

MULTILAYER PERCEPTRON (MLP)

IMPROVE
MOVEMENT

AGENDA

1. Data Split
2. Acelerando o treinamento
3. Configuração da topologia
4. Algoritmo Rprop
5. Cross-entropy cost function
6. Softmax
7. Função de ativação ReLU
8. Regularização

REPRESENTAÇÃO

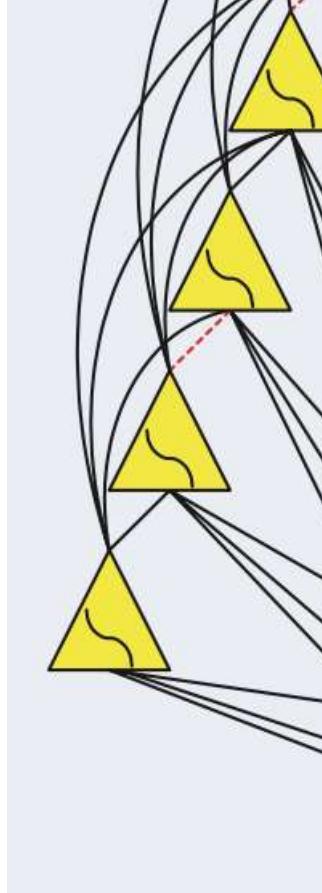
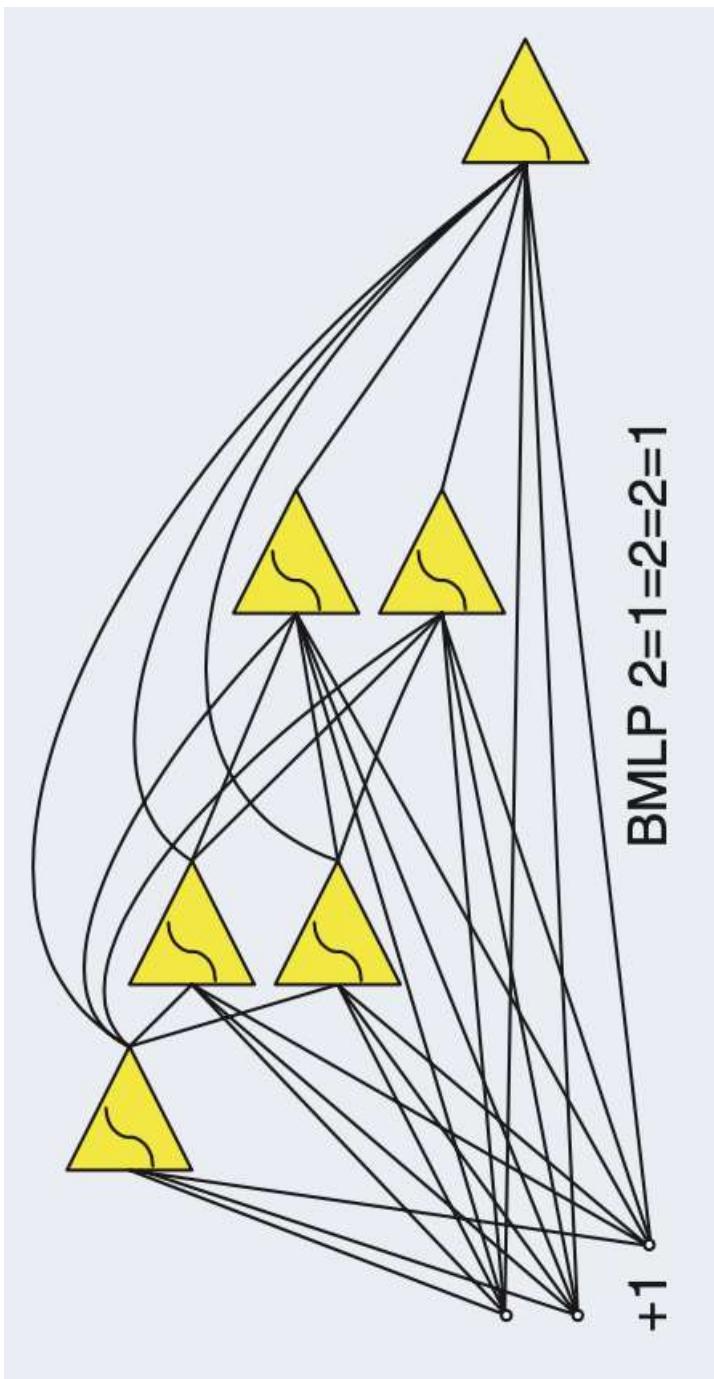
- O que cada neurônio de uma rede MLP representa
 - Neurônios das camadas ocultas?
 - Neurônios das camadas de saída?
- Em relação ao problema XOR, o que isso significa?

REPRESENTAÇÃO: ALGUMAS CONSIDERAÇÕES

• Redes com uma única camadas são limitadas

- Unidades intermediárias (ocultas) podem “melhorar” a representação do sinal de entrada
 - ▶ Novas representações (atributos)

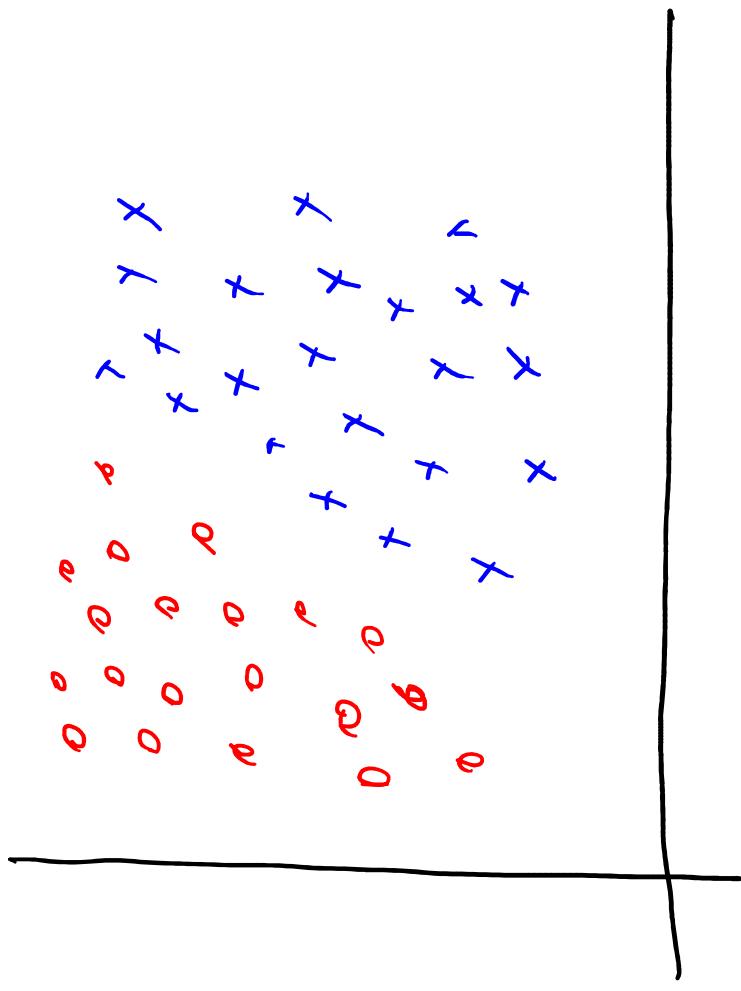
MLP - ARQUITETURAS NÃO TR



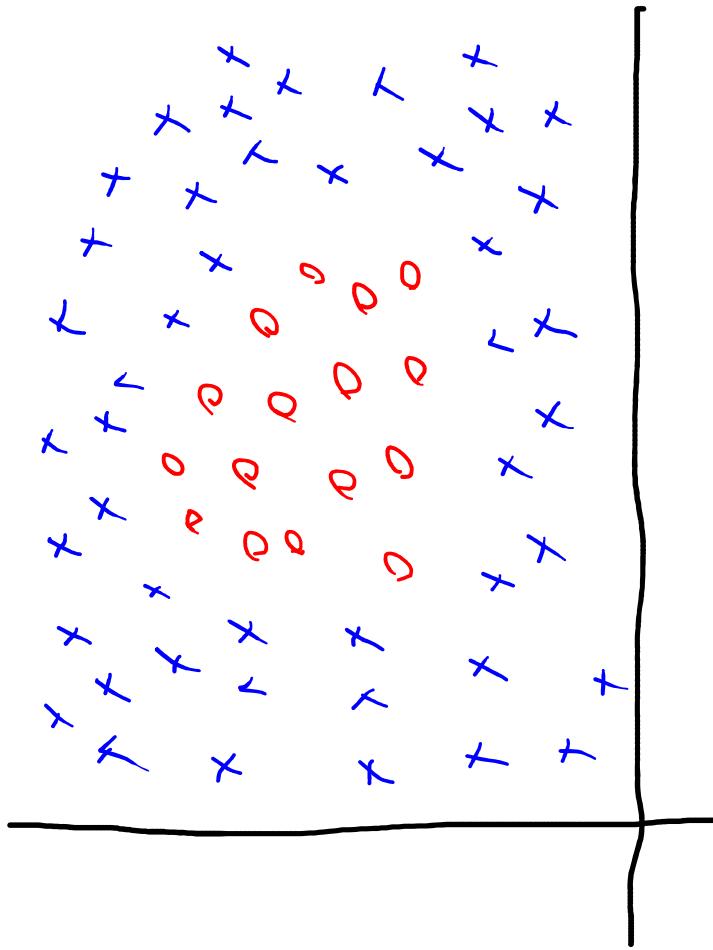
REPRESENTAÇÃO: ALGUMAS CONSIDERAÇÕES

- Problema apontado por Minsky & Papert (1969):
 - Existe uma regra muito clara de como treinar redes com um (perceptrons)
 - Contudo, não possuímos um algoritmo igualmente eficiente redes com camadas ocultas
- Backpropagation:
 - Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J.

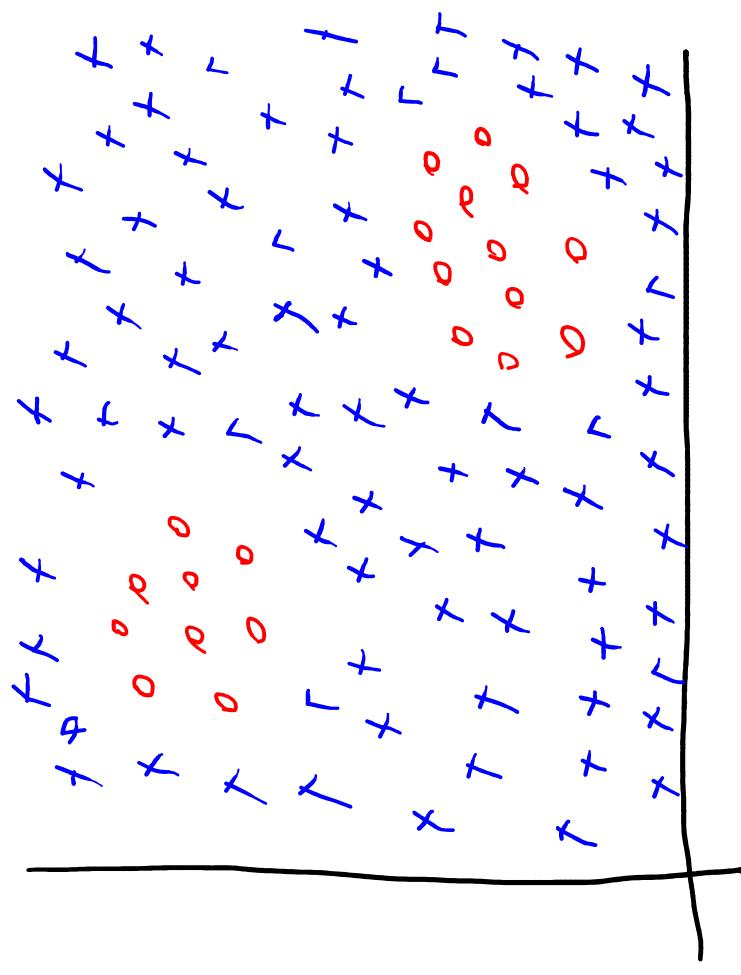
CONFIGURAÇÃO MANUAL DA TOPOLOGIA



CONFIGURAÇÃO MANUAL DA TOPOLOGIA



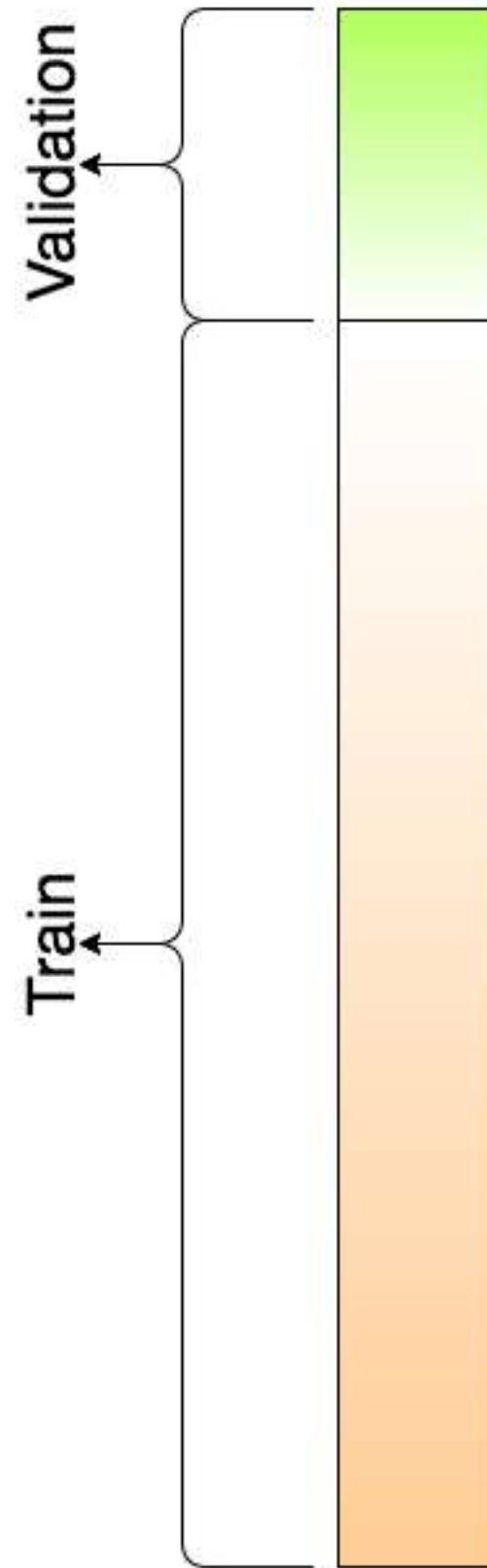
CONFIGURAÇÃO MANUAL DA TOPOLOGIA



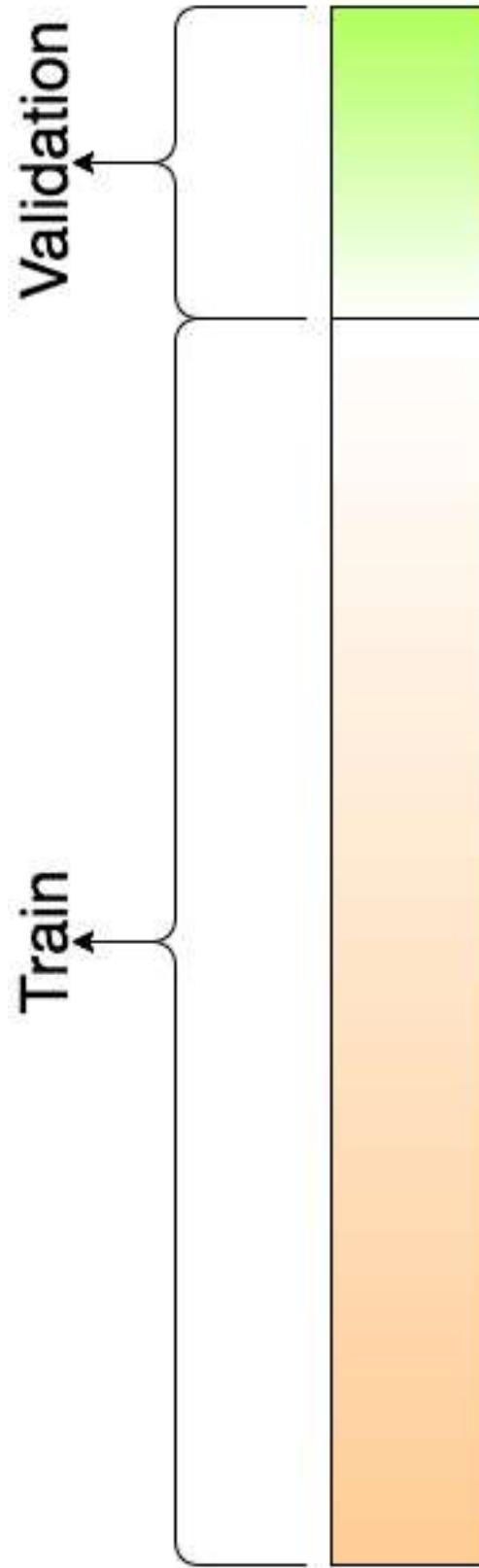
SETUP E TREINAMENTO DA RE

- O algoritmo de retropropagação permite o treinamento com múltiplas camadas. Porém, ainda temos alguns
- ▶ Otimização do **treinamento** (*convergência*)
- ▶ Generalização (*topologia/regularização*)

HOW TO SPLIT THE DATASETS

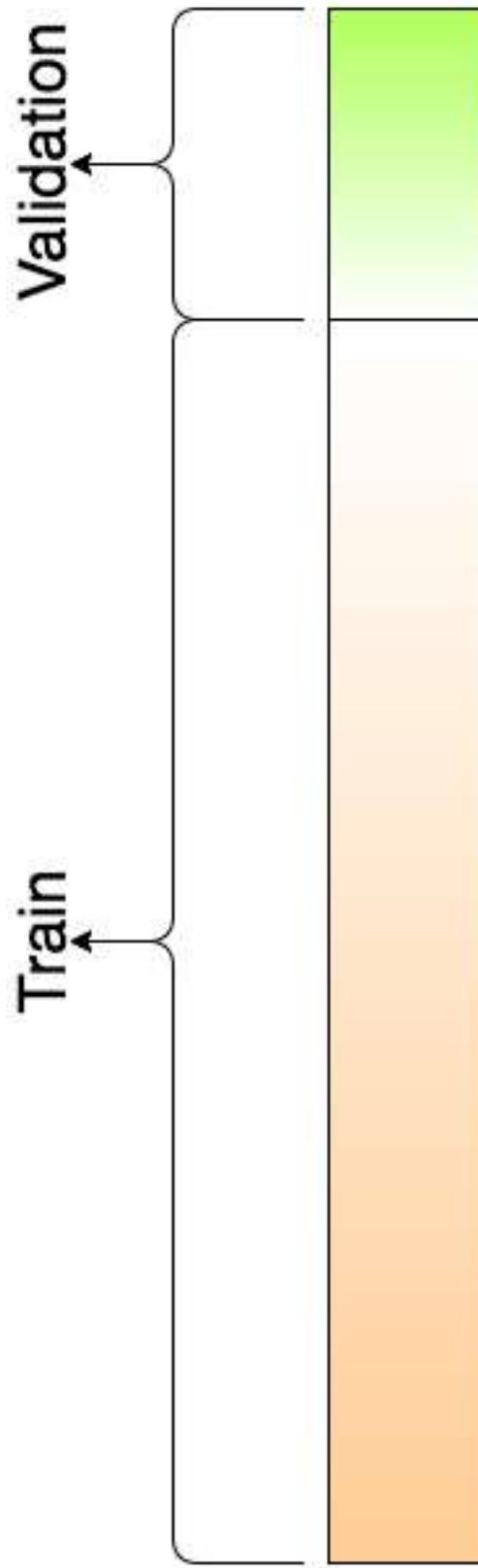


DATASETS



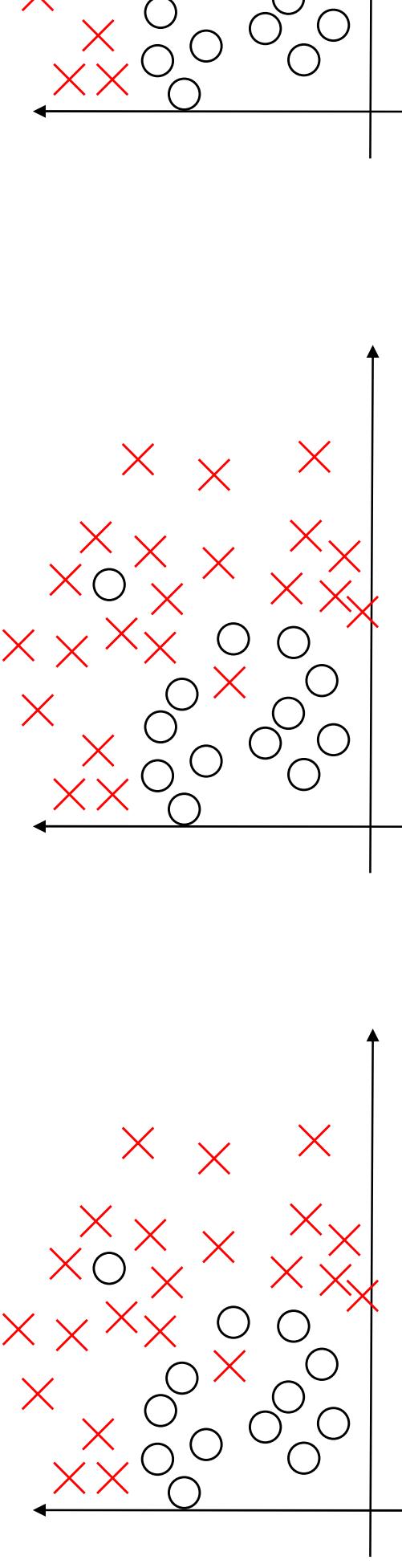
- What is the proportion of each set? (split ratio)

DATASETS

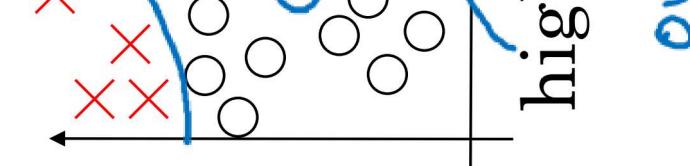
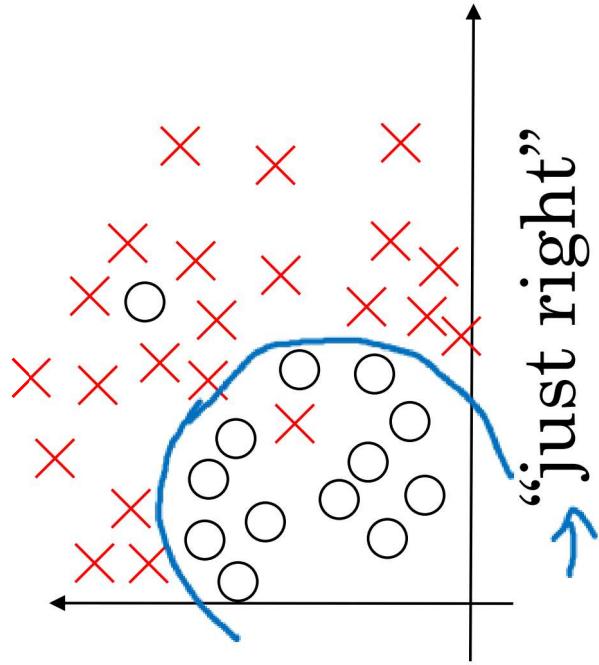
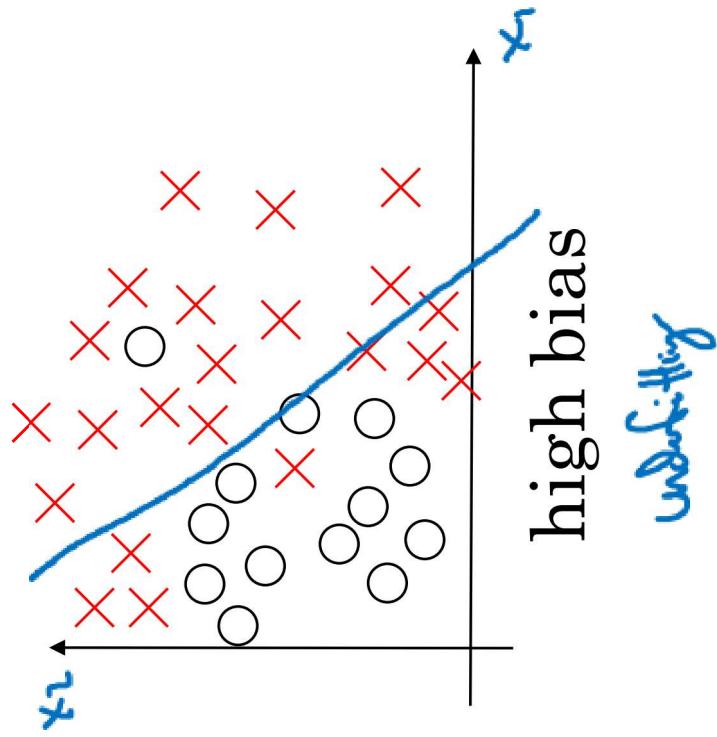


- Training, validation (dev), and test sets **must** come from the same distribution
- Test set (is it mandatory) ?

BIASS VS. VARIANCE



BIASS VS. VARIANCE



BIAS VS. VARIANCE

- ▶ How to define the architecture of the network?
- ▶ Hyperparameters vs. parameters

BIAS VS. VARIANCE

► **Adjust Bias (Architecture search) - Training data:**

1. Larger networks
2. Training longer

► **Adjust Variance - Validation data:**

1. Get more data
2. Regularization

COMO DEFINIR A TOPOLOGIA DA REDE

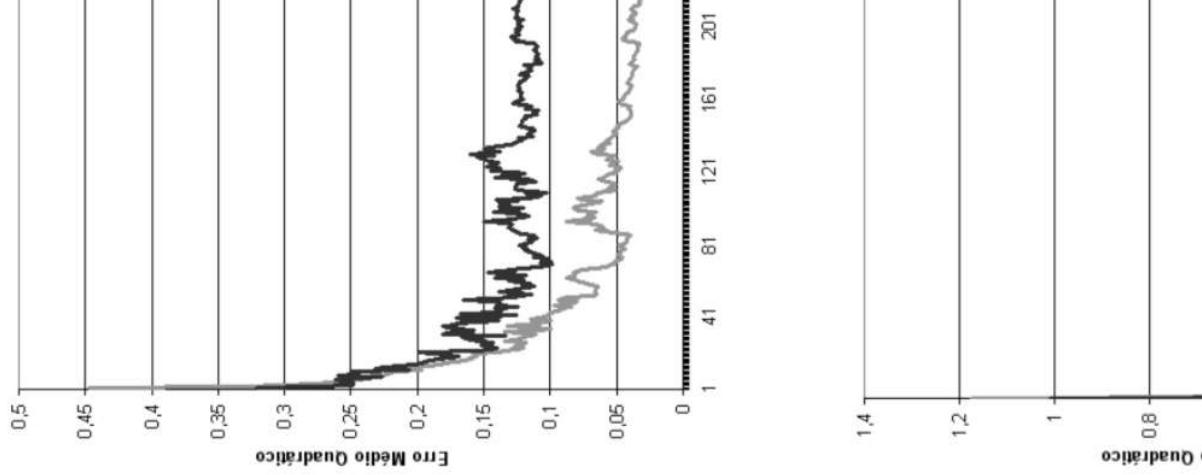
- Busca-se por uma rede com a menor quantidade de neurônios capaz de resolver o problema
 - ▶ Algoritmos Genéticos (ou outros métodos de otimização)
 - ▶ Avaliação dividindo o conjunto em treinamento e validação cruzada (*Cross-Validation*)
 - ▶ *Stratified Cross-Validations*
 - ▶ *K-fold cross validation*

<https://optur>

ACELERANDO O TREINAMENTO

- Formas de treinamento (revisão)
- Configuração inicial dos pesos
- Taxa de aprendizagem
- Normalização do conjunto de dados
- Utilizar o termo de momentum
- Adicionar nós intermediários

FORMAS DE TREINAMENTO



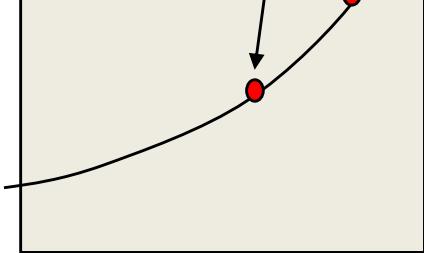
- On-line (padrão)
- Em lote (batch)
- Mini-batch

CONFIGURAÇÃO INICIAL DOS PESOS

- Iniciar a rede com pesos pequenos e aleatórios
 - Por que valores aleatórios?
 - Por que valores pequenos?
- Podemos ainda ponderar o valor inicial pela quantidade de sinapses do neurônio
 - Por quê?

TAXA DE APRENDIZAGEM

- Fixa vs. decrescente
- Como realizar o decaimento?
- Taxa única para toda a rede?



↑
 E

ALGUMAS HEURÍSTICAS

- **Heurística I:** Utilizar taxas de aprendizagem diferentes para cada neurônio/peso)
- **Heurística II:** Variar a taxa de aprendizagem sempre observando o erro observado
- **Heurística III:** Se a derivada da função de custo respeito a um peso tem o mesmo sinal durante as iterações, a taxa de aprendizagem deve ser aumentada.

NORMALIZAÇÃO DO CONJUNTO D

Unnormalized:

1. Centrar a média em zero

2. Variância = 1

3. Utilizar atributos não correlacionados



4. Cuidado: parâmetros para conjunto Normalizado:
de treinamento, validação e testes devem ser os mesmos

TERMO DE MOMENTUM

- ▶ Reduz o perigo de instabilidade
- ▶ Gera uma inércia na descida do gradiente
- ▶ Exponentially weighted averages

$$\Delta W^t = \eta \delta + \alpha \Delta W^{t-1}$$

$$\Delta W^t = \eta((1 - \alpha)\delta + \alpha \Delta W^{t-1})$$

Dropout

A MORE EFFICIENT VARIATION
BACKPROPAGATION ALGORITHM

PROBLEMAS DO BACKPROPAGATION

- Não possuímos informação completa sobre a função de custo
- Dificuldades para encontrar (configurar) uma taxa de aprendizado adequada
- Mínimo local em E
- Plateaus
- Oscilações
- As adaptações são realizadas em função da derivada parcial da função de custo

PROBLEMAS DO BACKPROPAGATION

- O que o algoritmo de retropropagação faz?
- Ele modifica os pesos em função da derivada parcial (∂)
- **Problema:** A magnitude da derivada não necessariamente representa o valor de correção adequado aos pesos
- **Possível solução:** Não utilizar o valor da derivada apenas seu sinal (direção) - $Rprop$

O ALGORITMO RPROP

- Criado por Martin Riedmiller em 1992
- Algoritmo de primeira ordem otimizado
- Utiliza apenas o sinal da derivada parcial da erro sobre todos os padrões do dataset
- Ignora sua magnitude (backpropagation)
- Age de forma isolada para cada peso da rede

O ALGORITMO RPROP

- Apresenta um esquema de aprendizado eficiente
- Realiza a adaptação direta dos pesos baseando-se na informação do gradiente local
 - Processamento em Lote
- Princípio básico: elimina a influência do valor da derivada na correção dos pesos.
- **Qual o significado disso?**
- Ele considera apenas o sinal (direção) da energia do erro não

RPROP: RESILIENT PROPAGATION

Backprop	RProp
$\Delta w_{ij} = -\eta (\partial E / \partial w_{ij})$	$\Delta w_{ij} = -\text{sign}(\partial E / \partial w_{ij})$

RPROP: ATUALIZAÇÃO

- Δ_{ij} representa a atualização do peso ij
- A taxa de atualização é definida exclusivamente por Δ_{ij}
- Δ_{ij} evolui ao longo do processo de treinamento baseado da derivada da energia do erro

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)} & \text{if } \frac{\partial E}{\partial w_{ij}}^{(t-1)} < 0 \end{cases} * \frac{\partial E}{\partial w_{ij}}$$

RPROP: ATUALIZAÇÃO

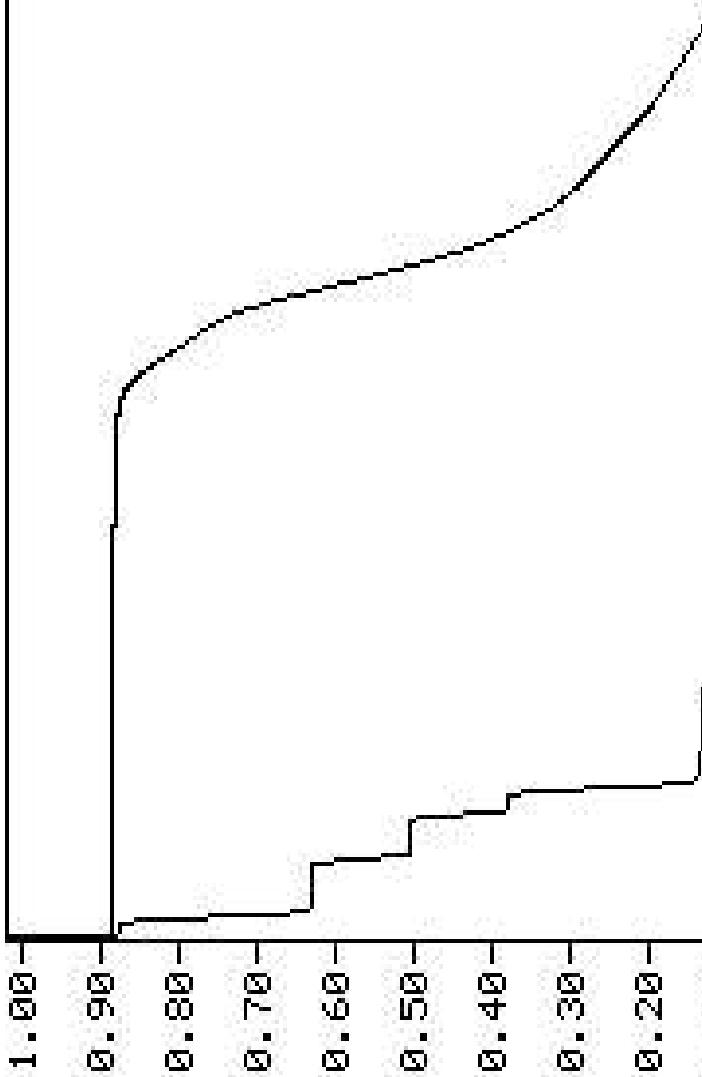
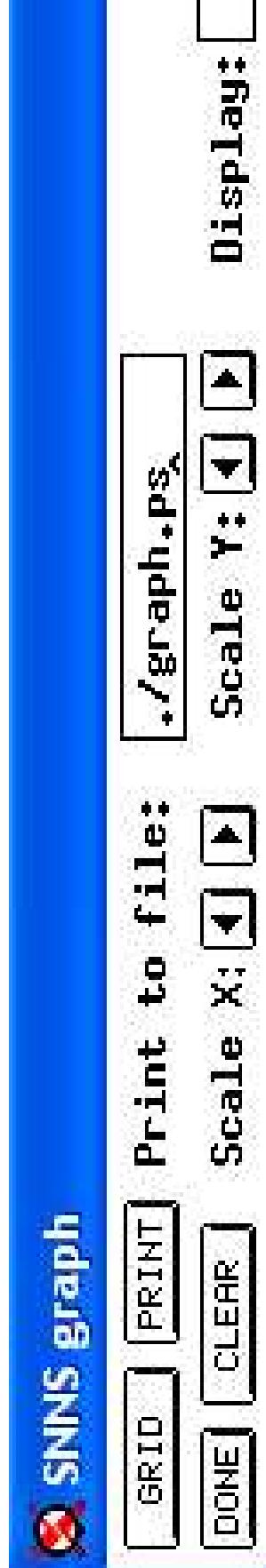
- A atualização segue a seguinte regra:
 - Se a derivada é positiva, o peso é decrementado de Δ
 - Se a derivada é negativa, o peso é incrementado de Δ

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t) & \text{if } \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ +\Delta_{ij}(t) & \text{if } \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ 0 & \text{else} \end{cases}$$

RPROP: PARÂMETROS

- Fatores de ajuste da taxa de atualização
 - ▶ $\eta^- = 0.5$ (fator de decremento)
 - ▶ $\eta^+ = 1.2$ (fator de incremento)
- Limites:
 - ▶ $\Delta_{\max} = 50.0$ (limite superior)
 - ▶ $\Delta_{\min} = 1e-6$ (limite inferior)
- Valores iniciais:

COMPARAÇÃO



COMPARAÇÃO

Tabela 5.6: Resultado do Processo de Treinamento para a Rede Neural para o Aprendizado da Posição e Forma dos Objetos - Topologia 4800X60X5

	Iterações	EQM	Desvio Padrão	Tempo Treinamento
Rprop	38	0,093670	0.02153000	≈ 50min
BP em Lote	2134	0.102270	0.01125000	≈ 35h
BP Padrão	500	0.099153	0.02271200	≈ 14min

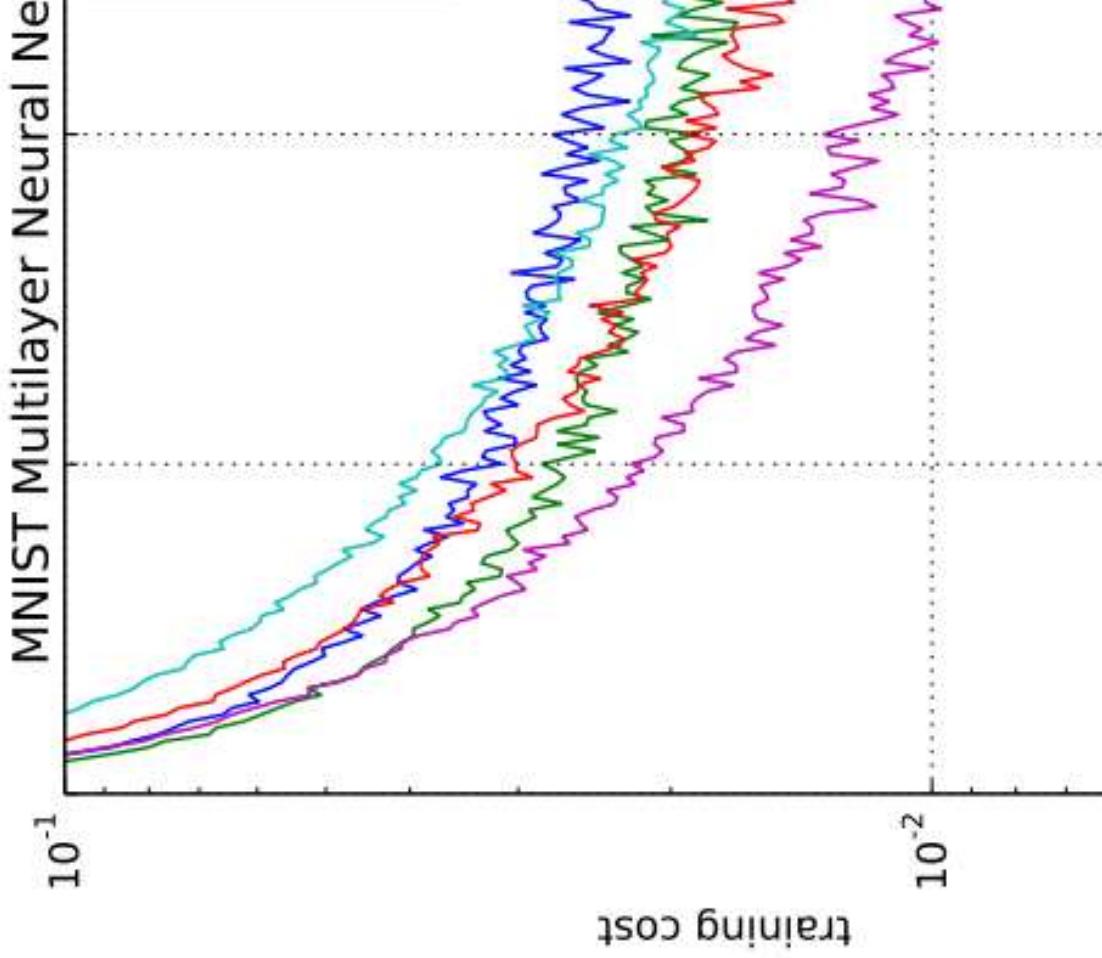
Tabela 5.7: Resultado do Processo de Treinamento para a Rede Neural para o Aprendizado da Posição e Forma dos Objetos - Topologia 4800X69X4.

	Iterações	EQM	Desvio Padrão	Tempo Treinamento

BIBLIOGRAFIA RPROP

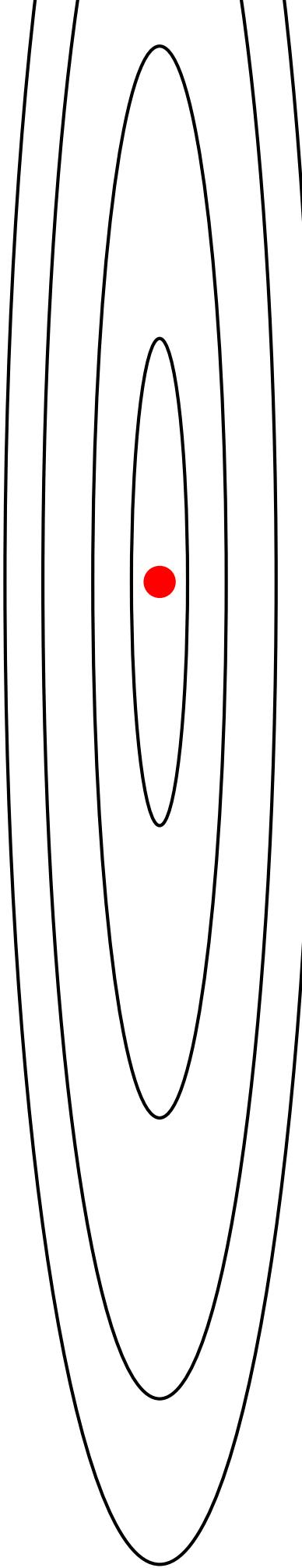
- 1.** Riedmiller, M. & Braun, H. (1992). Rprop- a fast learning algorithm. In Proc. of ISCI S VII.
- 2.** Riedmiller, M. (1994b). Rprop - description and implementation details. Technical report, Universität Karlsruhe - Institut für Logic, Komplexität und Deduktionssysteme.

OTHER ALGORITHMS



OTHER ALGORITHMS - RMSPROP

- RMSProp - Proposed by Geoff Hinton (unpublished)



OTHER ALGORITHMS - RMSPROP

- ▶ RMSProp - Proposed by Geoff Hinton (unpublished)
 - It keeps the moving average of the squared gradient of each weight

$$S_w = \beta S_w + (1 - \beta) \left(\frac{\delta_E}{\delta_w} \right)^2$$

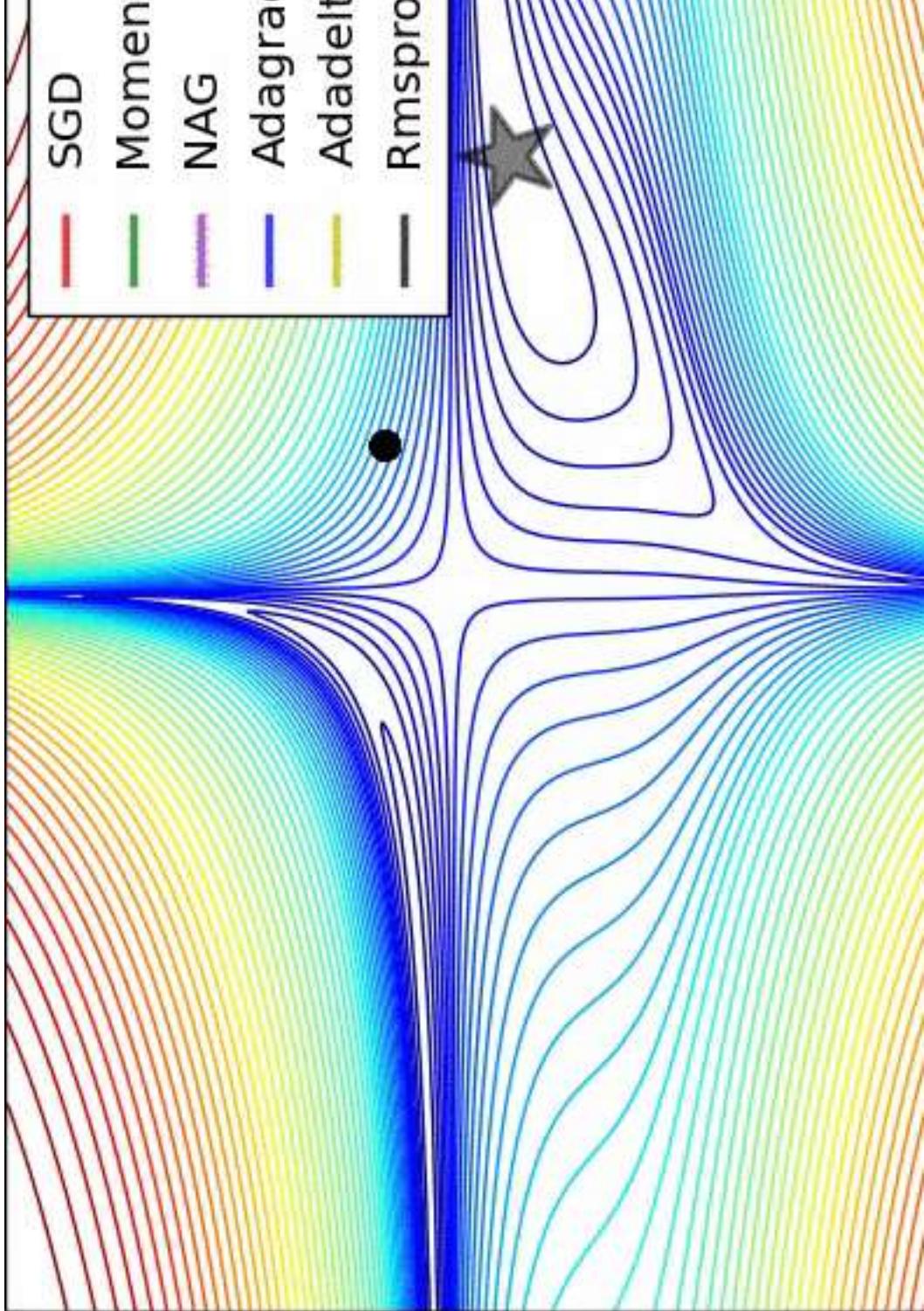
$\propto \delta_E$

OTHER ALGORITHMS - ADAM

- ▶ Adam [arxiv:1412.6980] (RMSProp with momentum)

$$M_w = \beta_1 M_w + (1 - \beta_1) \left(\frac{\delta E}{\delta w} \right)$$
$$S_w = \beta_2 S_w + (1 - \beta_2) \left(\frac{\delta E}{\delta w} \right)^2$$

ALGORITHMS (OPTIMIZERS)



COST FUNCTION

EVALUATING THE

FUNÇÃO DE CUSTO DO BACKPROPAGA

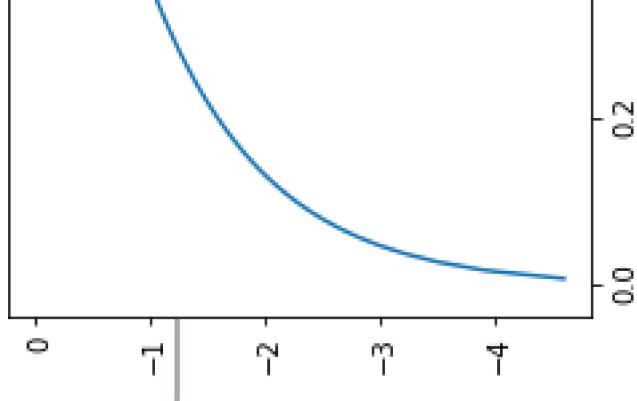
1. Função Quadrática:

$$C = \frac{1}{m} \sum_j (t_j - s_j)$$

FUNÇÃO DE CUSTO DO BACKPROPAGA

- Se considerarmos um problema de classificação gradiente dessa função, pode, em certas circunstâncias, permanecer muito tempo próximo a zero
- Uma alternativa:
 - ▶ *Função Cross-Entropy*

CROSS-ENTROPY



$$C = -\frac{1}{n} \sum_j [t_j \ln(s_j) + (1 - t_j) \ln(1 - s_j)]$$

- Qual é o funcionamento dessa equação?

► **Custo alto quando o valor desejado está longe**

CROSS-ENTROPY

$$C = -\frac{1}{n} \sum_j [t_j \ln(s_j) + (1 - t_j) \ln(1 - s_j)]$$

- Atualização dos pesos (após derivação em relação à w_{ij})

$$\frac{\partial C}{\partial w_{ji}} = -\frac{1}{n} \sum_j x_j (s_i - t_j)$$

ILUSTRAÇÃO

- <http://neuralnetworksanddeeplearning.com/>

RESUMO FUNÇÕES DE CUSTO

- Muitas funções disponíveis
- Seleção depende da aplicação
- Ver exemplos em:

► <https://keras.io/api/losses/>

► <https://pytorch.org/docs/stable/nn.html#loss-functions>

NEURON

ACTIVATION

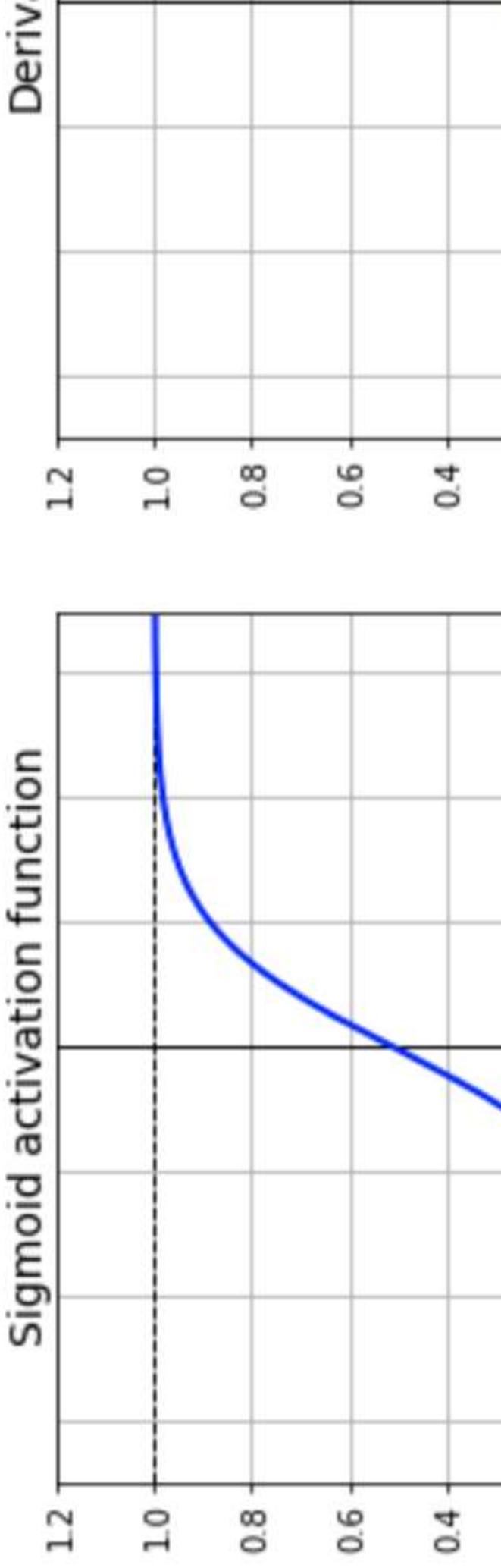
ACTIVATION FUNCTIONS: VÁRIAS

- Diversas funções disponíveis na literatura, principais:
 - Linear
 - Sigmoid: logística e tanh
 - ReLU (e variações)
 - Softmax (layer)
- Ver ilustrações em: <https://towardsdatascience.com/nonlinear-activation-functions-v0ii-should-know-in>

ACTIVATION FUNCTIONS: SIGMOIDÍSTICA

- Função:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

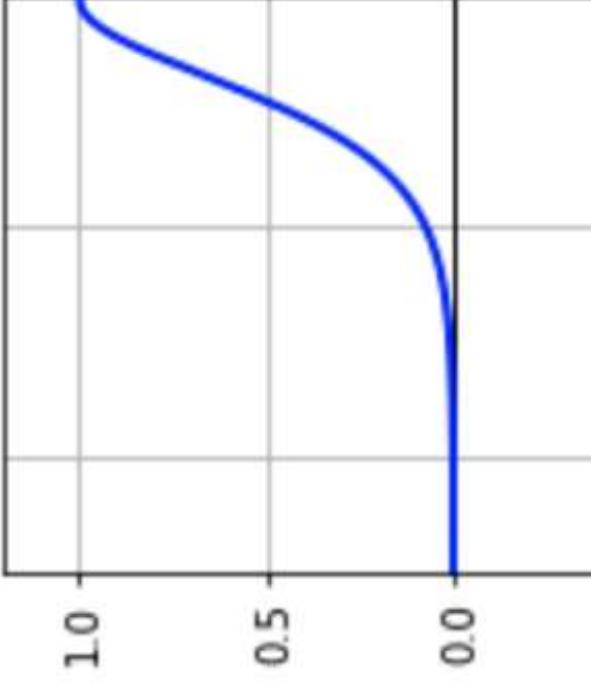


ACTIVATION FUNCTIONS: TANH

- Função:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

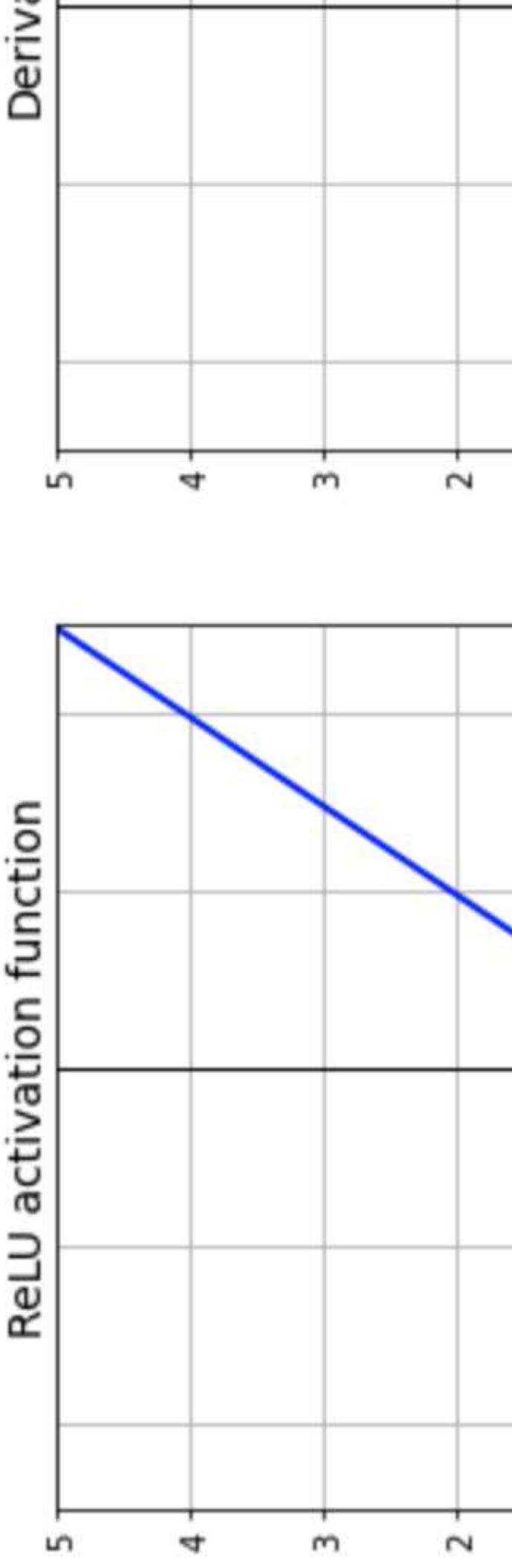
Tanh activation function



ACTIVATION FUNCTIONS: RELU

- Função:

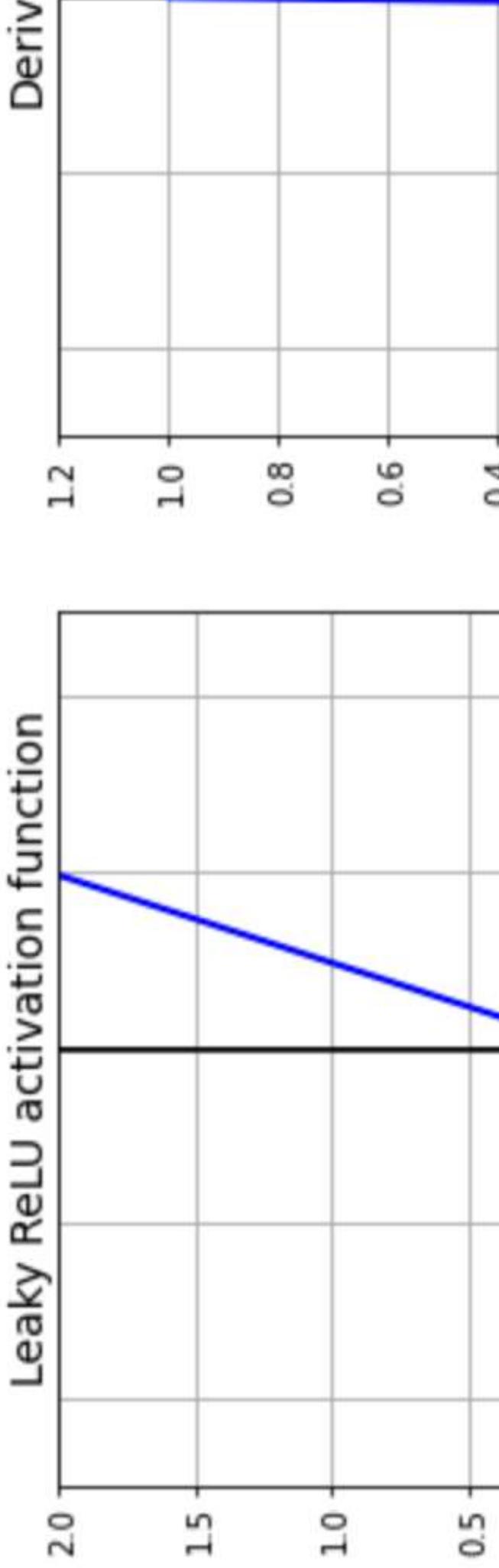
$$f(x) = \max(0, x)$$



ACTIVATION FUNCTIONS: LEAKY RELU

- Função:

$$f(x) = \max(\alpha x, x)$$



SOFTMAX

- Função de ativação sigmoid:

$$a = F(y) = \frac{1}{1 + \exp(-y)}$$

- Função de ativação Softmax

$$a = F(y) = \frac{\exp(y)}{\sum_k \exp(y_k)}$$

REGULARIZATION

AVOIDING OVERFITTING

REGULARIZAÇÃO DOS PESOS (I)

- Pode ser utilizado para evitar *overfitting* dos dados.
 - Estabelece-se um limite para os pesos evitando saturação de sinapses;

$$c = c_0 + \frac{1}{2^m} + \sum_{n=1}^{2^m} w^n$$

REGULARIZAÇÃO DOS PESOS

$$C = \frac{1}{2^m} \sum_j (t_j - s_j)^2 + \frac{\lambda}{2^m}$$

- Atualização dos pesos:

$$\partial C \quad \lambda$$

REGULARIZAÇÃO DOS PESOS

$$C = \frac{1}{2m} \sum_j (t_j - s_j)^2 + \frac{\lambda}{2m}$$

- Por que a regularização reduz o overfitt

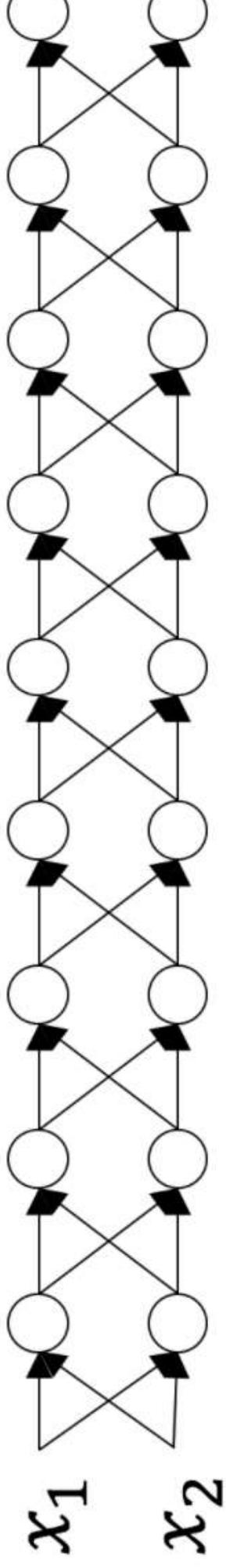
REGULARIZAÇÃO DROPOUT

- Desativar algumas unidades durante o treinamento
 - Evita que um “único” neurônio decore o padrão
 - A “carga” do exemplo é distribuída pelos pesos da rede
- **Parâmetro:** probabilidade de ativação das unidades

OUTRAS FORMAS PARA MELHORA

- Obter mais dados
- Data augmentation
- Parada prematura do treinamento (**treino vs. validação**)

GRADIENT VANISHING / EXPLODING



$$\mathbf{y} = \mathbf{W}^t \mathbf{x}$$

Ver material em:

PROJETO 1 – REDE MLP

- Selecionar dois datasets (não triviais)
- Um dataset para classificação
- Um dataset para regressão
- Separar em treino/validação/teste
- Treinar modelos MLP para os dois problemas

PROJETO 1 – REDE MLP

- Considerar:
 - Diferentes topologias ($>=5$ topologias, variar números de camadas e neurônios por camada)
 - Usar o algoritmo original SGD (não usar algoritmos optimizadores como ADAM)
- Avaliar o impacto do uso do Momentum
 - Avaliar o impacto do uso da regularização (i.e. L2)
- Ilustrar graficamente a evolução do treinamento (treino

QUESTÕES FINAIS

- Como configurar a topologia de uma rede neural Multicamada?
- Qual função de ativação utilizar?
- Qual função de custo?
- Qual algoritmo de treinamento utilizar?