br.pinterest.com/pin/148829962661029849

Técnicas de Treinamento

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Vetorização

Para facilitar a vetorização dos cálculos, tanto em CPU quanto em GPUs e afins, geralmente:

Cada camada possui uma matriz de pesos na forma de uma matriz *N x M*.

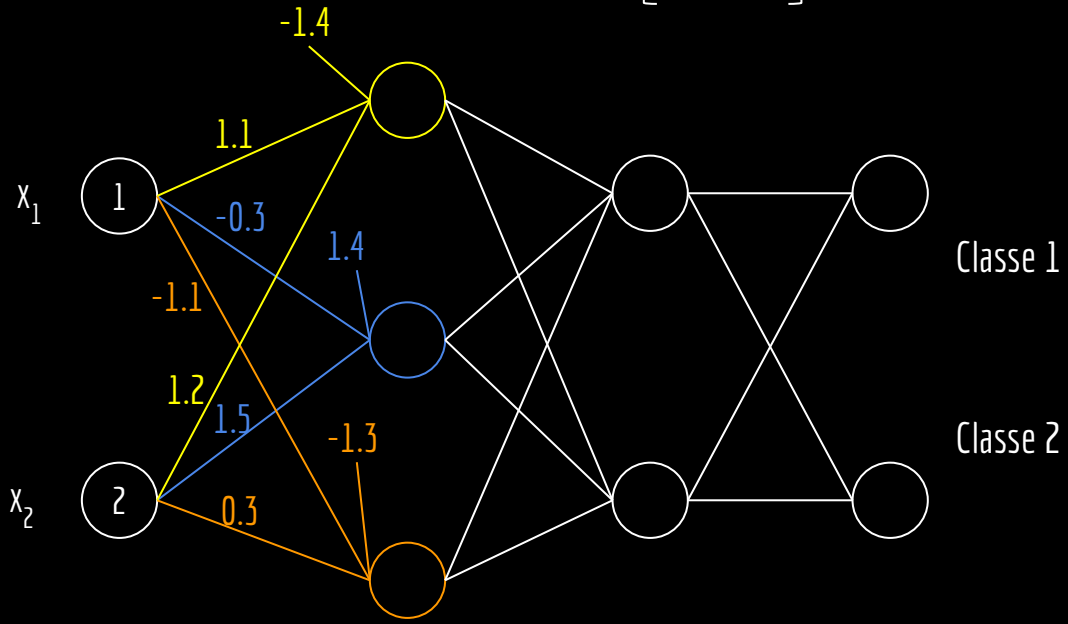
N é o número de neurônios na camada, e *M* é o número de entradas.

Cada camada possui um vetor de bias em um vetor de tamanho *M*.

A entrada da camada é representada em um vetor de tamanho *N*.

Exemplo

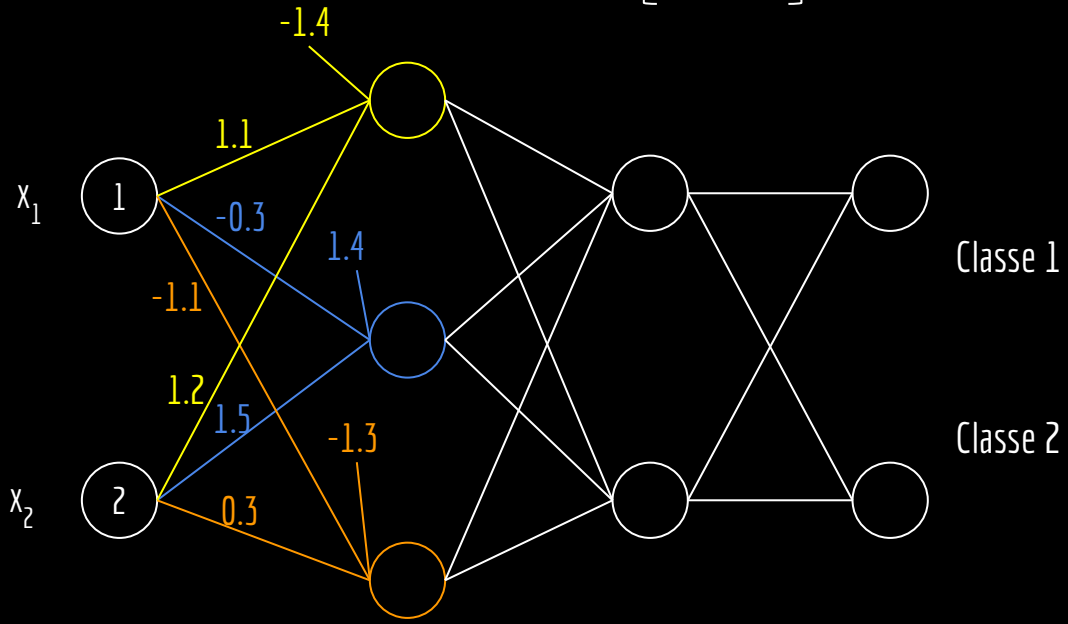
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

Pesos da camada em uma matriz de *N x M*.
N é o número de neurônios na camada, e *M* é o número de entradas.

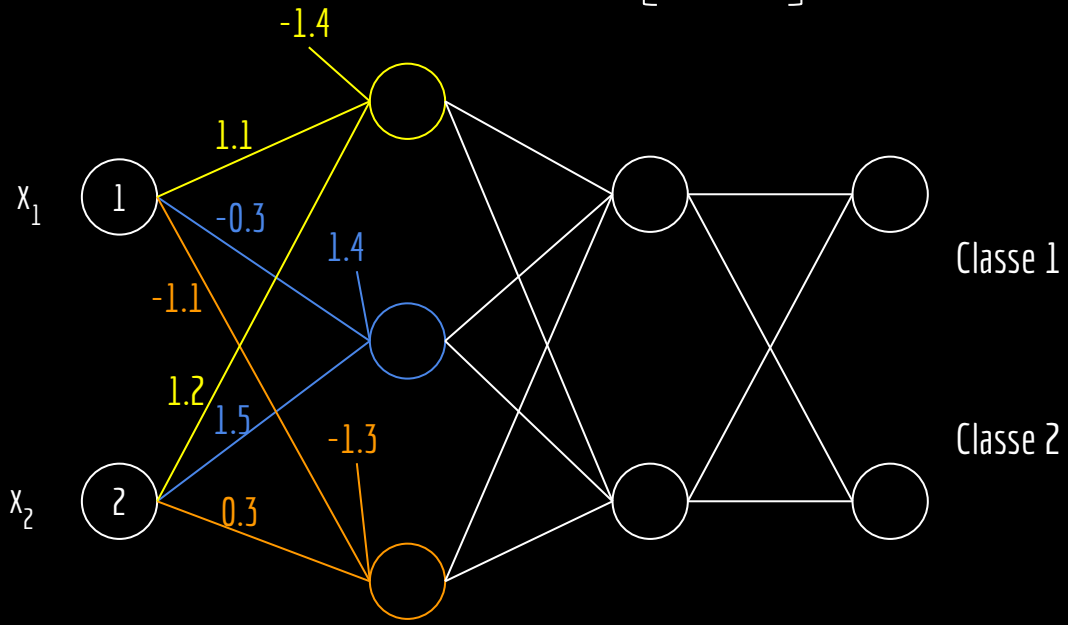
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

Entrada em um vetor de 1 x M

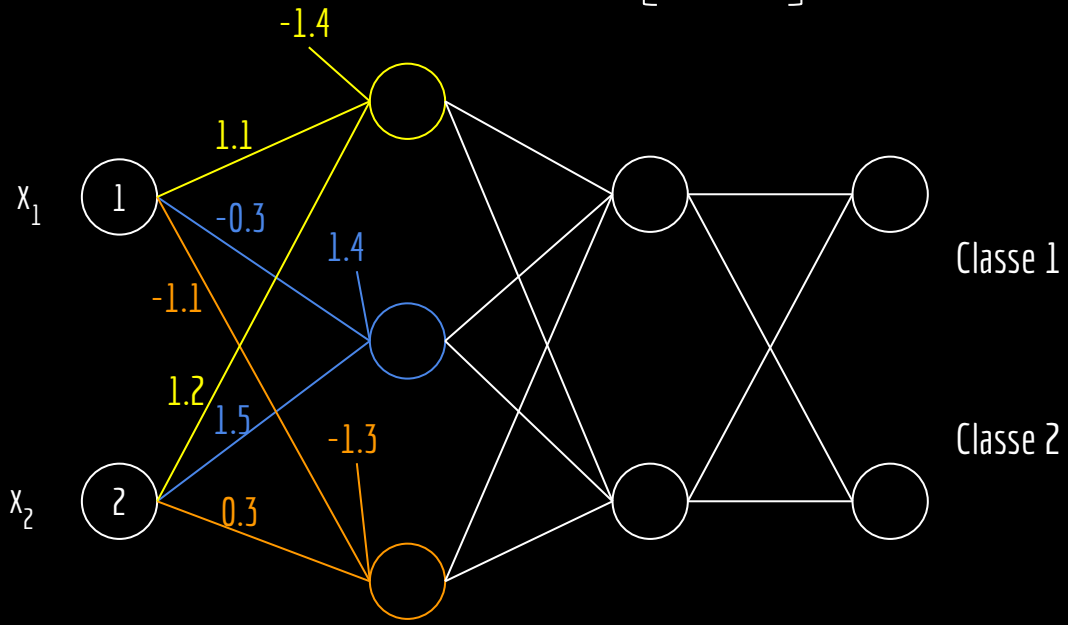
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

Bias em um vetor de 1 X M.

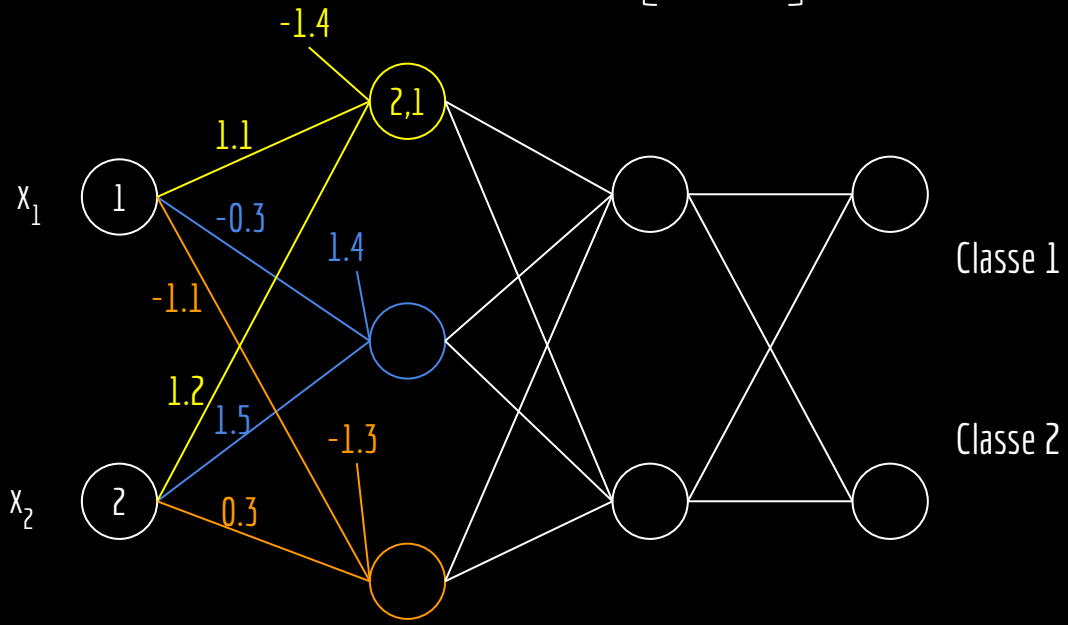
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

$$1.1*1 + 1.2*2 - 1.4$$

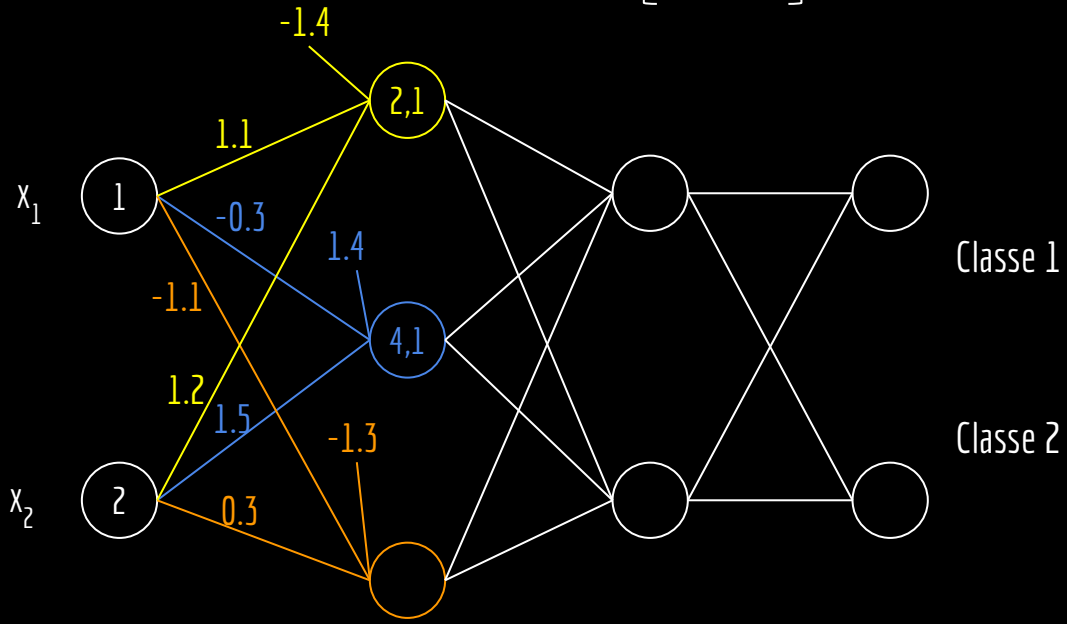
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

$-0.3*1 + 1.5*2 + 1.4$

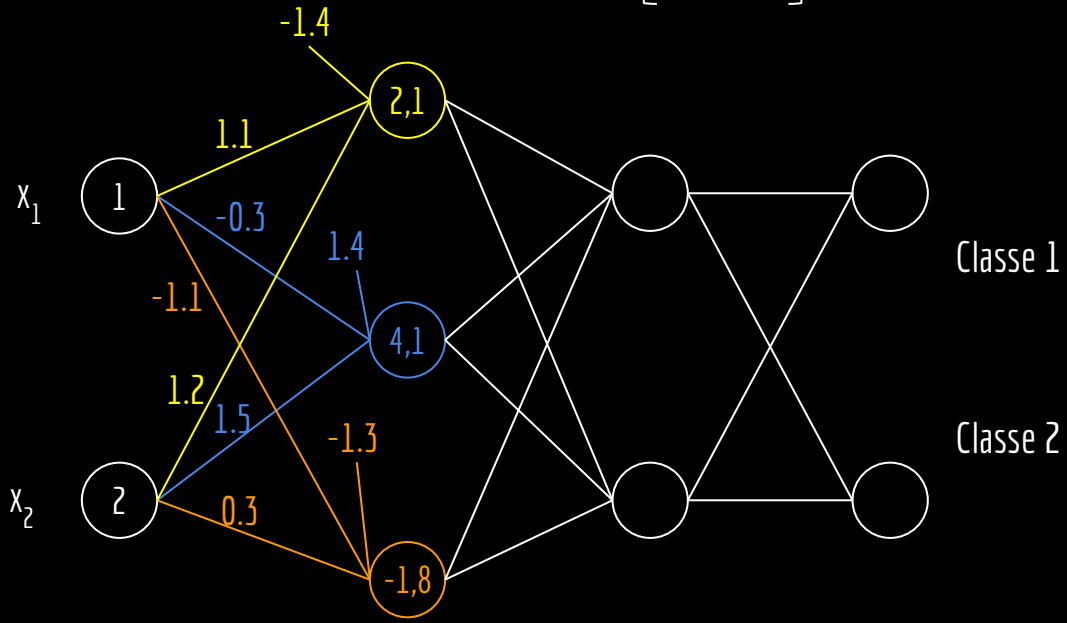
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

$$-1.1*1 + 0.3*2 - 1.3$$

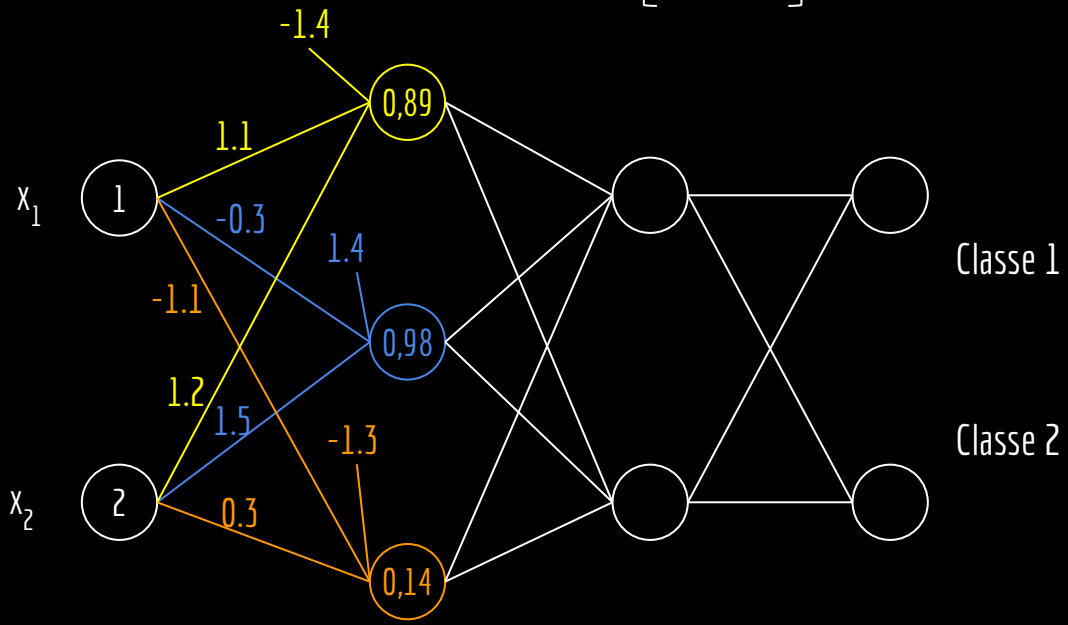
$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

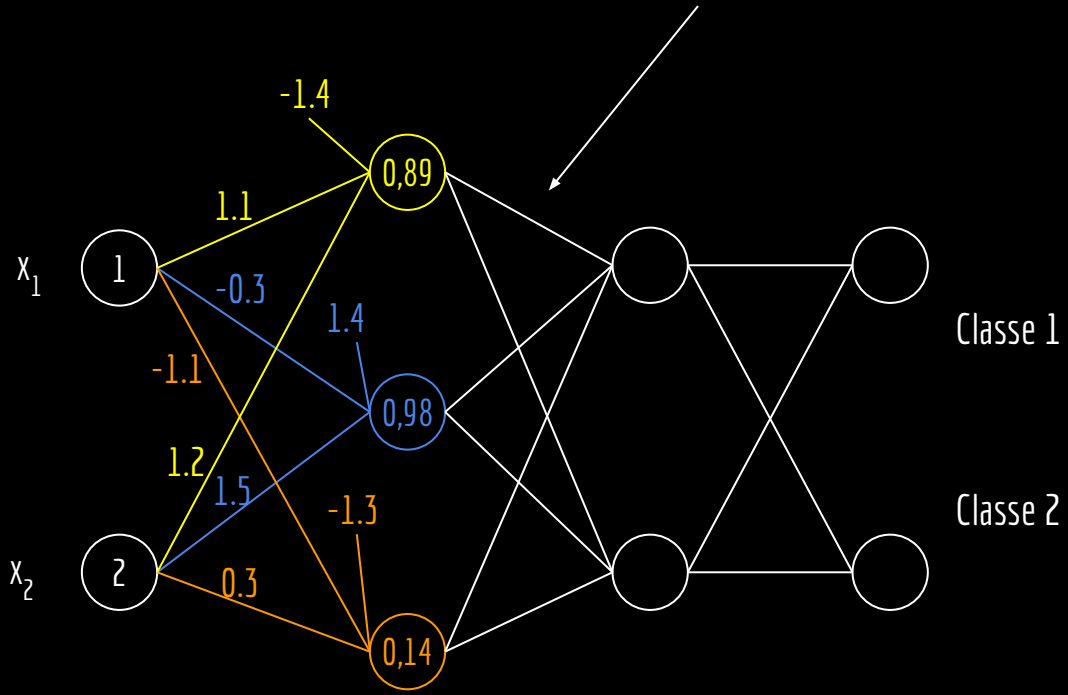
Aplica função de ativação. Exemplo: sigmóide.

$$\begin{bmatrix} 1.1 & 1.2 \\ -0.3 & 1.5 \\ -1.1 & 0.3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 \end{bmatrix} \qquad \begin{bmatrix} -1.4 \\ 1.4 \\ -1.3 \end{bmatrix}$$



Exemplo

Passa o dado para a camada seguinte, que possui sua própria matriz de pesos, e vetor de bias...



Representação

Essa forma se assemelha a representação de um sistema linear no formato **Ax** = **b**.

Representação

Essa forma se assemelha a representação de um sistema linear no formato **Ax** = **b**.

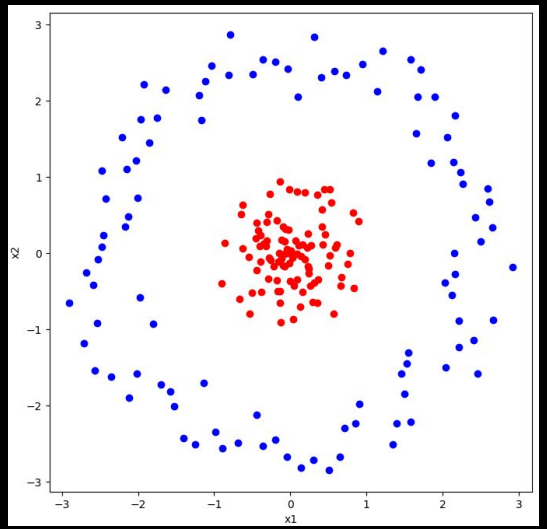
Facilita realizar os cálculos em paralelo e otimiza a organização de memórias cache, por exemplo.

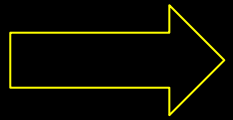
- Processamento SIMD e Vetorial em CPUs.
  - Veja https://prlalmeida.com.br/ci212-2022-01/Aula15ParalelismoSIMD.pdf
- Processamento em GPUs.
  - Veja https://prlalmeida.com.br/ci212-2022-01/Aula18GPUs.pdf

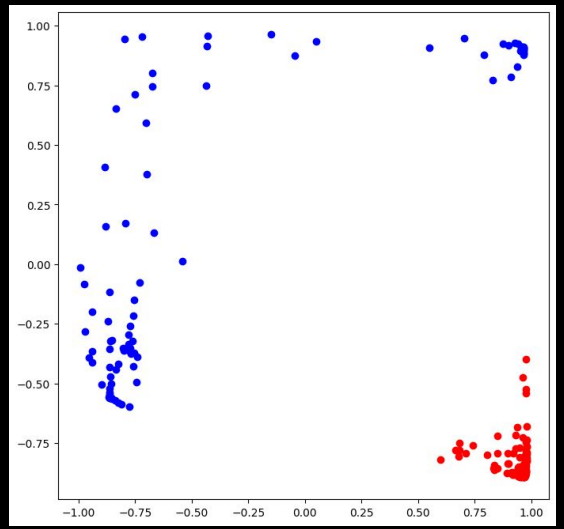
Faça você mesmo

As camadas ocultas de um MLP transformam os dados de entrada de forma que os dados se tornem separáveis.

Execute o programa disponibilizado no Google Colab, e veja como os a saída de uma das camadas internas roi selecionada para gerar uma representação separável dos dados originais.







Minibatch

O método de treinamento usado até agora é chamado de estocástico.

Para cada época

    Para cada instância de treinamento

        Calcular o loss, e realizar o ajuste dos pesos através de retropropagação.

Minibatch

Podemos reduzir o custo computacional através da técnica de Minibatches.

A ideia é usar um pequeno conjunto de dados (minibatch) para calcular um loss médio, antes de realizar a retropropagação do erro.

O batch tem um tamanho *m*, geralmente potência de 2.

Valores comuns são 32, 64, 128 e 256 instâncias.

Para cada época

Pegue as próximas *m* instâncias de treinamento para formar um batch.

Calcular o loss total no batch, e realizar o ajuste dos pesos através de retropropagação.

Minibatch

Uma vantagem imediata do Minibatch é a diminuição de overheads.

Podemos colocar o batch inteiro em uma GPU, por exemplo, e calcular as saídas para as múltiplas instâncias do batch com um baixo overhead de transferência de dados entre memórias.

Aleatorização

Aleatorize os dados de treinamento (mas não os de validação ou teste) a cada iteração de treinamento.

Reduz a chance do modelo ficar preso em mínimos locais.

- Relacionado com problema do esquecimento catastrófico.
  - Veremos mais adiante.

Dessa forma, o algoritmo pode ficar:

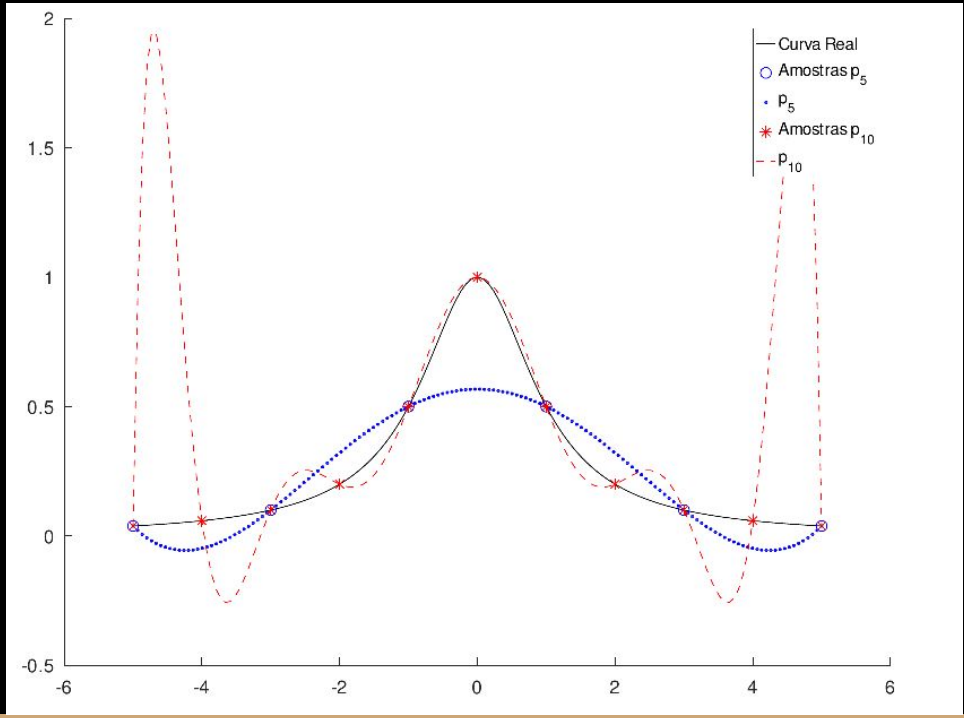
Para cada época

- Randomize os dados de treinamento.
- Pegue as próximas *m* instâncias de treinamento para formar um batch.
  - Calcular o loss total no batch, e realizar o ajuste dos pesos através de retropropagação.

Overfitting

Um modelo que apresenta uma boa acurácia (ou qualquer outra métrica) nos dados de treinamento, mas não no teste, é um modelo que está em ***overfitting*** (sobreajuste).

O modelo “decorou” os dados de treinamento,
e não consegue generalizar para outros dados.



Parâmetros versus Amostras

Uma rede neural com mais parâmetros (pesos e biases) pode evitar problemas de mínimos locais.

Veremos mais nas próximas aulas.

Detalhes no Capítulo 6.4.3 de Duda e Hart., 2012.

Overfitting - Parâmetros versus Amostras

Uma rede neural com mais parâmetros (pesos e biases) pode evitar problemas de mínimos locais.

Veremos mais nas próximas aulas.

Detalhes no Capítulo 6.4.3 de Duda e Hart., 2012.

Mas o excesso de parâmetros pode gerar overfitting.

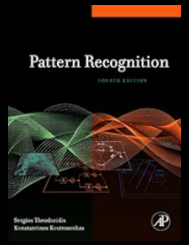
Ideia parecida com problemas gerados pelo excesso de pontos em um polinômio interpolador.

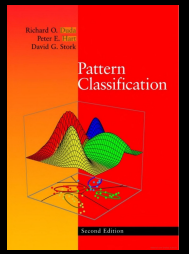
Veja sobre o Fenômeno de Runge: https://prlalmeida.com.br/ci202-2021-01/Aula23.pdf

Overfitting - Parâmetros versus Amostras

Uma regra geral é, dada uma rede com *N* parâmetros ter, pelo menos, *10N* amostras de treinamento **independentes** disponíveis.

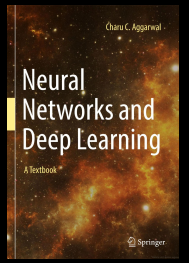
Veja em Theodoridis e Koutroumbas (2010) e em Duda, Hart, Stork (2012).





Alguns autores são menos conservadores, indicando que 3N pode ser o suficiente.

Exemplo: Aggarwal (2018).



Overfitting - Treinamento, teste e validação

Uma técnica comum é separar os **dados de treinamento** em dois **conjuntos disjuntos e independentes**.

Separar o conjunto entre treinamento e validação.

No total, teremos três conjuntos disjuntos e independentes: treinamento, validação e teste.

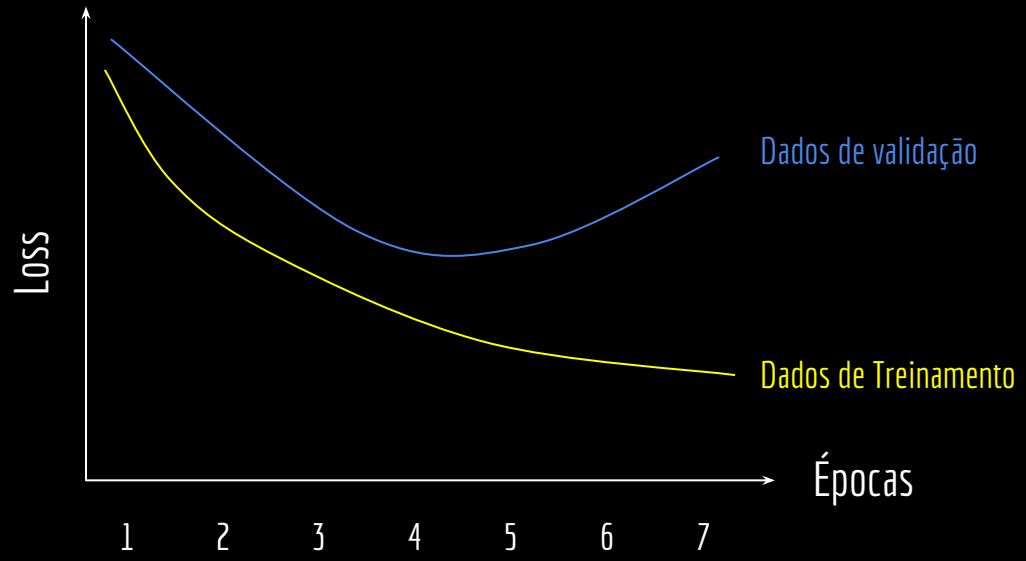
Ideia:

A cada época, usar apenas o conjunto de treinamento para treinar.

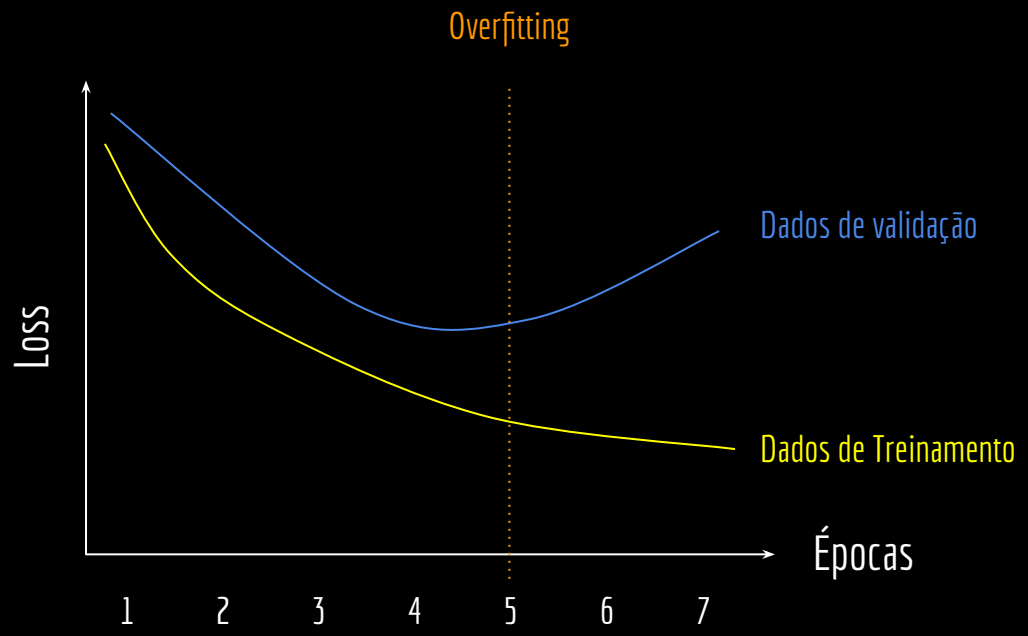
No final de cada época, avaliar (ex.: usando o loss) a qualidade do modelo nos dados de validação.

Escolher a época com o **menor loss nos dados de validação**.

Overfitting - Treinamento, teste e validação



Overfitting - Treinamento, teste e validação



Faça você mesmo

Considere o exemplo disponibilizado no Google Collab.

Verifique e entenda quais técnicas discutidas na aula já estão presentes no exemplo, e quais faltam.

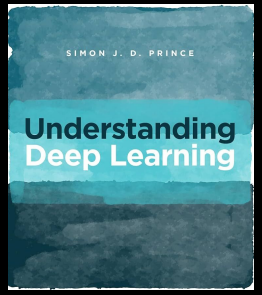
Inclua todas as técnicas discutidas para treinar o modelo.

Exercícios

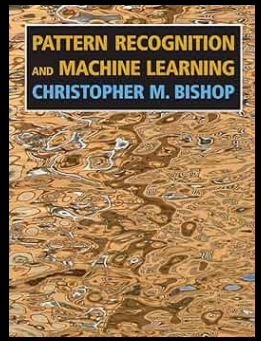
1. Leia sobre a técnica de treinamento em Batch. Como ela difere da técnica estocástica e de minibatchs? Quais vantagens e limitações?

Referências

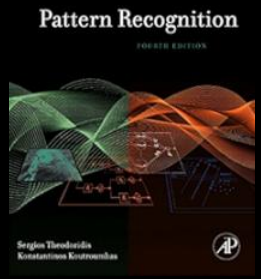
Prince, S. J. Understanding Deep Learning. 2023.



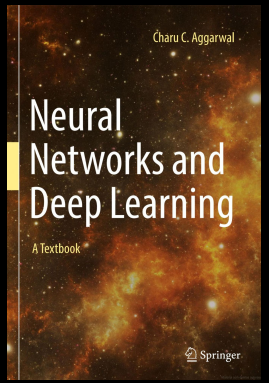
Bishop, C. M. Pattern Recognition and Machine Learning. 2006.



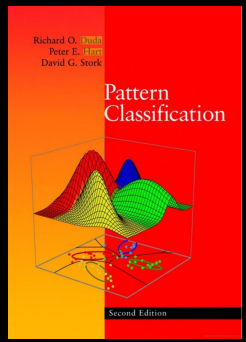
Theodoridis, S., Koutroumbas, K. Pattern Recognition & Matlab Intro. 2010.



Aggarwal. Neural Networks and Deep Learning. 2018.



Duda, R. O., Hart, P. E., Stork, D. G. Pattern Classification. 2012.



Licença