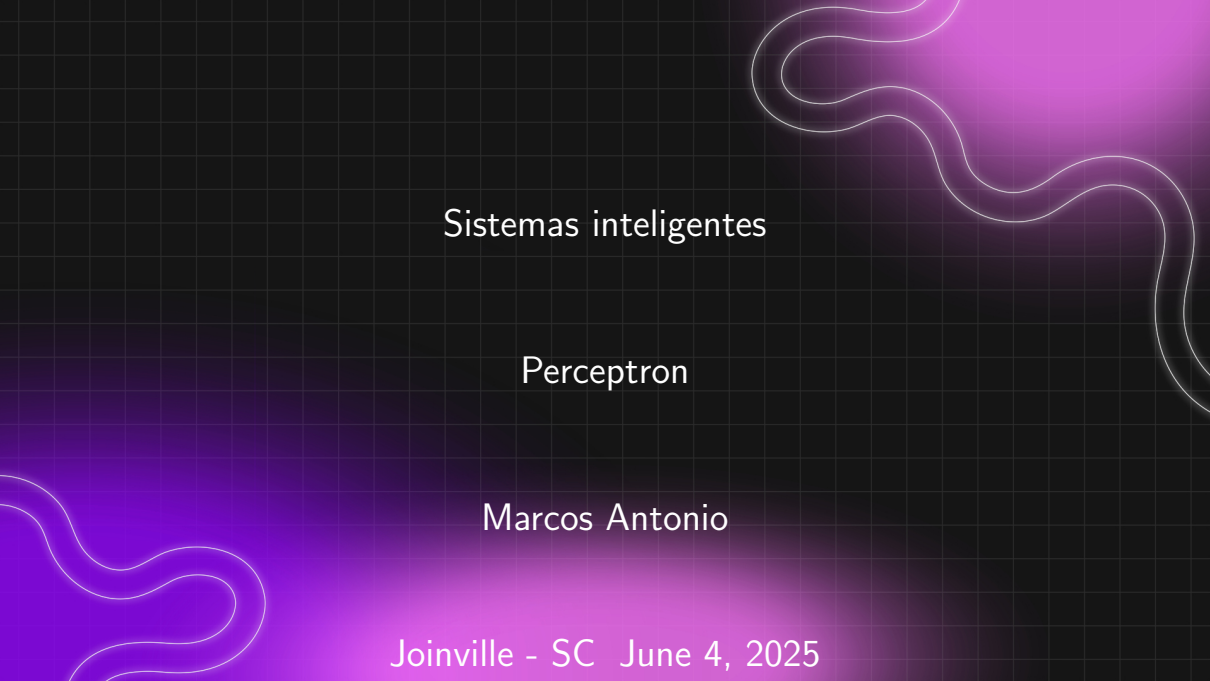


Sistemas inteligentes

Marcos Antonio

Joinville - SC, June 4, 2025



Sistemas inteligentes

Perceptron

Marcos Antonio

Joinville - SC June 4, 2025

Outline

- Pré-processamento

 - Importação Bibliotecas

- Implementação do Perceptron em C

 - Inicialização do perceptron

- Plotagem dos resultados do treinamento

 - plotando os dados

 - Plotagem dos dados Lineares

 - Plotagem dos dados não lineares

Importação de Bibliotecas

```
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
```

```
# Carregar a base Iris
iris = datasets.load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
```

Criação dos Dataframes

```
df = pd.DataFrame(X, columns=feature_names)
df["label"] = y
```

```
# Linearmente separavel: classes 0 e 1 (setosa e versicolor)
linear_df = df[df["label"].isin([0, 1])].reset_index(drop=True)
```

```
# Nao linearmente separavel: classes 1 e 2 (versicolor e virginica)
nonlinear_df = df[df["label"].isin([1, 2])].reset_index(drop=True)
```

Salvando os Folds

```
def save_kfolds(df, prefix, n_splits=5):  
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)  
    X = df[feature_names].values  
    y = df["label"].values  
  
    for fold_num, (train_idx, test_idx) in enumerate(kf.split(X), start=1):  
        train_data = df.iloc[train_idx]  
        test_data = df.iloc[test_idx]  
  
        train_path = f"build/data/{prefix}_fold{fold_num}_train.csv"  
        test_path = f"build/data/{prefix}_fold{fold_num}_test.csv"  
  
        train_data.to_csv(train_path, index=False)  
        test_data.to_csv(test_path, index=False)
```

Outline

- Pré-processamento

 - Importação Bibliotecas

- Implementação do Perceptron em C

 - Inicialização do perceptron

- Plotagem dos resultados do treinamento

 - plotando os dados

 - Plotagem dos dados Lineares

 - Plotagem dos dados não lineares

Inicialização do perceptron

```
Perceptron* create_perceptron(int num_inputs, float learning_rate) {  
    Perceptron *p = (Perceptron*) malloc(sizeof(Perceptron));  
    p->num_inputs = num_inputs;  
    p->learning_rate = learning_rate;  
    p->bias = 1.0;  
  
    p->weights = (float*) malloc((num_inputs + 1) * sizeof(float));  
    for (int i = 0; i <= num_inputs; i++) {  
        p->weights[i] = ((float) rand() / RAND_MAX) * 2.0f - 1.0f;  
    }  
  
    return p;  
}
```


Função de ativação

```
int activate(Perceptron *p, float *inputs) {  
    float sum = 0.0;  
  
    for (int i = 0; i < p->num_inputs; i++) {  
        sum += inputs[i] * p->weights[i];  
    }  
  
    sum += p->bias * p->weights[p->num_inputs];  
  
    return (sum > 0) ? 1 : 0;  
}
```

Treinamento

```
void train(Perceptron *p, float *inputs, int desired_output) {  
    int guess = activate(p, inputs);  
    int error = desired_output - guess;  
  
    if (error != 0) {  
        for (int i = 0; i < p->num_inputs; i++) {  
            p->weights[i] += p->learning_rate * error * inputs[i];  
        }  
        p->weights[p->num_inputs] += p->learning_rate * error * p->bias;  
    }  
}
```

Avaliação do Modelo

```
void evaluate(Perceptron *p, const char *test_path) {  
    FILE *test_file = fopen(test_path, "r");  
    if (!test_file) {  
        perror("Erro ao abrir base de teste");  
        return;  
    }  
  
    char line[1024];  
    int correct = 0, total = 0;  
  
    fgets(line, sizeof(line), test_file);  
  
    while (fgets(line, sizeof(line), test_file)) {  
        float inputs[4];  
        int label;
```

Avaliação do Modelo

```
        if (sscanf(line, "%f,%f,%f,%f,%d",
                    &inputs[0], &inputs[1], &inputs[2], &inputs[3], &label) == 5) {
            int prediction = activate(p, inputs);
            if (prediction == label)
                correct++;
            total++;
        }
    }

    fclose(test_file);

    float accuracy = 100.0f * correct / total;
    printf("Acuracia no teste: %.2f%% (%d corretos de %d)\n", accuracy, correct, total);
}
```

Desalocando memória

```
void destroy_perceptron(Perceptron *p) {  
    free(p->weights);  
    free(p);  
}
```

Outline

- Pré-processamento

 - Importação Bibliotecas

- Implementação do Perceptron em C

 - Inicialização do perceptron

- Plotagem dos resultados do treinamento

 - plotando os dados

 - Plotagem dos dados Lineares

 - Plotagem dos dados não lineares

Importação dos Módulos

```
import pandas as pd
import matplotlib.pyplot as plt
import os
```

Plotando os dados

```
plt.figure(figsize=(10, 6))

for fold in range(1, 6):
    log_path = f"build/data/train_log_fold{fold}.csv"
    df = pd.read_csv(log_path)

    plt.plot(df['epoch'], df['accuracy'], label=f'Fold {fold}')
```


Plotando os dados

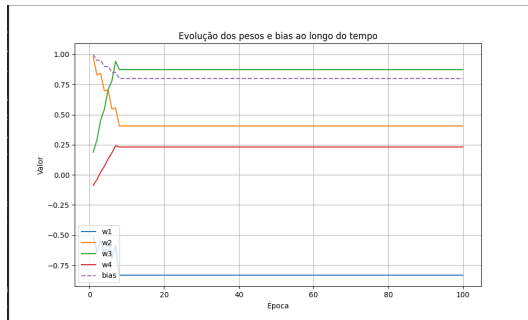


Figure: Evolução do Bias e pesos ao longo do tempo

Plotando os dados

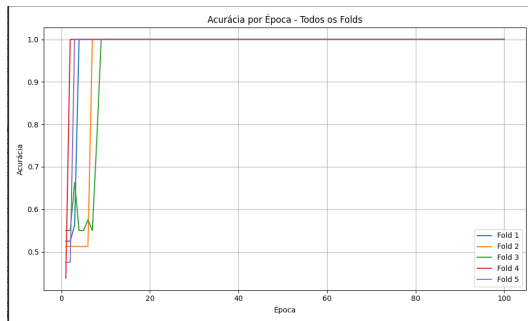


Figure: Acurácia por Épocas

Plotando os dados

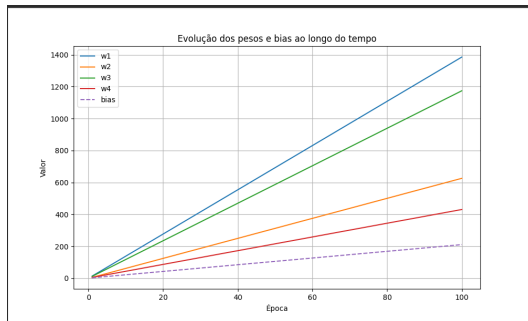


Figure: Evolução do Bias e pesos ao longo do tempo

Plotando os dados

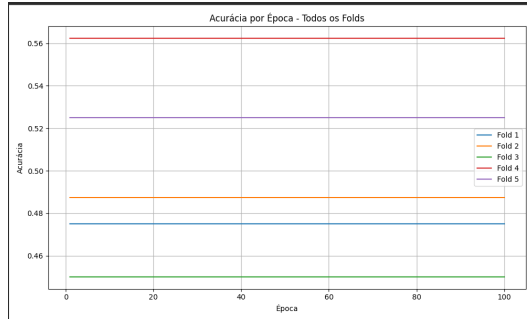


Figure: Acurácia por Épocas

Plotando os dados

```
plt.xlabel('Epoca')
plt.ylabel('Acuracia')
plt.title('Acuracia por Epoca - Todos os Folds')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig('build/acuracia_treinamento_folds.png')
plt.show()
```