Universidade Federal de Pelotas

Conceitos de Linguagem de Programação

Docente: Gerson Cavalheiros

Discentes: Gerson Menezes e Eduardo Timm

Documentação

1. Apresentação da Aplicação

O projeto consiste em uma aplicação de aprendizado de máquina que implementa uma rede neural simples para realizar previsões meteorológicas. O objetivo principal do modelo é, a partir de um conjunto de 8 características climáticas (como temperatura, umidade, vento, etc.), prever duas variáveis de saída: a **sensação térmica** e a **probabilidade de chuva**.

A aplicação é dividida em duas partes principais:

1. Um "motor" de processamento de alto desempenho, responsável pelo treinamento e execução da rede neural.
2. Uma interface de orquestração e visualização, que compila, executa e monitora o treinamento em tempo real, exibindo a evolução da taxa de erro (Loss/MSE) em um gráfico dinâmico.

2. Divisão de Tarefas por Linguagem

A arquitetura do projeto explora as principais vantagens de duas linguagens de programação distintas, C e Python, atribuindo a cada uma a tarefa onde ela se destaca.

- **Linguagem C (`main.c`)** A linguagem C foi escolhida para implementar as partes da aplicação que exigem maior desempenho computacional. Suas responsabilidades incluem:
    - A implementação completa da lógica da rede neural (propagação direta e retropropagação).
    - O carregamento e processamento do conjunto de dados de treinamento a partir de um arquivo `.csv`.
    - A execução do laço de treinamento, que realiza cálculos matemáticos intensivos repetidamente.
    - O salvamento e carregamento do modelo treinado em um arquivo binário (`model.bin`).
    - A disponibilização de um modo interativo via terminal para realizar previsões com o modelo já treinado.

- **Linguagem Python (`run_and_plot.py`)** Python atua como o orquestrador de alto nível e a interface de visualização do projeto. Suas responsabilidades são:
    - Gerenciar o ciclo de vida da aplicação, compilando o código C (caso o executável não exista) e iniciando sua execução.
    - Atuar como o processo "pai", executando o programa em C como um subprocesso filho.
    - Capturar a saída de texto do programa em C em tempo real durante o treinamento.
    - Interpretar (fazer o *parsing*) das informações de progresso enviadas pelo C, como a época atual e o erro quadrático médio (MSE).
    - Utilizar a biblioteca `matplotlib` para plotar um gráfico dinâmico que exibe a curva de aprendizado do modelo em tempo real.

3. Método de Interface entre Linguagens

A comunicação entre o script Python e o programa em C é realizada através de uma técnica de **Comunicação entre Processos (IPC)**, utilizando os fluxos de entrada e saída padrão (`standard I/O`). O método não envolve a vinculação direta de bibliotecas ou o compartilhamento de memória, mas sim a troca de mensagens de texto através de um "cano" (pipe).

O processo ocorre da seguinte forma:

1. **Execução do Subprocesso**: O script Python inicia o programa C compilado como um subprocesso, utilizando a biblioteca `subprocess`. Crucialmente, ele redireciona a saída padrão (`stdout`) do processo C para um pipe (`stdout=subprocess.PIPE`), permitindo que o Python leia tudo o que o C imprimiria no terminal.
2. **Protocolo de Comunicação Textual**: Foi definido um protocolo simples baseado em texto. Durante o treinamento, a cada época, o programa C formata e imprime uma string específica contendo o número da época e o valor do erro, prefixada com "LOSS:": `printf("LOSS:%d,%.9f\n", e+1, mse);`. Ao final, ele envia um sinal de conclusão: `printf("TRAINING_DONE\n");`.
3. **Garantia de Transmissão em Tempo Real**: Para que os dados sejam enviados imediatamente, sem aguardar o preenchimento de um buffer, o comando `fflush(stdout);` é chamado no código C logo após cada `printf`. Isso força a "descarga" do buffer de saída, garantindo que o Python receba os dados em tempo real.
4. **Leitura Não-Bloqueante**: No lado do Python, uma `Thread` separada é criada com a única função de ler a saída do pipe do subprocesso C. Isso é fundamental para que a interface gráfica (o gráfico) não congele enquanto o script espera por novos dados. A thread de leitura popula as listas de dados de forma segura, usando um `Lock` para evitar condições de corrida.
5. **Controle de Comportamento**: O Python também controla o modo de execução do C através de argumentos de linha de comando. Ao iniciar o subprocesso, ele passa a flag `"--no-interactive"`, instruindo o programa C a encerrar sua execução após o treinamento, em vez de entrar no modo de previsão interativo.