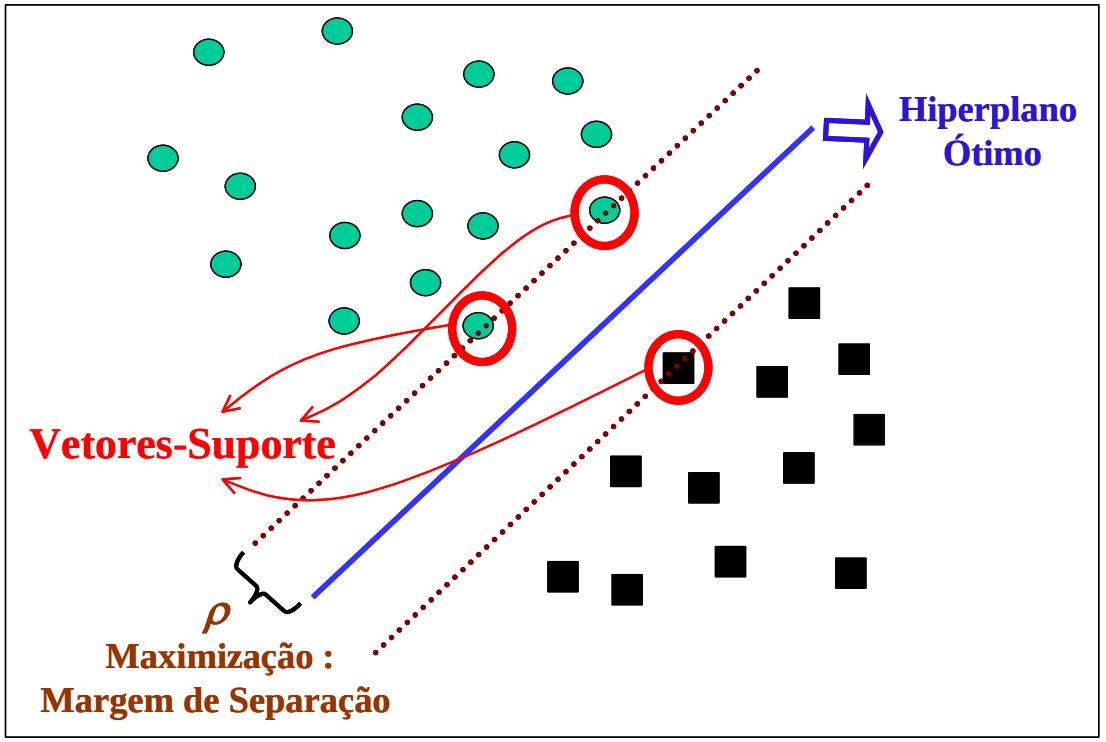
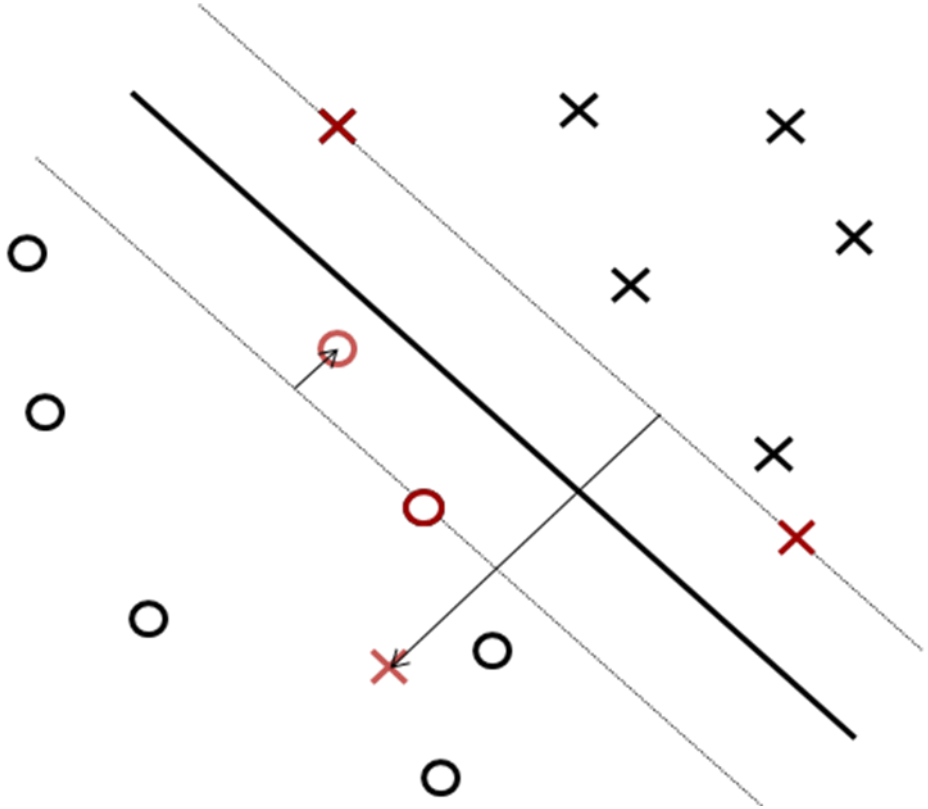


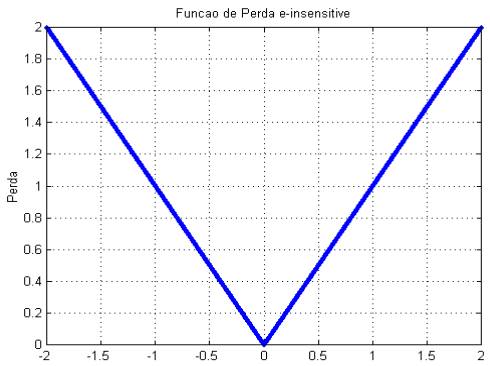
Ilustração do hiperplano de máxima margem

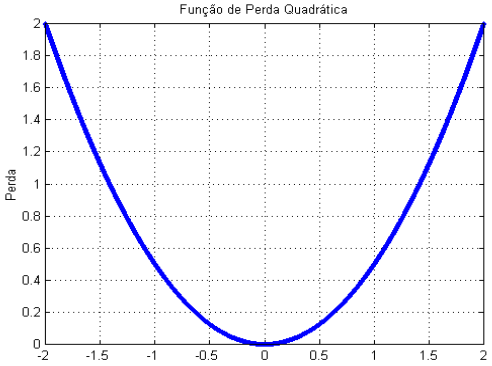
- Para um dado vetor de pesos **w** e intercepto *b*, a separação entre o hiperplano $g(\mathbf{x}) = (\mathbf{w}^T\mathbf{x}) + b = 0$ e o dado de entrada mais próximo é chamada de *margem de separação* e é denotada por ρ.
- Sempre que for possível obter um $\rho > 0$, existirão infinitos hiperplanos, dentre os quais se busca um hiperplano particular em que a *margem de separação* ρ *é maximizada*. De acordo com esta condição, a superfície de decisão é dita ser *o hiperplano ótimo* e o método de aprendizado de máquina é denominado máquina de vetores-suporte (SVM, do inglês *support vector machines*), sendo que os dados de treinamento que se encontram à distância ρ do hiperplano são chamados vetores-suporte (*support vectors*).

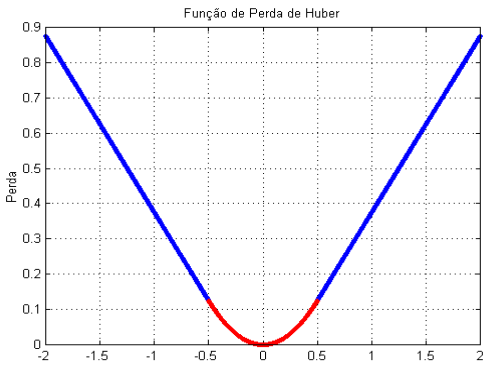


Problema primal:

$$\Phi(w,\xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} F(\xi_i) \qquad \text{s.a.} \qquad y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1-\xi_i, \qquad i = 1,\ldots,N$$







Problema dual:

$$\min_{\alpha} W(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i$$

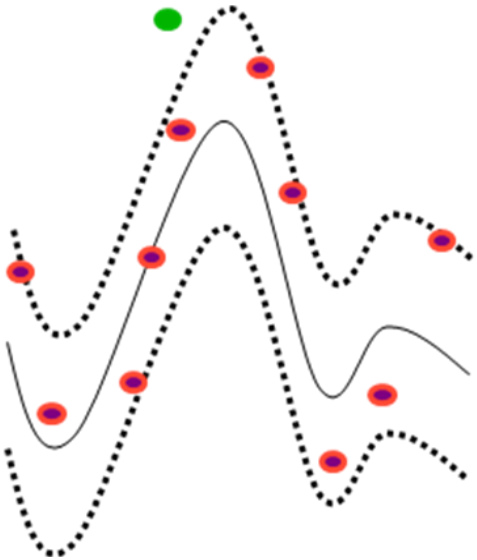
s.a. $\quad 0 \le \alpha_i \le C, \qquad i = 1,\ldots,N$

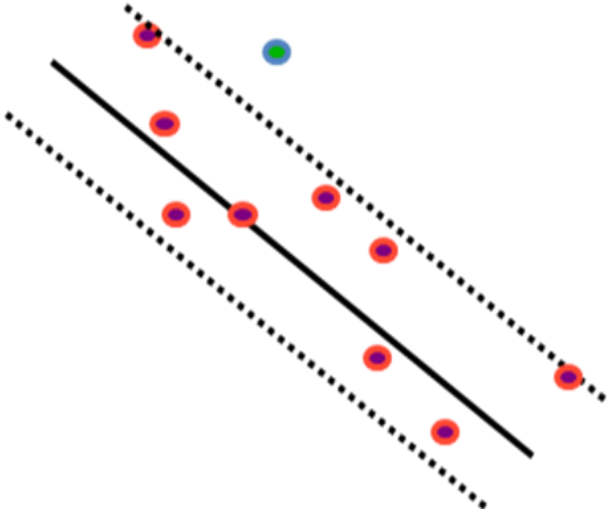
$$\sum_{i=1}^{N}\alpha_i y_i = 0$$

- **Truque do kernel** (*kernel trick*): substitui-se o cálculo do produto interno entre os vetores (dados) em um espaço de elevada dimensão M (que será chamado de espaço de características, do inglês *feature space*) pela aplicação da função *kernel* $K(\mathbf{x}_i, \mathbf{x}_j)$ sobre os dados no espaço original.
- Desta forma, a função $\phi(\cdot): \Re^m \rightarrow \Re^M$ que realizaria o mapeamento do espaço original para o espaço de características nem precisa ser conhecida (ela é implícita ao *kernel* escolhido).

25.2 Classificação × Regressão

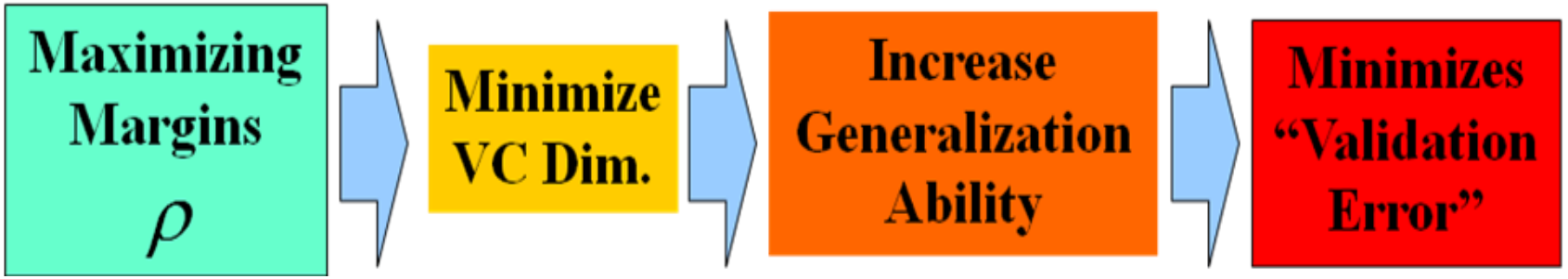
Classificação





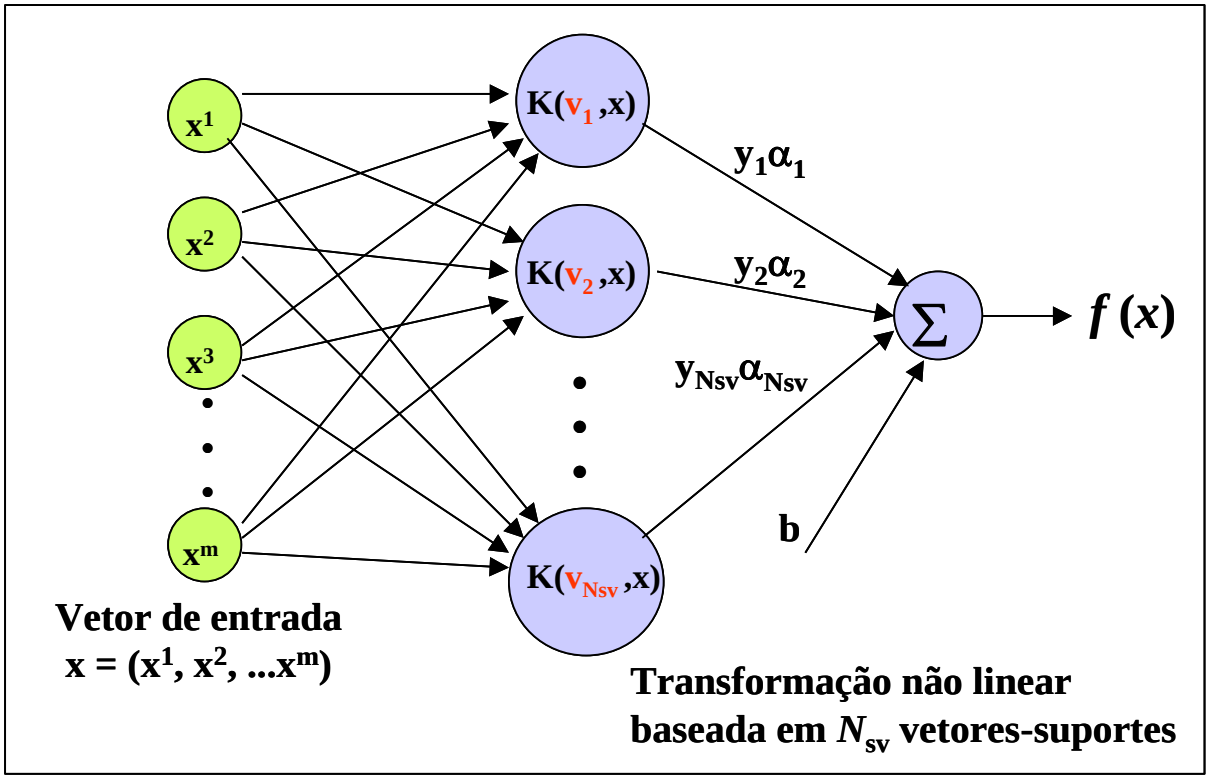
Regressão

25.3 Embasamento teórico



- A função de decisão dada pela SVM assume a forma:

$$f(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{N_{sv}} \alpha_i^{sv\,*} y_i^{sv} K\left(\mathbf{x}_i^{sv}, \mathbf{x}\right) + b^* \right).$$



Decisão de uma SVM analisada sob o ponto de vista de uma estrutura típica de rede neural.

25.4 Funções kernel mais empregadas

i. Linear

$$K(x, y) = x \cdot y$$

ii. Polynomial

$$K(x, y) = (x \cdot y + 1)^d$$

iii. Gaussian Radial Basis Function

$$K(x, y) = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$$

iv. Exponential Radial Basis Function

$$K(x, y) = \exp\left(-\frac{|x-y|}{2\sigma^2}\right)$$

v. Sigmoid

$$K(x, y) = \tanh(b(x \cdot y) + c)$$

vi. Fourier Series

$$K(x, y) = \frac{sin(N + \frac{1}{2})(x-y)}{sin\left(\frac{1}{2}(x-y)\right)}$$

vii. Linear Splines

$$K(x, y) = 1 + xy + xy\min(x, y) - \frac{(x+y)}{2}(\min(x, y))^2 + \frac{1}{3}(\min(x, y))^3$$

viii. Bn-splines

$$K(x, y) = B_{2n+1}(x-y)$$

26. Referências

BATTITI, R. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation*, vol. 4, no. 2, pp. 141-166, 1992.

BISHOP, C. Exact Calculation of the Hessian Matrix for the Multilayer Perceptron. *Neural Computation*, vol. 4, no. 4, pp. 494-501, 1992.

COVER, T.M. & HART, P.E. "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, 13:21-27, 1967.

CYBENKO, G. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303-314, 1989.

GOMES, L.C.T. & VON ZUBEN, F.J. A Neuro-Fuzzy Approach to the Capacitated Vehicle Routing Problem. *Proceedings of the IEEE International Joint Conference on Neural Networks* (IJCNN'2002), vol. 2, pp. 1930-1935, Honolulu, Hawaii, May 12-17, 2002.

HARTMAN, E.J., KEELER, J.D., KOWALSKI, J.M. Layered neural networks with gaussian hidden units as universal approximators. *Neural Computation*, vol. 2, no. 2, pp. 210-215, 1990.

HAYKIN, S. "Neural Networks: A Comprehensive Foundation", 2nd edition, Prentice-Hall, 1999.

HEBB, D. O. "The Organization of Behavior", Wiley, 1949.

HINTON, G.E. "Connectionist learning procedures", *Artificial Intelligence*, 40: 185-234, 1989.

HINTON, G. E. & SEJNOWSKI, T.J. "Learning and relearning in Boltzmann machines", *in* D. E. Rumelhart, J. L. McClelland & The PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, vol. 1, pp. 282-317, 1986.

HORNIK, K., STINCHCOMBE, M., WHITE, H. Multi-layer feedforward networks are universal approximators. *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

HORNIK, K., STINCHCOMBE, M., WHITE, H. Universal approximation of an unknown function and its derivatives using multilayer feedforward networks. *Neural Networks*, vol. 3, no. 5, pp. 551-560, 1990.

HORNIK, K., STINCHCOMBE, M., WHITE, H., AUER, P. Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives. *Neural Computation*, vol. 6, no. 6, pp. 1262-1275, 1994.

HUANG, G.-B., CHEN, L., SIEW, C.-K. Universal Approximation Using Incremental Constructive Feedfoward Networks with Random Hidden Nodes. *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

HUANG, G.-B., WANG, D.H., LAN, Y. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, vol. 2, pp. 107-122, 2011.

HUANG, G.-B., ZHOU, H., DING, X., ZHANG, R. Extreme Learning Machines for Regression and Multiclass Classification. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 42, no. 2, pp. 513-529, 2012.

HUANG, G.-B., ZHU, Q.-Y., SIEW, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. *Proceedings of the International Joint Conference on Neural Networks* (IJCNN'2004), vol. 2, pp. 985-990, 2004.

HUANG, G.-B., ZHU, Q.-Y., SIEW, C.-K. Extreme learning machine: theory and applications. Neurocomputing, vol. 70, pp. 489-501, 2006.

JAIN, A.K., MURTY, M.N. & FLYNN, P.J. "Data Clustering: A Review", ACM Computing Surveys, vol. 31, no. 3, pp. 264-323, 1999.

MACQUEEN, J. "Some methods for classification and analysis of multivariate observation", *in* L.M. LeCun and J Neyman (eds.) *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1:281-297, 1967.

MARR, D. "A theory for cerebral neocortex", *Proceedings of the Royal Society of London, Series B*, 176: 161-234, 1970.

NERRAND, O., ROUSSEL-RAGOT, P., PERSONNAZ, L., DREYFUS, G. Neural Networks and Nonlinear Adaptive Filtering: Unifying Concepts and New Algorithms. *Neural Computation*, vol. 5, no. 2, pp. 165-199, 1993.

PARK, J., SANDBERG, I.W. Universal approximation using radial basis function networks. *Neural Computation*, vol. 3, no. 2, pp. 246-257, 1991.

POTVIN, J.-I. & ROBILLARD, C. "Clustering for Vehicle Routing wih a Competitive Neural Network", *Neurocomputing*, 8, 125-139, 1995.

SMITH, K.A. "Neural Networks for Combinatorial Optimization: A Review of More than a Decade of Research", *INFORMS Journal on Computing*, 11, 1, 15-34, 1999.

"*Muitos não fizeram quando deviam porque não quiseram quando podiam*".

François Rabelais (1483/94-1553)