Relatório Atividade M1

Leonardo Santos

Objetivo

O objetivo desta atividade foi estender a metodologia da Atividade M1 para o caso em que os sinais de entrada e saída são complexos. Além disso, utilizou-se um conjunto de dados de extração (treinamento) e validação para avaliar a generalização do modelo.

Diferentemente da Atividade M1, onde o vetor de erro era retornado diretamente, nesta atividade o retorno da função foi definido como o módulo do vetor de erro, garantindo que o otimizador trabalhe apenas com valores reais.

Metodologia

A atividade foi desenvolvida em duas etapas:

1. Função de erro
   - Argumentos: números reais representando as partes real e imaginária de cada coeficiente.
   - Internamente, os coeficientes complexos são reconstruídos como:

$$c_k = Re(c_k) + jIm(c_k)$$

   - Para cada amostra, calcula-se a saída estimada do Polinômio de Memória (MP) e obtém-se o erro:

$$e(n) = y_{\text{out}(n)} - y_{\text{est}(n)}$$

   - O retorno da função é o módulo do erro, ou seja:

$$r(n) = |e(n)|$$

   .

2. Otimização não linear
   - Em Python, utilizou-se `scipy.optimize.least_squares`.
   - Em Octave, utilizou-se `fminunc`.
   - O vetor de parâmetros otimizado contém `2C` valores reais (partes real e imaginária).
   - Os coeficientes complexos finais são reconstituídos após a otimização.
   - O desempenho do modelo foi avaliado pelo MSE tanto no conjunto de extração quanto no de validação.

Implementação

Jupyter Notebook (Python)

- Dados carregados do arquivo `.mat` (`in_extraction`, `out_extraction`, `in_validation`, `out_validation`).
- Implementação da função `erro_mp_complex`, retornando `np.abs(erro)`.
- Otimização com `least_squares` e reconstrução dos coeficientes complexos.
- Cálculo do MSE na validação:

```
mse_val = np.mean(erro_val**2)
```

Octave

- Script `.m` criado para leitura do mesmo `.mat`.
- Implementada a função `erro_mp_complex`, retornando `abs(y_out - y_est)`.
- Definição da função objetivo como a média do quadrado dos módulos do erro.

- Utilização do `fminunc` com vetor inicial de zeros para as partes real e imaginária.
- Reconstrução dos coeficientes complexos após a otimização e avaliação no conjunto de validação.

Resultados

Os coeficientes obtidos foram muito próximos nos dois ambientes:

- Python (SciPy):

```
  1.2392+0.1927j   0.0133-0.0076j   0.0021+0.0022j
 -0.8041-0.2932j  -0.0205+0.0048j  -0.0004-0.0055j
  0.4477+0.1131j   0.0075+0.0005j  -0.0013+0.0035j
```

- Octave (fminunc):

```
  1.2780+0.1832j   0.0137-0.0073j   0.0021+0.0021j
 -0.8801-0.2747j  -0.0210+0.0041j  -0.0004-0.0053j
  0.4865+0.1036j   0.0078+0.0010j  -0.0013+0.0034j
```

As diferenças observadas são pequenas (ordem de $10^{\{-3\}}$ a $10^{\{-2\}}$) e atribuídas a:
- Algoritmos de otimização diferentes (`least_squares` vs `fminunc`),
- Critérios de parada distintos,
- Pequenas diferenças numéricas no cálculo.

Apesar disso, ambos os modelos apresentam desempenho equivalente no cálculo do MSE, confirmando a consistência da abordagem.

Conclusão

A Atividade M2 mostrou que é possível aplicar a metodologia de otimização não linear também para o caso de sinais complexos, garantindo a consistência dos resultados em diferentes ambientes (Python e Octave). O uso do módulo do erro como retorno da função foi essencial para adequar o problema ao espaço real requerido pelos otimizadores. Os coeficientes encontrados em ambos os ambientes foram equivalentes dentro da precisão numérica, e o MSE de validação confirmou a qualidade do modelo obtido.