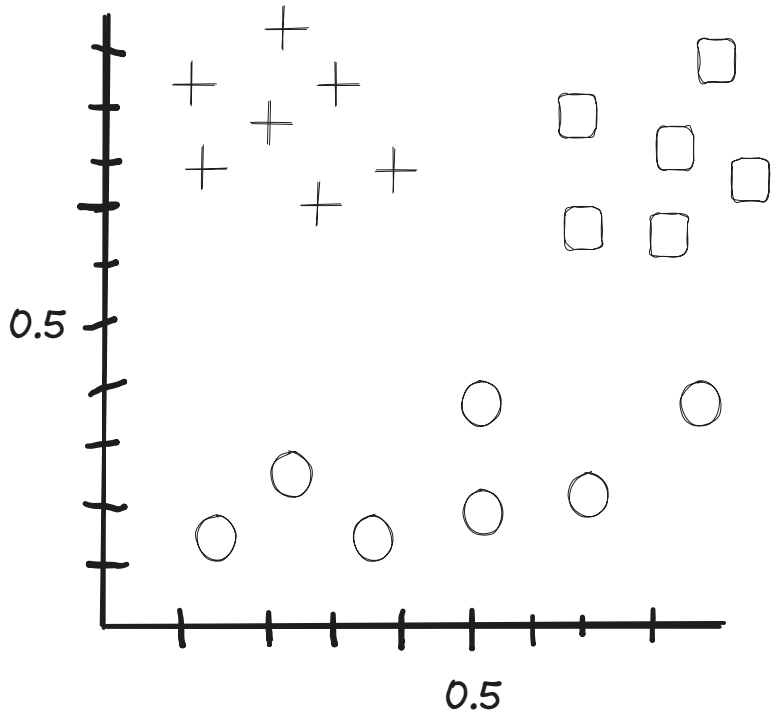
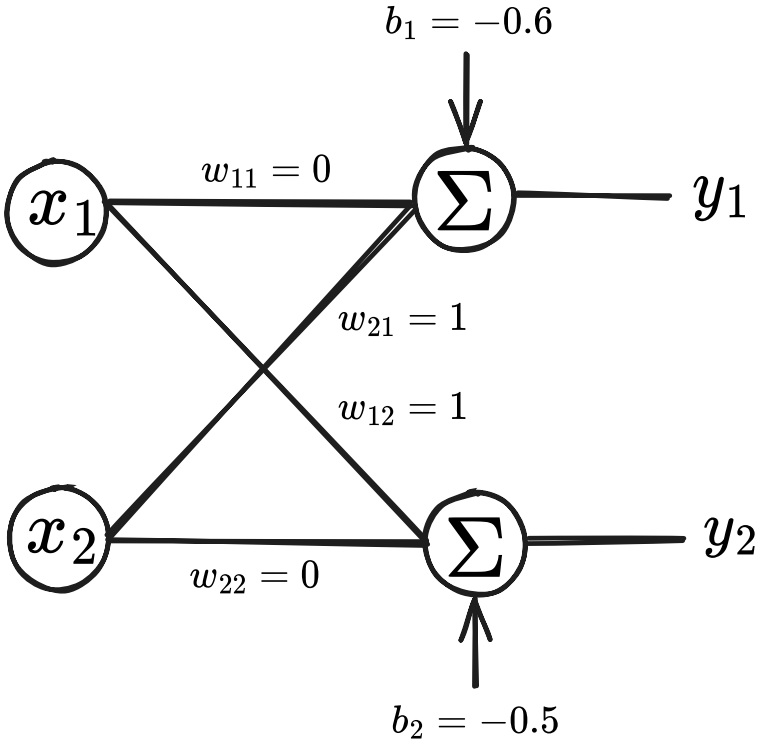
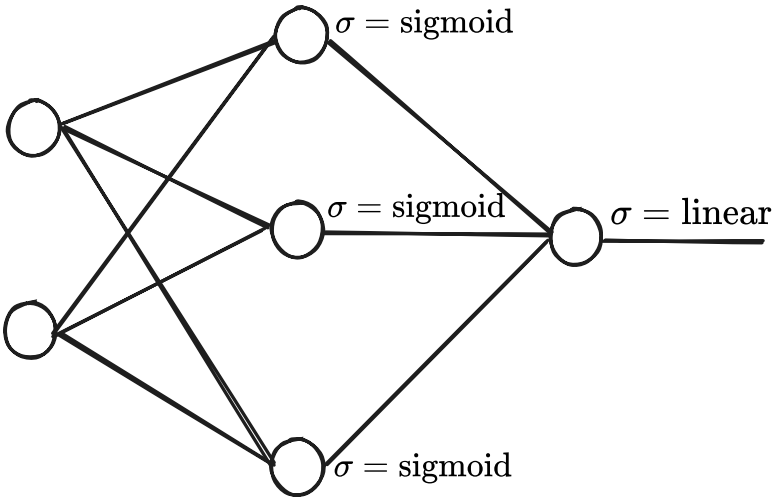
1. Desenha a estrutura de uma rede de perceptrons que classifique corretamente todos os exemplos destas 3 classes:



- Resposta:



2. O que são parâmetros e hiper-parâmetros? Dá exemplos concretos baseado nesta rede.



- Resposta:
  - Parâmetros são os weights e biases da rede, que são atualizados durante o treino.
  - Hiper-parâmetros são valores não treináveis definidos antes do treino que influenciam o processo de aprendizagem. Nesta rede em específico, temos informação sobre os seguintes hiper-parâmetros:
    - Uma output layer com 2 neurónios
    - Uma hidden layer com 3 neurónios
    - Uma output layer com 1 neurónio
    - A função de ativação na hidden layer é sigmoide.
    - A função de ativação na output layer é linear.

3. Porque é que nas redes mais modernas se está a substituir cada vez mais a activation function sigmoidal pela ReLU?

- Resposta:
  - A derivada da sigmoide é muito pequena, está no range $[0, 0.25]$, o que causa o problema do vanishing gradient em redes profundas. A ReLU tem uma derivada constante de 1 para valores positivos, o que ajuda a manter gradientes maiores durante o backpropagation, facilitando o treino de redes profundas. Para além disso, a ReLU é mais computacionalmente eficiente, envolve apenas uma operação de thresholding, ao contrário da sigmoide que envolve funções exponenciais.

4. Explica o que é o overfitting e dá três estratégias para o evitar.

- Resposta:
  - Overfitting acontece quando um modelo aprende demasiado bem o dataset de treino, incluindo o ruído e as variações específicas desses dados, o que resulta num um desempenho mau em dados novos ou não vistos, o modelo não é capaz de generalizar.
  - Estratégias para evitar o overfitting:

- Métodos de regularização como L1, L2 ou dropout.
- Early Stopping
- Data augmentation
- Diminuir a complexidade da rede
- ...

5. Uma camada convolucional de uma CNN recebe como input uma imagem gray-scale 6×6, aplica-se um kernel 3×3 com $\text{stride} = 1$ e $\text{padding} = 0$. Qual a altura e a largura do feature map obtido? Não uses fórmulas para responder a esta pergunta, explica apenas o raciocínio.

- Resposta:
  - A altura e largura do feature map são ambos 4.
  - Como não há padding, o kernel 3×3 vai "caber" na imagem 6×6 de forma a que a sua última posição possível seja na linha 4 e coluna 4 da imagem original (posição onde o kernel cobre as linhas 4, 5, 6 e colunas 4, 5, 6). Como o stride é 1, o kernel move-se um pixel de cada vez. Portanto, começando na posição (1,1) até à posição (4,4), temos um total de 4 posições possíveis tanto na vertical como na horizontal. Assim, o feature map de output é de 4×4.

6. Nas CNNs, onde são extraídas as features da imagem? Onde é que são extraídas as features mais simples e onde é que são extraídas as features mais complexas?

- Resposta:
  - Na componente convolucional. As features mais simples (edges, cantos, etc.) são extraídas nas primeiras camadas, estas são as low-level features. As features mais complexas são extraídas nas camadas mais profundas, uma vez que estas combinam as low-level features para formar representações mais abstratas e complexas, (e.g., combinar edges para criar formas geométricas e depois combinar formas geométricas para criar objetos inteiros), estas são as high-level features.

7. Para que serve a operação de pooling? Qual é o seu papel na extração de features?

- Resposta:
  - O pooling reduz a dimensionalidade dos feature maps, isto diminui o número de parâmetros da rede, o que reduz overfitting e custo computacional.
  - Para além disso, o pooling também ajuda a remover dependências espaciais das imagens, deste modo tornando o modelo mais robusto a transformações (e.g, pequenas translações ou rotações).

8. O que é o algoritmo de otimização ADAM e quais são as suas vantagens em comparação ao Gradient Descent "standard".

- Resposta:
  - O ADAM é um algoritmo que combina as vantagens do mecanismo de momentum e learning rate adaptativo (como implementado no RMSProp). Ele mantém uma moving exponential average dos gradientes passados (momento) e uma moving exponential average dos quadrados dos gradientes (para ajustar o learning rate).
  - Vantagens sobre o Gradient Descent "standard":
    - Convergência mais rápida e estável.
    - Menos sensível à escolha do learning rate.
    - Menos suscetível a mínimos locais.

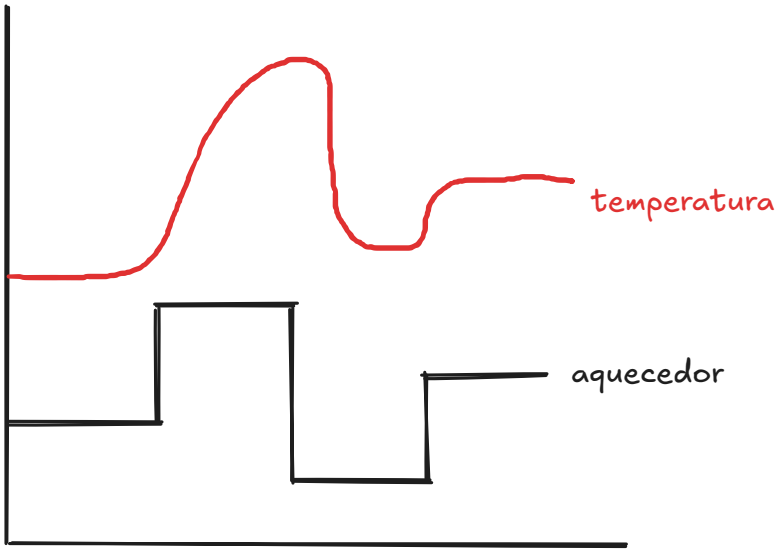
9. Faz uma iteração de backpropagation para atualizar os valores dos weights entre a hidden layer e a output layer da seguinte rede:
   - 2 neurónios de input, 3 neurónios na hidden layer, 1 neurónio na output layer.
   - Learning Rate $\eta = 1$
   - Loss function $\mathcal{L} = (t - y)^2 = e^2$
   - Input $X = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
   - Weights entre a input layer e a hidden layer $W_1 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.2 \\ 0.5 & -0.5 \end{bmatrix}$
   - Weights entre a hidden layer e a output layer $W_2 = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}$
   - Target $t = 0$
   - A activation function é linear ($\sigma(x) = x$) em todas as layers

- Resposta:
  - Forward pass:
    1. Ativações da Hidden Layer: $h = W_1 \cdot X = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.2 \\ 0.5 & -0.5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.4 \\ -0.5 \end{bmatrix}$
    2. Output: $y = W_2 \cdot h = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.1 \\ 0.4 \\ -0.5 \end{bmatrix} = 0.1 - 0.4 - 0.5 = -0.8$
    3. Erro: $e = t - y = 0 - (-0.8) = 0.8$
  - Backward pass:
    1. Sensibilidade da loss em relação ao output:
       $\delta_o = \frac{\partial \mathcal{L}}{\partial y} =$
       $= \frac{\partial \mathcal{L}}{\partial e} \cdot \frac{\partial e}{\partial y} =$
       $= 2e \cdot -1 =$
       $= -2(t - y) =$
       $= 2(y - t) =$

$= 2(-0.8 - 0) =$
$= -1.6$

2. Gradiente da loss em relação aos weights W_2:
$\frac{\partial \mathcal{L}}{\partial W_2} = \delta_o \cdot h^T =$
$= -1.6 \cdot [0.1 \quad 0.4 \quad -0.5] =$
$= [-0.16 \quad -0.64 \quad 0.8]$
3. Atualização dos weights W_2:
$W_2^{new} = W_2 - \eta \cdot \frac{\partial \mathcal{L}}{\partial W_2} =$
$= [1 \quad -1 \quad 1] - 1 \cdot [-0.16 \quad -0.64 \quad 0.8] =$
$= [1.16 \quad -0.36 \quad 0.2]$

10. Explica o que introduziu cada uma das seguintes arquiteturas de CNNs:
    - LeNet-5
        - Resposta:
            - Conceito básico de CNNs que ainda é usado até hoje.
    - AlexNet
        - Resposta:
            - Uso de ReLU como função de ativação.
            - Uso de Dropout para evitar overfitting.
            - Treino em GPUs.
            - Camadas mais profundas
    - VGG
        - Resposta
            - Uso de kernels 3×3 stacked para aumentar a profundidade da rede.
    - ResNet
        - Resposta:
            - Introdução das skip connections para resolver o problema do vanishing gradient em redes muito profundas.
11. Descreve a estrutura de uma RNN de recorrência externa para modelar o seguinte sistema de modo a prever o valor da temperatura com base nos valores anteriores.



    - Resposta:

- Usa-se uma rede feed-forward standard.
- Estrutura: A rede não tem um "estado de memória" interno que se propaga.
- Input: O input da rede é uma "janela" de tempo fixa com valores passados, tanto da temperatura (T) como do aquecedor (A). Por exemplo:
  $X(t) = [A(t), A(t-1), A(t-2), \ldots, T(t), T(t-1), T(t-2) \ldots]$
- Output: A rede mapeia esta janela diretamente para a previsão da temperatura no próximo passo de tempo: $Y(T) = \text{Previsao de } T(t-1)$

12. Igual à pergunta anterior mas com recorrência interna.
    - Resposta:
      - Recorrência interna usa um hidden state (h) para manter memória do histórico.
      - Estrutura (a cada timestep t):
        1. Input $X(t)$: Os valores atuais do aquecedor: $X(t) = [A(t)]$.
        2. Hidden state $h(t)$: A memória da rede. É calculada com base no input atual $X(t)$ E no estado escondido anterior $h(t-1)$.
           $h(t) = f(W_{hh} \cdot h(t-1) + W_{xh} \cdot X(t))$
        3. Output $Y(t)$: A previsão é gerada a partir do estado escondido atual.
           $Y(t) = g(W_{hy} \cdot h(t))$
      4. O output $Y(t)$ é a previsão da temperatura no próximo passo, $T(t+1)$.
13. O que é a função de ativação softmax e porque é que ela é usada no contexto de CNNs? Dá um exemplo concreto da sua utilização.
    - Resposta:
      - A função softmax converte os *scores* brutos (logits) da camada final de uma rede neural numa distribuição de probabilidades, ou seja, um conjunto de valores entre 0 e 1, cuja soma é 1.
      - Exemplo concreto:
        - Problema de classificação.