Redes Neurais Artificiais e Máquinas de Aprendizado (Parte 2)

Índice

1. Rede neural com função 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 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 definidos previamente 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 1(a));
- Uma função de base radial monotonicamente crescente típica é a função multiquádrica dada na forma:

- $h_j(x) = \dfrac{\sqrt{r_j^2 + \left(x - c_j\right)^2}}{r_j}$, para o caso escalar (veja Figura 1(b));

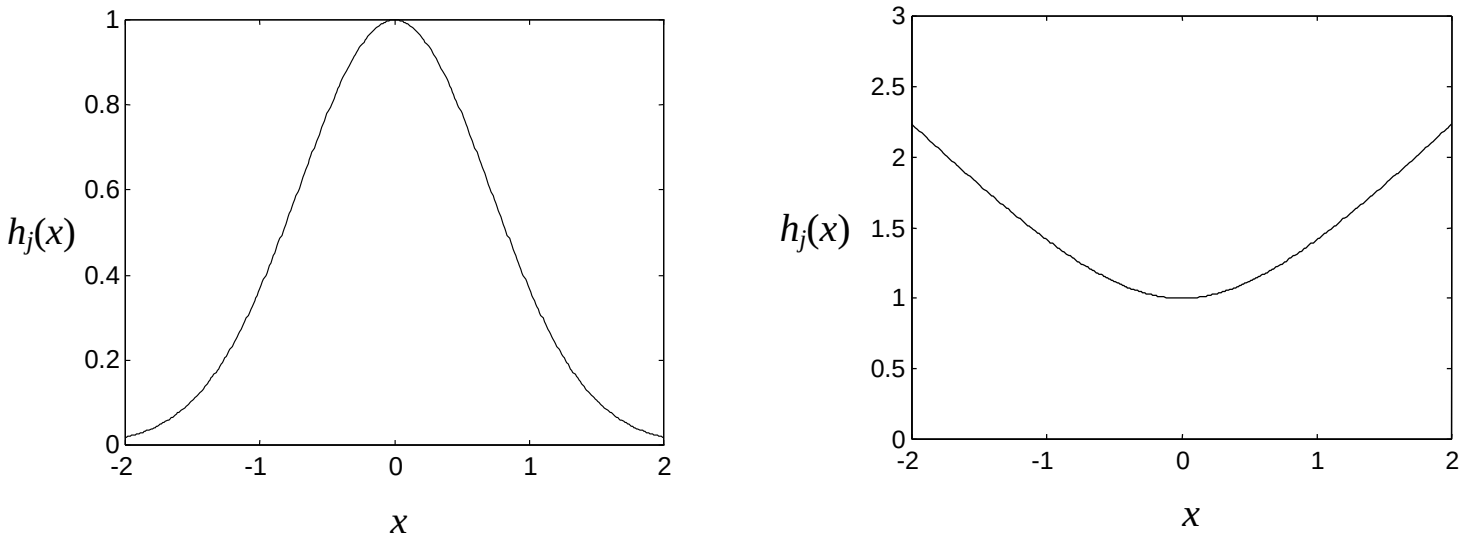


Figura 1 – Exemplos de funções de base radial monovariáveis, com $c_j = 0$ e $r_j = 1$

- 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:

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

- Neste caso, os elementos do vetor $\sigma_j = \begin{bmatrix} \sigma_{j1} & \sigma_{j2} & \cdots & \sigma_{jn} \end{bmatrix}^T$ 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 e o centro da função de base radial.

1.1 Formulação matemática

- 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 de múltiplas camadas.
- O fato de o 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.
- Como exposto acima, se apenas os pesos da camada de saída formarem o conjunto de parâmetros ajustáveis, então a rede neural é 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 é não-linear nos parâmetros, admitindo o próprio algoritmo de retro-propagação do erro para o processo de ajuste via treinamento supervisionado, como feito no caso

do perceptron multicamadas, embora aqui os mínimos locais tenham uma influência muito maior.

- A arquitetura da rede é apresentada na Figura 2, 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 3.
- A consequência imediata do uso de funções de ativação de base radial está na forma como as entradas são processadas pelos neurônios da camada intermediária. 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.

1.2 Capacidade de aproximação universal de redes RBF

- 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).

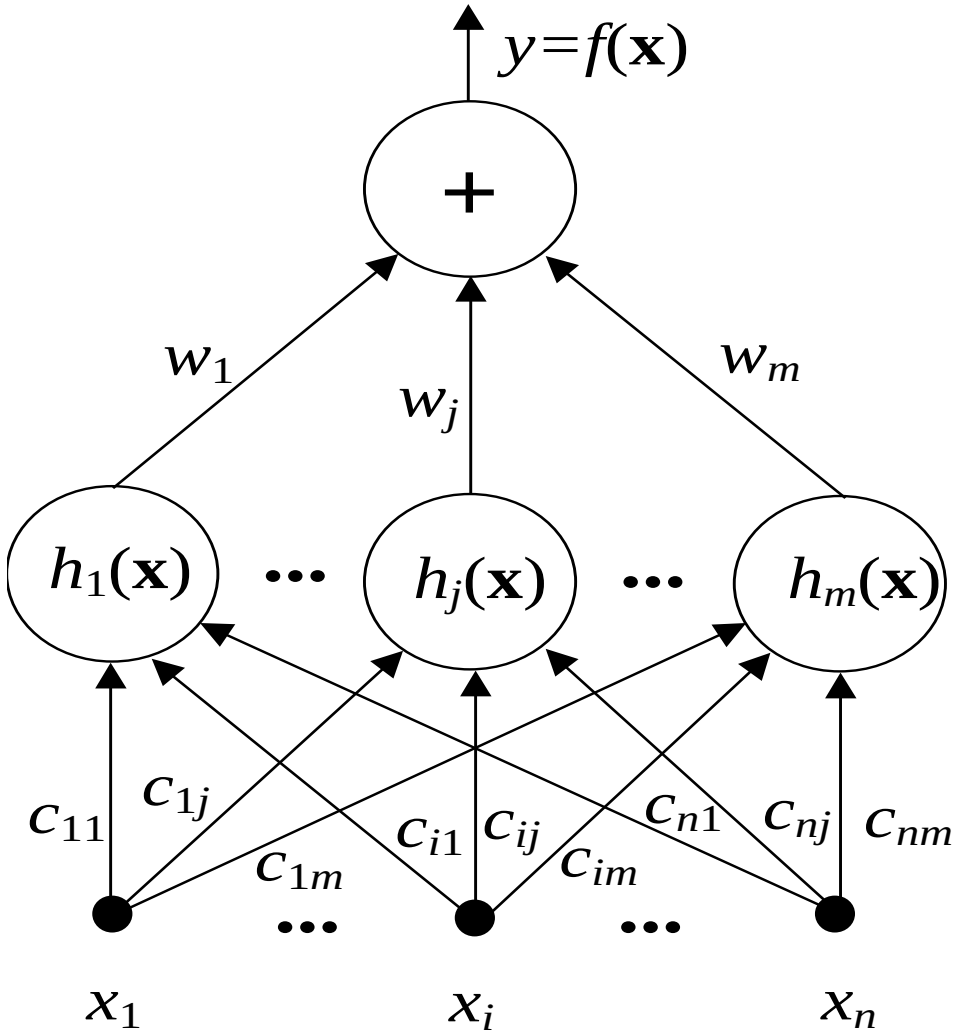


Figura 2 – Rede neural de base radial (BROOMHEAD & LOWE, 1988). Observação: Geralmente é também considerada uma entrada constante para o neurônio de saída.

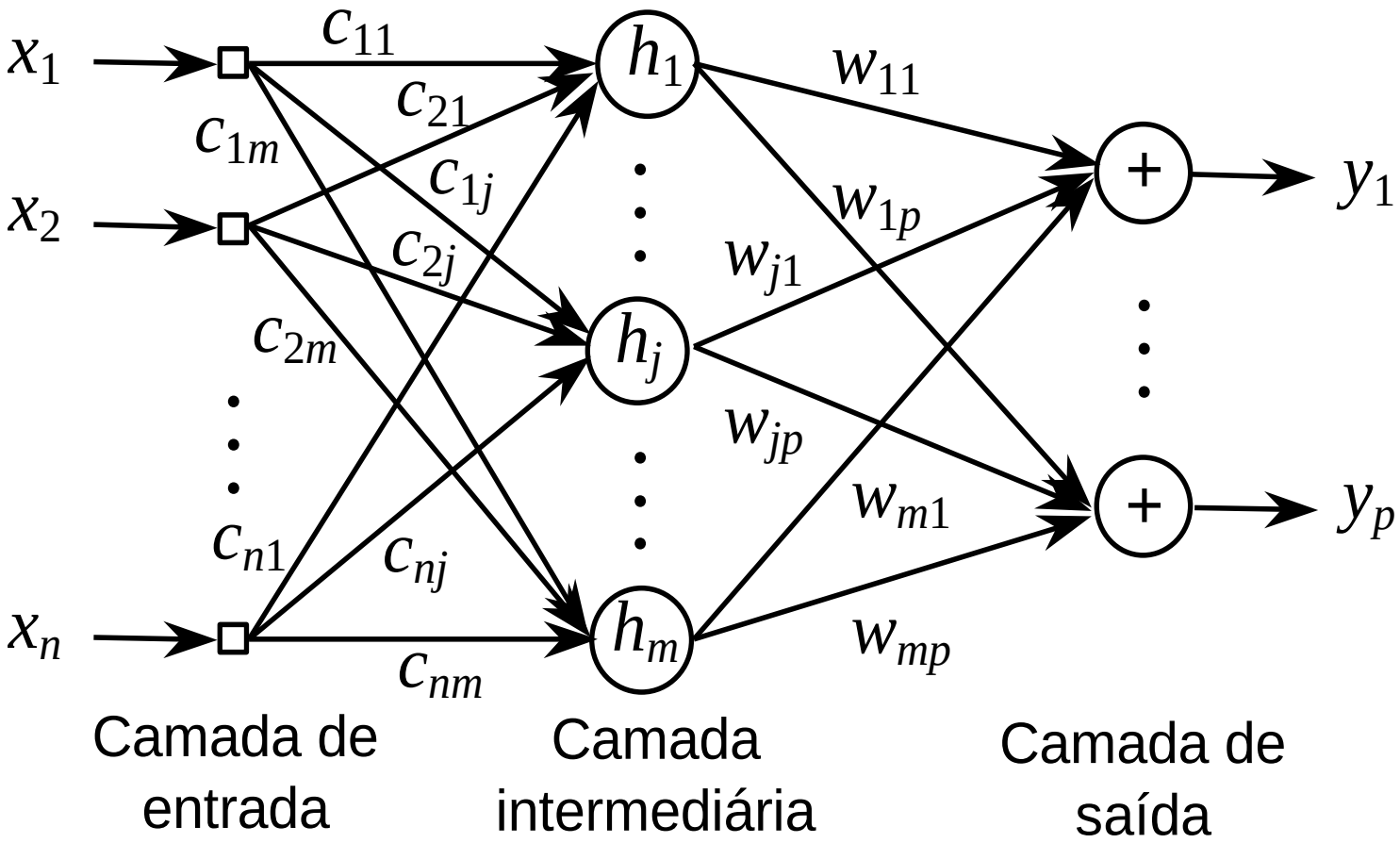


Figura 3 – Rede neural de base radial com múltiplas saídas. Observação: Geralmente é também considerada uma entrada constante para cada neurônio de saída.

1.3 Métodos de treinamento já propostos na literatura

- Várias abordagens para o treinamento de redes neurais com funções de base radial já foram propostas. Geralmente, elas podem ser divididas em duas partes:
  - Definição dos centros, forma e dispersão das funções de base radial, normalmente baseada em treinamento não-supervisionado (quantização vetorial ou algoritmo de treinamento competitivo) ou computação evolutiva;
  - Aprendizado dos pesos da camada de saída, responsáveis pela combinação linear das ativações da camada intermediária, empregando regressão linear.

1.4 O método dos quadrados mínimos para modelos lineares nos parâmetros

- Quando o treinamento supervisionado é aplicado a modelos lineares nos parâmetros, o método dos quadrados mínimos conduz a um problema de otimização que apresenta solução na forma fechada.
- 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.

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

1.5 Obtenção da solução do problema de regressão linear

- 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 $f(\cdot)$, 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 $\{w_r, r = 1,...,m\}$. Desde que os centros não sejam coincidentes, este sistema de equações possui 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 $m \leq N$.

1.6 Exemplo

- O modelo linear de regressão mais simples é a reta, aplicada nos casos em que a entrada é escalar: $f(x) = w_1 h_1(x) + w_2 h_2(x)$, onde $h_1(x) = 1$ e $h_2(x) = x$.
- Considere que foram amostrados, na presença de ruído, três pontos da curva $y = x$, gerando o conjunto de treinamento: $\{(x_i, s_i)\}_{i=1}^{3} = \{(1,1.1),(2,1.8),(3,3.1)\}$.
- Obviamente, não se conhece a equação da curva, mas apenas estes três pontos amostrados. Para estimar w_1 e w_2, vamos proceder de acordo com os passos do método dos quadrados mínimos.

$$H = \begin{bmatrix} h_1(x_1) & h_2(x_1) \\ h_1(x_2) & h_2(x_2) \\ h_1(x_3) & h_2(x_3) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \qquad \mathbf{s} = \begin{bmatrix} 1.1 \\ 1.8 \\ 3.1 \end{bmatrix} \qquad \mathbf{w} = \left(H^T H\right)^{-1} H^T \mathbf{s} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Para o mesmo conjunto de treinamento, considere agora que

$$f(x) = w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x),$$

onde $h_1(x) = 1$, $h_2(x) = x$ e $h_3(x) = x^2$. Enquanto no caso anterior tínhamos $m < N$, agora temos $m = N$.

- O efeito da adição da função-base extra $h_3(x)$ representa a adição de uma coluna $\mathbf{h}_3 = \begin{bmatrix} h_3(x_1) \\ h_3(x_2) \\ h_3(x_3) \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}$ junto à matriz H, e a solução assume a forma $\mathbf{w} = \begin{bmatrix} 1 \\ -0.2 \\ 0.3 \end{bmatrix}$.

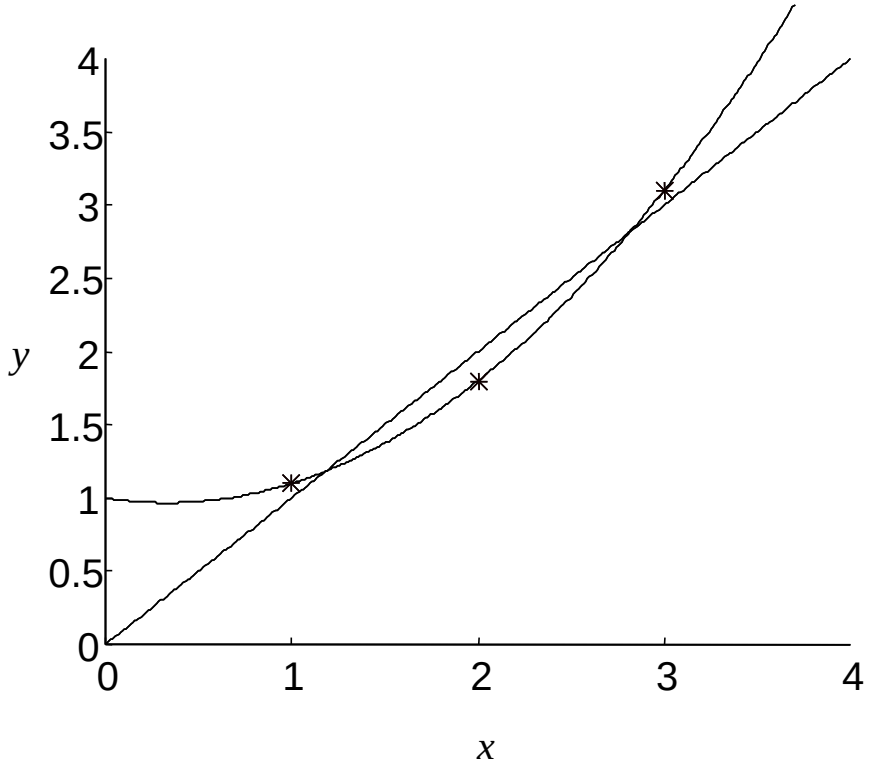
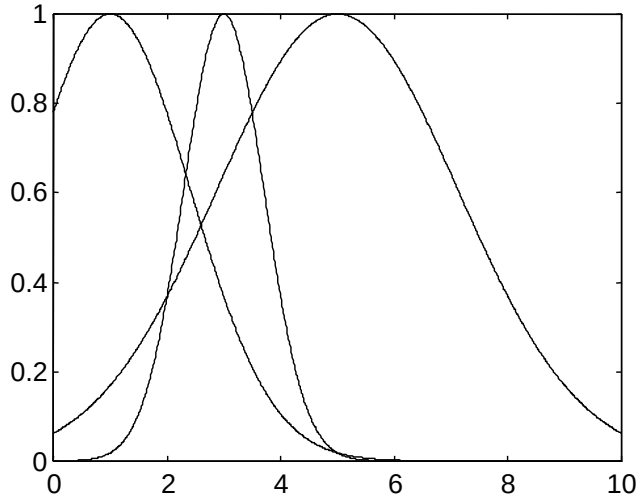


Figura 4 – Modelos de regressão linear (ordem 1 e ordem 2).

- Observe que ambos os modelos são lineares nos parâmetros (daí a denominação de regressão linear), embora para $m = 3$ tenhamos um modelo não-linear. É fundamental compreender o conceito de modelo não-linear, mas linear nos parâmetros.

1.7 Aproximação usando rede neural RBF

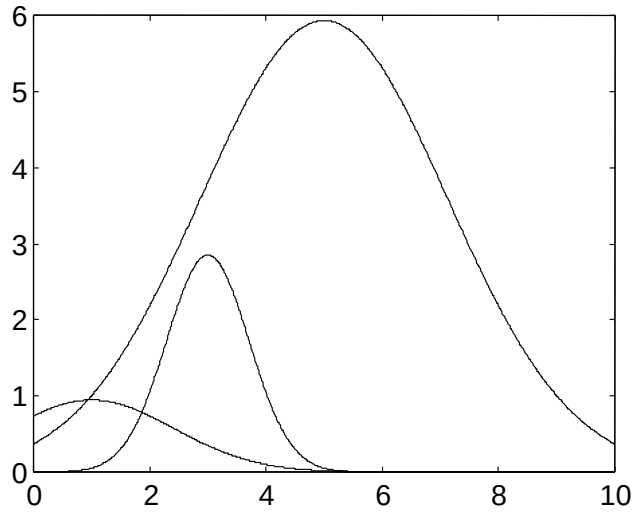


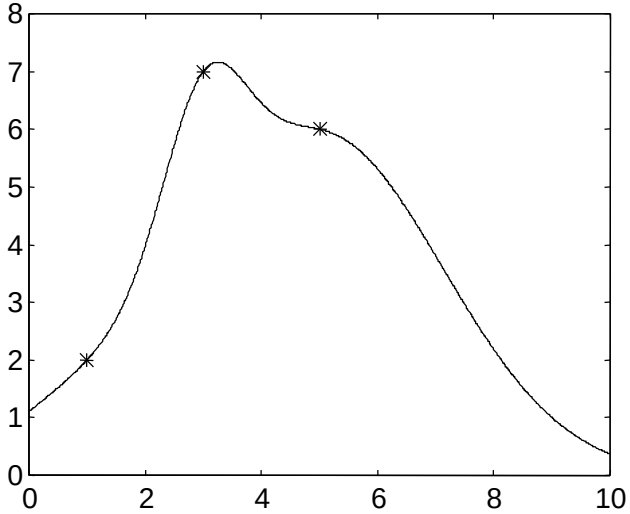
Caso 1: $m = N$

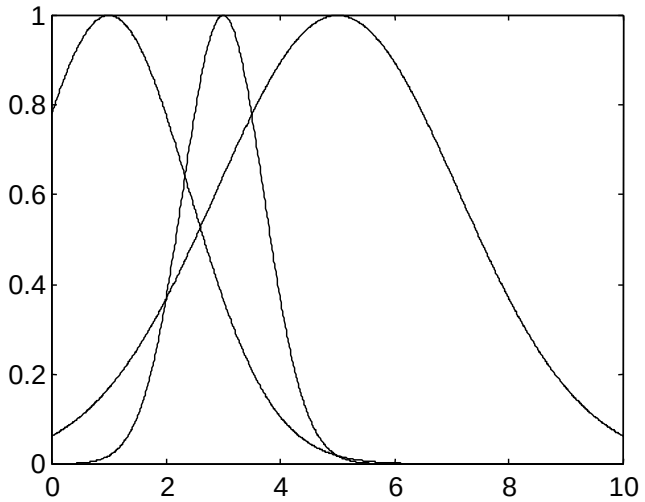
Pontos amostrados: (1,2); (3,7); (5,6)

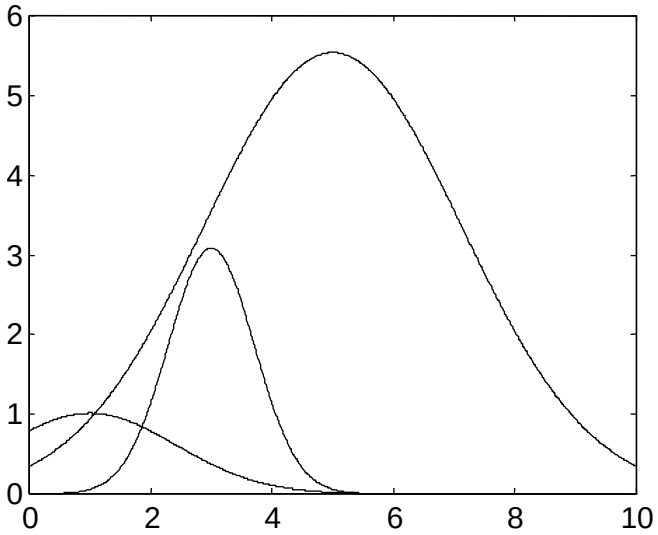
$$\mathbf{c} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}; \quad \mathbf{r} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}; \quad \mathbf{w} = \begin{bmatrix} 0.945 \\ 2.850 \\ 5.930 \end{bmatrix}$$

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







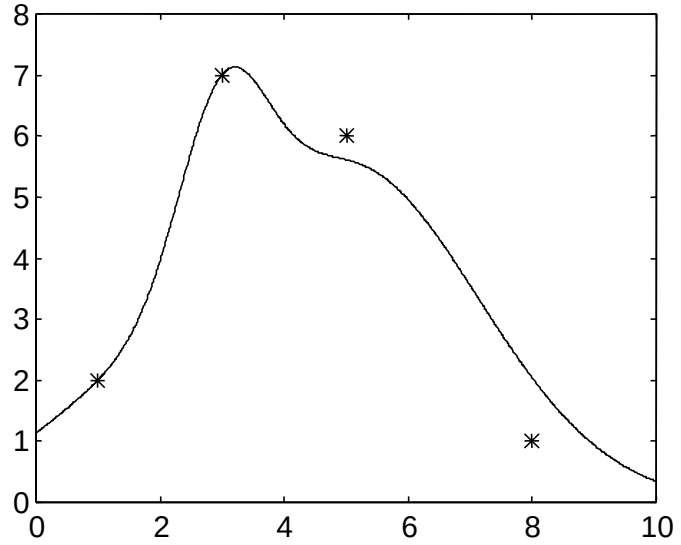


Caso 2: $m < N$

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.



1.8 Determinação dos centros e dispersões

- No caso de algoritmos que se ocupam apenas com o ajuste dos pesos da camada de saída de uma rede RBF (modelos lineares nos parâmetros), é necessário estabelecer algum critério para fixação dos centros.
- Existem critérios para o caso de número variável de centros (redes construtivas, por exemplo), mas serão mencionados aqui apenas aqueles geralmente empregados para o caso de um número fixo e previamente especificado de centros.
- Existem basicamente 3 alternativas:
    1. Espalhar os centros uniformemente ao longo da região em que se encontram os dados;
    2. Escolher aleatoriamente, ou segundo algum critério específico, um subconjunto de padrões de entrada como centros;
    3. Auto-organizar os centros, de acordo com a distribuição dos dados de entrada. Exemplo: Empregando *k-means*.
- Quanto às dispersões das funções de base radial, 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.

1.9 Aplicação das propostas de determinação de centros e dispersão

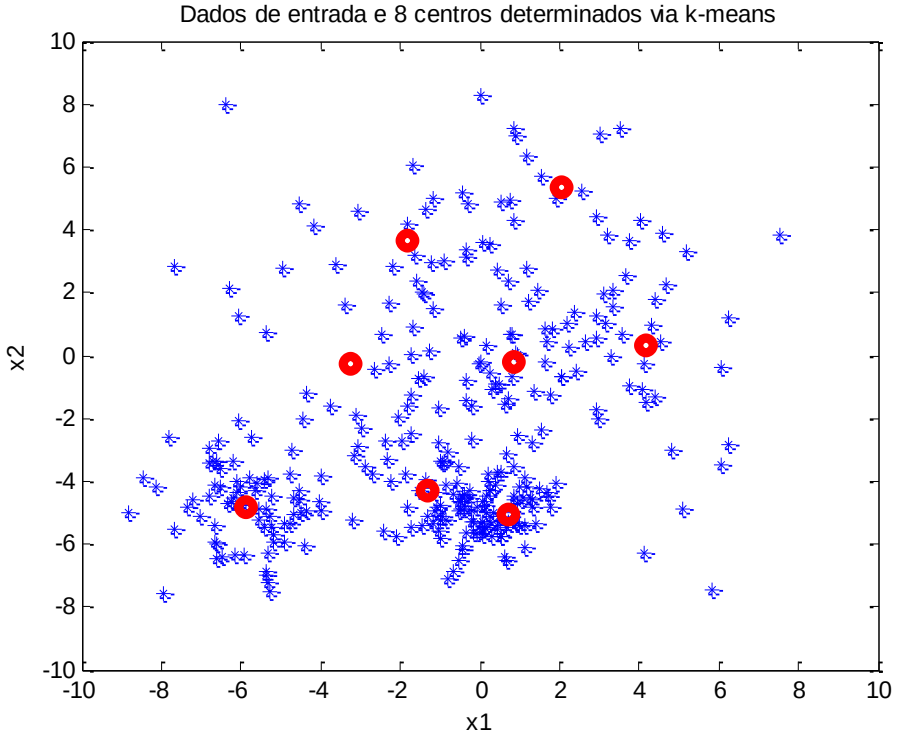


Figura 5 – Proposta de posicionamento dos centros das funções de base radial para uma rede neural RBF com 8 neurônios na camada intermediária e duas entradas, empregando o algoritmo *k-means*.

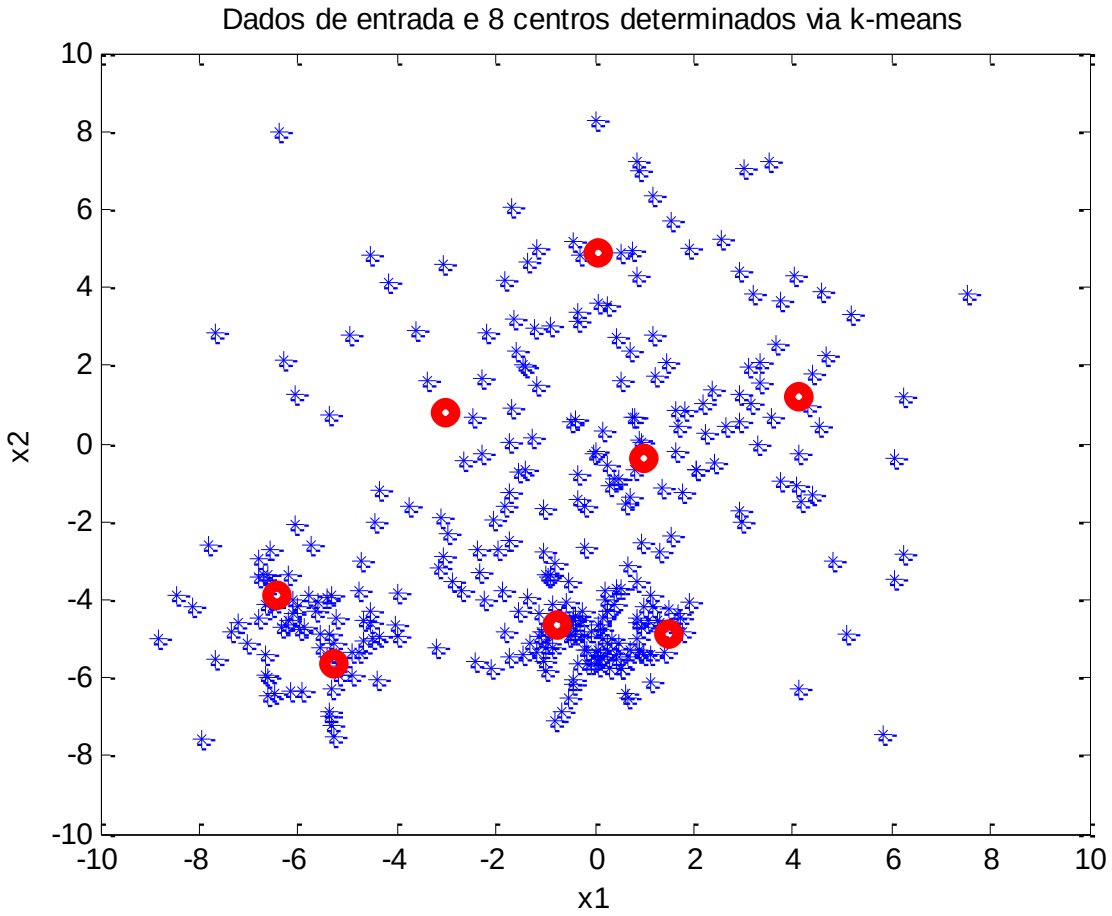


Figura 6 – Outra proposta de posicionamento dos centros para os mesmos dados, produzida por uma segunda execução do algoritmo *k-means*.

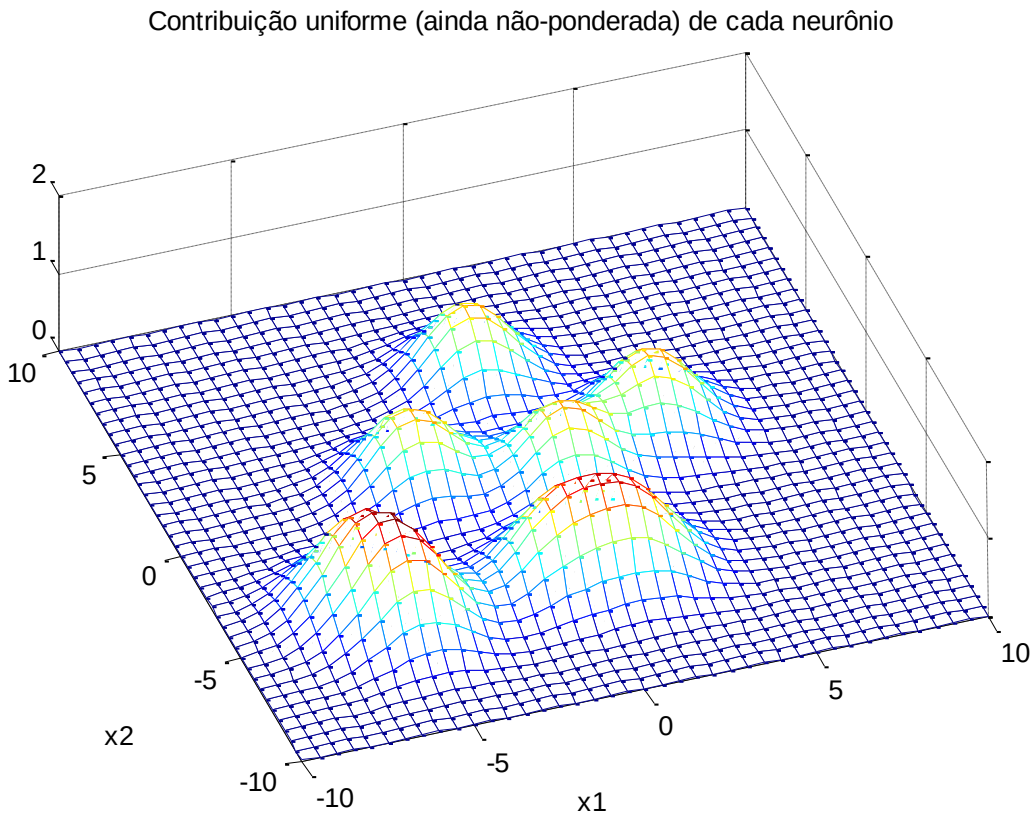


Figura 7 – Ativação dos neurônios da rede neural RBF com os centros da Figura 6, considerando todos os pesos de saída iguais a 1 e ausência de peso de bias. A dispersão é a mesma para todas as funções de ativação, dada pela fórmula da pg. 22.

- Com o critério de dispersão da pg. 22, evita-se que as funções de base radial sejam excessivamente pontiagudas, ou então com uma base demasiadamente extensa.

1.10 Referências para redes neurais RBF

BROOMHEAD, D.S. & LOWE, D. "Multivariate functional interpolation and adaptive networks", *Complex Systems*, 2: 321-355, 1988.

HAYKIN, S. "Neural Networks and Learning Machines", 3rd edition, Prentice Hall, 2008.

PARK, J. & SANDBERG, I.W. "Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2): 246-257, 1991.

1.11 Bibliografia complementar para redes RBF

BISHOP, C.M. "Improving the generalisation properties of radial basis function neural networks", *Neural Networks*, 3(4): 579-588, 1991.

BISHOP, C.M. "Neural Networks for Pattern Recognition", Clarendon Press, 1995.

CHEN, C.-L., CHEN, W.-C. & CHANG, F.-Y. "Hybrid learning algorithm for Gaussian potential function networks", *IEE Proceedings D*, 140(6): 442-448, 1993.

CHEN, S., CHNG, E.S. & ALKADHIMI, K. "Regularized Orthogonal Least Squares Algorithm for Constructing Radial Basis Function Networks", *International Journal of Control*, 64(5): 829-837, 1996.

CHEN, S., COWAN, C.F.N. & GRANT, P.M. "Orthogonal Least Squares Algorithm for Radial Basis Function Networks", *IEEE Transactions on Neural Networks*, 2(2): 302-309, 1991.

DE CASTRO, L.N. & VON ZUBEN, F.J. Automatic Determination of Radial Basis Functions: An Immunity-Based Approach. *International Journal of Neural Systems*, vol. 11, no. 6, pp. 523-535, 2001.

FREEMAN, J.A.S. & SAAD, D. "Learning and Generalization in Radial Basis Function Networks", *Neural Computation*, **7**: 1000-1020, 1995.

FRITZKE, B. "Fast learning with incremental RBF Networks", *Neural Processing Letters*, 1(1): 2-5, 1994.

GOMM, J.B. & YU, D.L. "Selecting Radial Basis Function Network Centers with Recursive Orthogonal Least Squares Training", *IEEE Transactions on Neural Networks*, 11(2):306-314, 2000.

HWANG, Y.-S. & BANG, S.-Y. "An Efficient Method to Construct a Radial Basis Function Neural Network Classifier", *Neural Networks*, 10(8): 1495-1503, 1997.

KARAYIANNIS, N.B. "Gradient Descent Learning of Radial Basis Neural Networks", *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1815-1820, 1997.

KARAYIANNIS, N.B. & MI, G.W. "Growing Radial Basis Neural Networks: Merging Supervised and Unsupervised Learning with Network Growth Techniques", *IEEE Transactions on Neural Networks*, 8(6): 1492-1506, 1997.

KUBAT, M. "Decision trees can initialize radial-basis function networks", *IEEE Transactions on Neural Networks*, 9(5): 813-821, 1998.

LIPPMANN, R.P. "Pattern Classification Using Neural Networks", *IEEE Communications Magazine*, November, pp. 47-63, 1989.

MICCHELLI, C.A. "Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions", *Constructive Approximation*, 2: 11-22, 1986.

MOODY, J. & DARKEN, C. "Fast Learning in Networks of Locally-Tuned Processing Units", *Neural Computation*, 1: 281-294, 1989.

MULGREW, B. "Applying Radial Basis Functions", *IEEE Signal Processing Magazine*, pp. 50-66, March 1996.

ORR, M.J.L. "Introduction to Radial Basis Function Networks", *Technical Report*, Centre for Cognitive Science, University of Edinburgh, Scotland, 1996. (*http://www.anc.ed.ac.uk/~mjo/papers/intro.ps*)

ORR, M.J.L. "Recent Advances in Radial Basis Function Networks", *Technical Report*, Institute for Adaptive and Neural Computation, University of Edinburgh, Scotland, 1999. (*http://www.anc.ed.ac.uk/~mjo/papers/recad.ps*)

ORR, M.J.L. "Regularisation in the Selection of Radial Basis Function Centres", *Neural Computation*, **7**(3): 606-623, 1995.

POGGIO, T. & GIROSI, F. "Networks for Approximation and Learning", *Proceedings of the IEEE*, 78(9): 1481-1497, 1990.

SUTANTO, E.L., MASON, J.D. & WARWICK, K. "Mean-tracking clustering algorithm for radial basis function centre selection. *International Journal of Control*, 67(6): 961-977, 1997.

WANG, Z. & ZHU, T. "An Efficient Learning Algorithm for Improving Generalization Performance of Radial Basis Function Neural Networks", *Neural Networks*, 13(4-5): 545-553, 2000.

WETTSCHERECK, D. & DIETTERICH, T. "Improving the Performance of Radial Basis Function Networks by Learning Center Locations", *Advances in Neural Information Processing Systems*, 4:1133-1140, 1992.

WHITEHEAD, B.A. & CHOATE, T.D. "Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction", *IEEE Transactions on Neural Networks*, **7**(4): 869-880, 1996.

WHITEHEAD, B.A. & CHOATE, T.D. "Evolving Space-Filling Curves to Distribute Radial Basis Functions Over an Input Space", *IEEE Transactions on Neural Networks*, 5(1): 15-23, 1994.

YINGWEI, L., SUNDARARAJAN, N. & SARATCHANDRAN, P. "A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks", *Neural Computation*, pp. 461-478, 1996.

2. Máquinas de aprendizado extremo (ELMs)

- Todas as propostas de redes neurais não-recorrentes (*feedfoward*) já apresentadas no curso, como o perceptron de múltiplas camadas (MLP) e a rede neural com funções de ativação de base radial (RBF), produzem a sua saída (podendo ser múltiplas saídas) como uma combinação linear das ativações dos neurônios da camada anterior.
- Tomando uma única camada intermediária, pode-se afirmar, portanto, que redes neurais MLP e RBF sintetizam mapeamentos multidimensionais de entrada-saída por meio de uma composição aditiva de funções-base, na forma:

$$\hat{s}_{kl} = \sum_{j=1}^{n} w_{kj} f\left(\mathbf{v}_j, b_j, \mathbf{x}_l\right) + w_{k0}$$

onde

- $\hat{s}_{kl}$ é a k-ésima saída da rede neural para o l-ésimo padrão de entrada $\mathbf{x}_l$;
- $f\left(\mathbf{v}_j, b_j, \bullet\right)$ é a j-ésima função da base de funções-base.

- No caso da rede neural MLP, as funções-base são funções de expansão ortogonal (*ridge functions*), enquanto que, no caso da rede neural RBF, as funções-base têm um comportamento radial em relação a um centro de ativação máxima.
- Nos dois casos, como em outros casos de composição aditiva de funções-base, há demonstração teórica da capacidade de aproximação universal. A capacidade de aproximação universal é uma **propriedade existencial**. Ela afirma que existe um número *n* finito de neurônios e uma certa configuração de pesos sinápticos que permitem obter um erro de aproximação arbitrariamente baixo para os dados de treinamento, supondo que se considera uma região compacta do espaço de entrada e que o mapeamento original, que é amostrado para produzir os dados de treinamento, é contínuo.
- É intuitivo concluir, também, que quanto maior o número *n* de neurônios na camada intermediária, maior é a flexibilidade do modelo matemático resultante, ou seja, maiores são as "possibilidades de contorção" do mapeamento a ser sintetizado.

- Por outro lado, é sabido também que há o risco de sobre-ajuste aos dados, produzindo modelos que generalizam mal frente a novos dados de entrada-saída. A máxima capacidade de generalização está associada a modelos otimamente regularizados, ou seja, que se contorcem na medida certa, de acordo com as demandas de cada aplicação.
- Com isso, uma definição adequada do número de neurônios e dos pesos sinápticos é fundamental para garantir uma boa capacidade de generalização.
- Um resultado fundamental da literatura, restrito a problemas de classificação de padrões, foi apresentado por BARTLETT (1997; 1998). Nesses trabalhos, como o próprio título indica, conclui-se que controlar a norma dos pesos sinápticos é mais relevante para a capacidade de generalização do que controlar o tamanho da rede neural, ou seja, o número *n* de neurônios na camada intermediária.
- De fato, pode-se introduzir o conceito de ⟨número efetivo de neurônios na camada intermediária⟩, o qual é determinado pela configuração dos pesos da camada de saída da rede neural.

- As máquinas de aprendizado extremo exploram este resultado “de forma extrema”, ou seja, jogam toda a responsabilidade por garantir uma boa capacidade de generalização aos pesos da camada de saída, permitindo que os pesos da camada intermediária, responsáveis por definir as funções-base, sejam definidos de modo aleatório.
- Por serem definidos de modo aleatório, portanto desvinculados das demandas da aplicação, deve-se considerar um valor elevado para *n*, podendo inclusive ultrapassar o valor de *N*, que representa o número de amostras para treinamento.
- Por mais que pareça estranho trabalhar com valores de *n* elevados e até maiores que *N*, as máquinas de aprendizado extremo se sustentam em três argumentos muito poderosos:
  - ✓ O problema de treinamento passa a ser linear nos parâmetros ajustáveis, o que representa uma enorme economia de recursos computacionais para se realizar o treinamento supervisionado;

- ✓ A capacidade de generalização pode ser maximizada controlando-se a norma dos pesos na camada de saída, não dependendo de forma significativa do número *n* de neurônios na camada intermediária;
- ✓ Há recursos computacionais disponíveis para implementar redes neurais sobredimensionadas.

- E já que as funções-base podem ser definidas aleatoriamente, então não há razão também para que elas tenham formas sigmoidais ou tenham base radial. Logo, o elenco de funções-base pode ser também arbitrário, embora as demonstrações de capacidade de aproximação universal para ELMs restrinjam ainda as alternativas de funções-base.
- Por outro lado, são incluídas funções trigonométricas e até a função sinal.

2.1 Exemplos de máquinas de aprendizado extremo

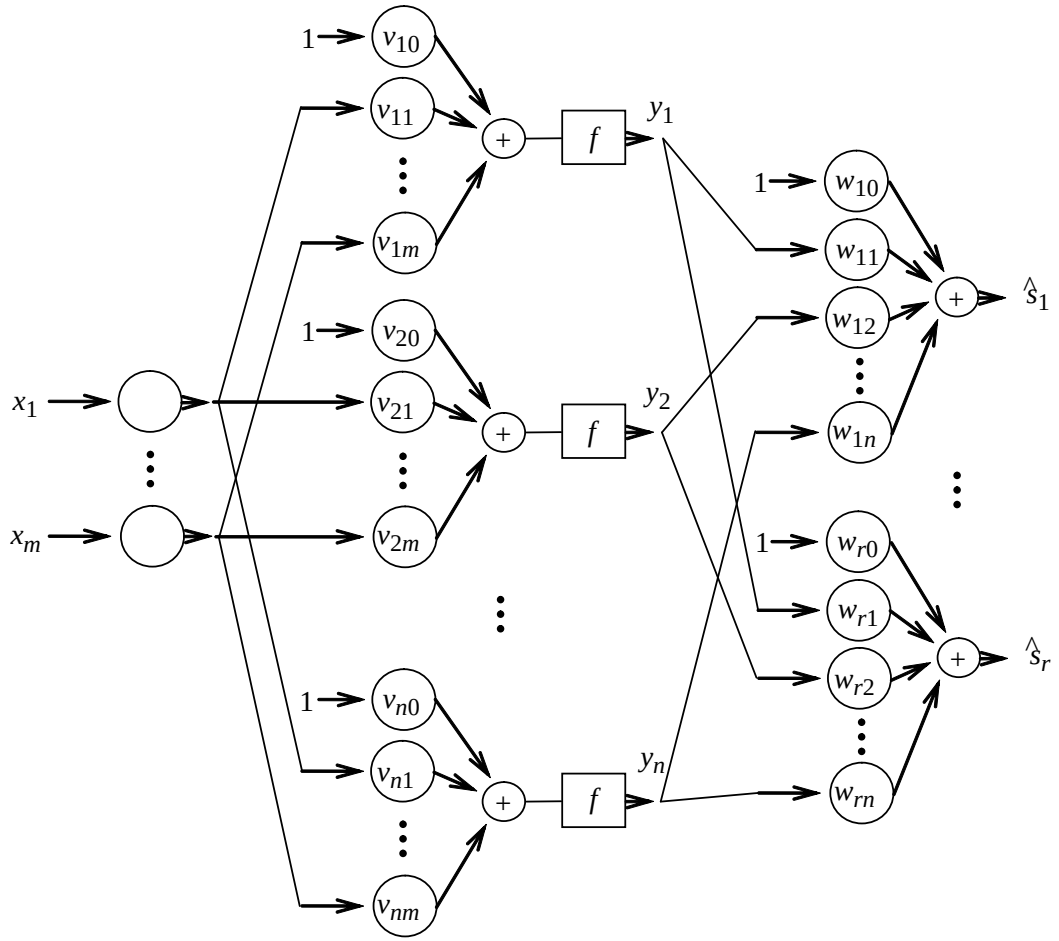


Figura 8 – Rede neural perceptron com uma camada intermediária, mas agora sem a possibilidade de ajuste das conexões sinápticas associadas á camada intermediária (pesos v_{ji}).

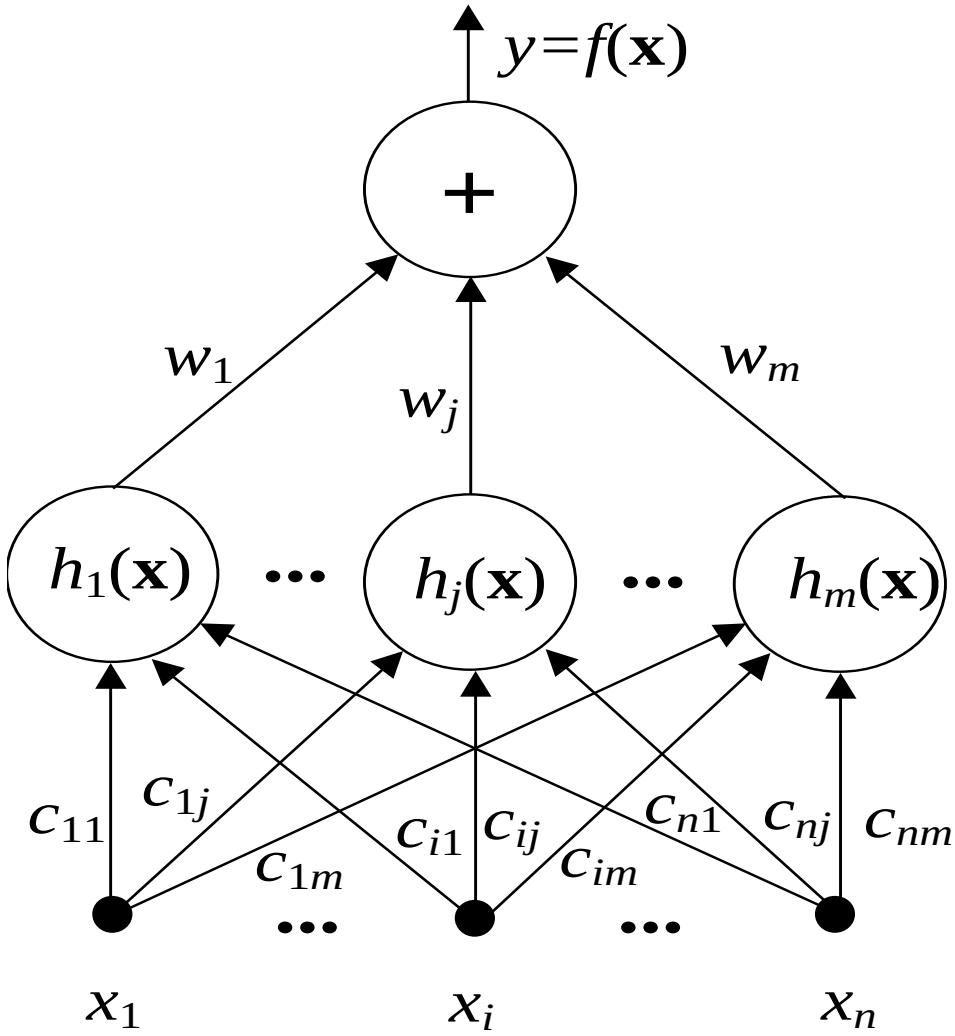


Figura 9 – Rede neural com funções de ativação de base radial. Observação: Geralmente é também considerada uma entrada constante para o neurônio de saída.

2.2 Treinamento das ELMs

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

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

onde

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

2.3 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 da camada de saída 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 como segue:

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

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

2.4 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 empregando um conjunto de validação. O valor ótimo de C_k é aquele que minimiza o erro junto ao conjunto de validação.

2.5 Referências bibliográficas para ELMs

BARTLETT, P.L. For valid generalization the size of the weights is more important than the size of the network. *Advances in Neural Information Processing Systems*, volume 9, pp. 134-140, 1997.

BARTLETT, P.L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525-536,1998.

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.

KULAIF, A.C.P. Técnicas de Regularização para Máquinas de Aprendizado Extremo. Tese de Mestrado, FEEC, Unicamp, 2014.

3. Deep learning

- A possibilidade de lidar com cada vez mais dados no treinamento de redes neurais, visando obter ganhos continuados de desempenho em tarefas de classificação, regressão e extração de atributos, levou à proposição e ao estabelecimento de redes neurais com muitas camadas intermediárias (mais que 5 camadas) e com milhões ou até bilhões de conexões sinápticas.
- Devido à existência de muitas camadas intermediárias, essas redes neurais são ditas serem de arquitetura profunda e o seu treinamento, portanto, é denominado *deep learning*. Existem camadas totalmente conectadas, como no caso de MLPs, mas como grande parte das aplicações envolve tratamento de imagens digitais, também é bastante difundido o emprego de camadas convolucionais. Também existem redes neurais recorrentes com blocos LSTM (*long short-term memory*).
- O aprendizado geralmente envolve o ajuste das conexões sinápticas e se dá de modo supervisionado, não-supervisionado ou por reforço.

3.1 Guia de temas em deep learning

1. Livro-texto:

- ✓ CHOLLET, F. "Deep Learning with Python", Manning Publications, 2017.
- ✓ GOODFELLOW, I.; BENGIO, Y. & COURVILLE, A. "Deep Learning", The MIT Press, 2016. (http://www.deeplearningbook.org/)
- ✓ GULLI, A. & PAL, S. "Deep Learning with Keras: Implementing deep learning models and neural networks with the power of Python", Packt Pub., 2017.
- ✓ PATTERSON, J. & GIBSON, A. "Deep Learning: A Practitioner's Approach", O'Reilly Media Inc., 2017.

2. Livro online:

- ✓ NIELSEN, M. "Neural Networks and Deep Learning", 2012-2019. (http://neuralnetworksanddeeplearning.com/)
- ✓ Deep Learning Book Brasil, 2018. (http://deeplearningbook.com.br/)

3. Livro de apoio:

- ✓ RASCHKA, S. "Python Machine Learning", Packt Publishing Ltd., 2015. (destaque para Chapter 13)

4. Papers marcantes + reviews:

- ✓ https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html
- ✓ LECUN, Y.; BENGIO, Y. & HINTON, G. "Deep learning", Nature, vol. 521, pp. 436-444, 28 May 2015.
- ✓ BENGIO, Y.; COURVILLE, A. & VINCENT, P. "Representation Learning: A Review and New Perspectives", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2013.
- ✓ BENGIO, Y. "Learning Deep Architectures for AI", Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

- ✓ SCHMIDHUBER, J. "Deep learning in neural networks: An overview", Neural Networks, vol. 61, pp. 85-117, 2015.
- ✓ DENG, L. "Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey", APSIPA Trans. on Signal and Information Processing, 2012.
- ✓ VINYALS, O., TOSHEV, A., BENGIO, S. & ERHAN, D. "Show and tell: A neural image caption generator", arXiv:1411.4555v2, 2015.
- ✓ XU, K., BA, J., KIROS, R., COURVILLE, A., SALAKHUTDINOV, R., ZEMEL, R. & BENGIO, Y. "Show, attend and tell: Neural image caption generation with visual attention", arXiv:1502.03044v3, 2016.

5. Vídeos / apresentações:

- ✓ https://www.youtube.com/watch?v=_1Cyyt-4-n8
- ✓ Tutorial proferido por Geoffrey Hinton, Yoshua Bengio & Yann LeCun [http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf].
- ✓ E muito mais cursos e tutoriais ...

6. Cursos acadêmicos:

- ✓ Aprendizado de máquina – Andrew Ng – Stanford University (Coursera) [https://www.coursera.org/learn/machine-learning]
- ✓ Redes neurais para aprendizado de máquina – Geoffrey Hinton – University of Toronto [https://www.coursera.org/learn/neural-networks]
- ✓ Aprendizado de máquina – Nando de Freitas – Oxford University [https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/]
- ✓ Redes neurais artificiais – Hugo Larochelle – Université de Sherbrooke [http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html]
- ✓ Redes convolucionais para reconhecimento de padrões em imagens – Andrej Karpathy (versão 2016) – Stanford University [http://cs231n.stanford.edu/]
- ✓ Processamento de linguagem natural com *deep learning* – Christopher Manning – Stanford University [http://web.stanford.edu/class/cs224n/] e David Socher – Stanford University [http://cs224d.stanford.edu/]

7. Escolha de hiperparâmetros:

- ✓ How do we know how many layers to use, how many conv layers, what are the filter sizes, or the values for stride and padding? These are not trivial questions and there isn't a set standard that is used by all researchers. This is because the network will largely depend on the type of data that you have. Data can vary by size, complexity of the image, type of image processing task, and more. When looking at your dataset, one way to think about how to choose the hyperparameters is to find the right combination that creates abstractions of the image at a proper scale. (https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/)
- ✓ BERGSTRA, J., BARDENET, R., BENGIO, Y. & KÉGL, B. "Algorithms for Hyper-Parameter Optimization", Advances in Neural Information Processing Systems 24 (NIPS'2011), 9 pages, 2011.

8. Transfer learning:

- ✓ Mesmo não dispondo de "zilhões" de amostras, é possível produzir redes neurais com arquiteturas profundas de alto desempenho, adotando na etapa de pretraining os extratores de atributos de redes neurais treinadas (por outros desenvolvedores) com "zilhões" de amostras e foco de aplicação distinto.

9. Outros conceitos relevantes:

- ✓ ReLU (e suas extensões)
- ✓ Mini-batch: LI, M., ZHANG, T., CHEN, Y. & SMOLA, A.J. "Efficient Mini-batch Training for Stochastic Optimization", KDD'2014.
- ✓ Manifold (high-dimensional data lies in a low-dimensional manifold)
- ✓ Softmax (https://en.wikipedia.org/wiki/Softmax_function)
- ✓ Cross-entropy: JANOCHA, K. & CZARNECKI, W.M. "On Loss Functions for Deep Neural Networks in Classification", Schedae Informaticae, vol. 25, pp. 49-59, 2016.

10. Data augmentation:

- ✓ Estratégia para evitar overfitting: Wu, R., Yan, S., Shan, Y., Dang, Q. & Sun, G. "Deep Image: Scaling up Image Recognition", arXiv:1501.02876v2, 2015.

11. Dropout:

- ✓ Estratégia para evitar overfitting: SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I. & SALAKHUTDINOV, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

12. Autoencoder (pretraining):

- ✓ VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y. & MANZAGOL, P.-A. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion", Journal of Machine Learning Research, vol. 11, pp. 3371-3408, 2010.

13. Outras técnicas que contribuem para a regularização e o aprendizado:

- ✓ Max-norm regularization (permite taxas de aprendizado mais altas)
- ✓ Normalização / Escalamento da saída de cada camada intermediária
- ✓ Weight sharing
- ✓ Pooling | Downsampling

14. Métodos de otimização para o ajuste de pesos:

- ✓ LE, Q.V., NGIAM, J., COATES, A., LAHIRI, A., PROCHNOW, B., NG, A.Y. "On optimization methods for deep learning", Proceedings of the 28th International Conference on Machine Learning (ICML'2011), pp. 265-272, 2011.
- ✓ PASCANU, R. & BENGIO, Y. "Revisiting natural gradiente for deep networks", arXiv:1301.3584v7, 2014.

15. Camadas convolucionais:

- ✓ ZEILER, M.D. & FERGUS, R. "Visualizing and Understanding Convolutional Networks", Proceedings of the European Conference on Computer Vision (ECCV'2014), Lect. N. on Comp. Science, Springer, vol. 8689, pp. 818-833, 2014.
- ✓ https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

16. Deep Recurrent Neural Networks

- ✓ GREFF, K., SRIVASTAVA, R.K., KOUTNÍK, J., STEUNEBRINK, B.R. & SCHMIDHUBER, J. "LSTM: A Search Space Odyssey", IEEE Transactions on Neural Networks and Learning Systems, to appear, 2017.
- ✓ VISIN, F., KASTNER, K., CHO, K., MATTEUCCI, M., COURVILLE, A. & BENGIO, Y. "ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks", arXiv:1505.00393v3, 2015.

- ✓ CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., BENGIO, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, arXiv:1406.1078v3, 2014.
- ✓ Christopher Olah [http://colah.github.io/posts/2015-08-Understanding-LSTMs/]
- ✓ Andrej Karpathy [http://karpathy.github.io/2015/05/21/rnn-effectiveness/]

17. Attention models:

- ✓ BA, J., MNIH, V. & KAVUKCUOGLU, K. "Multiple object recognition with visual attention", arXiv:1412.7755v2, 2015.

18. Generative Adversarial Nets:

- ✓ GOODFELLOW, I.J., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIRY, S., COURVILLE, A., BENGIO, Y. "Generative Adversarial Nets", arXiv:1406.2661v1, 2014.

- ✓ RADFORD A.; METZ, L.; CHINTALA, S "Unsupervised representation learning with deep convolutional generative adversarial networks", arXiv:1511.06434v2, 2016.
- ✓ CHEN, X., DUAN, Y., HOUTHOOFT, R., SCHULMAN, J., SUTSKEVER, I. & ABBEEL, P. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets", arXiv:1606.03657v1, 2016.
- ✓ http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

19. Restricted Boltzmann machines + Deep Belief Networks:

- ✓ FISCHER, A. & IGEL, C. "An Introduction to Restricted Boltzmann Machines", Proceedings of the 17th Iberoamerican Congress on Pattern Recognition (CIARP'2012), Lecture Notes on Computer Science, Springer, vol. 7441, pp. 14-36, 2012.
- ✓ http://www.cs.toronto.edu/~hinton/deeprefs.html
- ✓ https://deeplearning4j.org/restrictedboltzmannmachine

20. Deep Reinforcement Learning:

- ✓ LI, Y. "Deep Reinforcement Learning: An Overview", arXiv:1701.07274v2, 2017.
- ✓ http://hunch.net/~beygel/deep_rl_tutorial.pdf
- ✓ Gorila (General Reinforcement Learning Architecture)
- ✓ Games e sistemas de recomendação

21. Ambientes de programação para *deep learning*:

- ✓ TensorFlow e Pytorch representam as duas bibliotecas de código aberto mais populares para *deep learning*.
- ✓ TensorFlow foi desenvolvido pelo Google e é adotado em várias soluções internas, como no sistema de reconhecimento de fala, no Gmail e no sistema de busca.
- ✓ PyTorch está sendo desenvolvido por pesquisadores da NVIDIA e de importantes universidades, como Stanford, Oxford e ParisTech. Twitter e Facebook usam PyTorch.

- ✓ Theano é outra biblioteca de código aberto para *deep learning*, muito similar ao TensorFlow.
- ✓ Keras é outra biblioteca para modelos de *deep learning*, podendo encapsular códigos de Theano e TensorFlow e, assim, implementar modelos complexos em poucas linhas de código.
- ✓ https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software
- ✓ https://en.wikipedia.org/wiki/Deep_learning#Software_libraries

22. Casos de estudo que evidenciam o poder de *deep learning*:

- ✓ No link [https://www.quora.com/How-do-I-learn-deep-learning-in-2-months], foram apresentadas frentes de aplicação que podem ser tomadas como casos de estudo (há hiperlinks de fácil acesso):

- As is tradition, start with classifying the MNIST dataset
- Try face detection and classification on ImageNet. If you are up to it, do the ImageNet Challenge 2016.
- Do a Twitter sentiment analysis using RNNs or CNNs
- Teach neural networks to reproduce the artistic style of famous painters (A Neural Algorithm of Artistic Style)
- Compose Music With Recurrent Neural Networks
- Play ping-pong using Deep Reinforcement Learning
- Use Neural Networks to Rate a selfie
- Automatically color Black & White pictures using Deep Learning

Observação: O material a ser usado para cobrir Deep Learning é aquele associado ao Tópico 8 do curso de pós-graduação IA353, no link:

http://www.dca.fee.unicamp.br/~vonzuben/courses/ia353.html