TP555 - AI/ML

Lista de Exercícios #3

Regressão para Modelos Não-Lineares e Regressão Polinomial

1. Exercício sobre regressão não-linear. Dada a seguinte função não-linear

$$y = a_0 x^{a_1}$$

Como você poderia linearizar essa função (lembre-se que a nova função é linear em relação aos pesos e não em relação ao atributo)? Qual é a função hipótese que você poderia usar para encontrar os pesos da função geradora? Imagine que você só tem acesso aos vetores x e y. Supondo que a_0 = 1.4 e a_1 = 3.0 e x é igual a um vetor com M = 1000 amostras retiradas de uma distribuição uniforme no intervalo [0.0, 10.0), faça o seguinte:

   a. Plote o gráfico da função não-linear (i.e., x versus y).
   b. Linearize a função não-linear. Apresente todas as transformações feitas na equação para encontrar a função linearizada.
   c. Plote o gráfico da função linearizada, i.e., x versus y.
   d. Utilizando a função hipótese que lineariza a função-geradora,
      i. Plote a superfície de erro.
      ii. Encontre os valores ótimos dos pesos da função hipótese utilizando a equação normal.
      iii. Encontre os pesos da função hipótese utilizando o algoritmo do gradiente descendente em batelada com ***critério de parada*** definido como sendo quando a ***diferença absoluta*** entre o erro da iteração atual e a anterior caia abaixo de 0.00001 ou que o número máximo de épocas tenha sido atingido. Faça o número máximo de épocas igual a 1000. (**OBS**.: não se esqueça de encontrar o melhor valor para o passo de aprendizagem).
      iv. Plote a superfície de contorno mostrando o histórico de atualização dos pesos e seus valores ótimos.
      v. Plote o gráfico de erro versus número de épocas.
   e. Quais os valores dos pesos encontrados pelo o algoritmo do gradiente descendente em batelada?

2. Suponha que você esteja usando regressão polinomial. Você plota as **curvas de aprendizado** e percebe que há uma grande diferença entre o erro de treinamento e o erro de validação. O que está acontecendo? Quais são as três maneiras de resolver isso?

   OBS.: **Curvas de aprendizado**: são gráficos mostrando o desempenho do modelo no conjunto de treinamento e no conjunto de validação em função do tamanho do conjunto de treinamento (ou da iteração do treinamento).

3. Exercício de comparação entre as regressões Ridge e LASSO. Dada a seguinte versão ruidosa da função objetivo y_noisy = 2 + x + 0.5*x^2 + n, onde x é um vetor coluna com M = 100 elementos retirados de uma distribuição aleatória uniformemente distribuída variando entre -3 e 3 e n é o vetor ruído com M elementos retirados de uma distribuição aleatória Gaussiana com média 0 e variância unitária. Utilize um polinômio de ordem 90, padronização de atributos (ou seja, remoção da média e divisão pelo desvio padrão) e regressão LASSO (utilize a biblioteca SciKit-Learn) com λ variando entre 1e-10 e 1 (utilize *np.linspace*(10**-10, 1, 1000)). Utilizando a função "*train_test_split*", divida os exemplos em um conjunto de treinamento e outro de validação com proporção 70% e 30%, respectivamente. Faça o seguinte
    a. Plote um gráfico mostrando a função objetivo e sua versão ruidosa.
    b. Crie um loop para testar cada um dos 1000 valores de λ. Para cada novo valor de λ, treine o modelo, execute a predição e calcule os erros de treinamento e validação.
    c. Para cada iteração do loop, armazene os valores do erro de treinamento e validação em um vetor.
    d. Para cada iteração do loop, verifique se o valor do erro de validação atual é menor do que o erro de validação mínimo. Se sim, armazene o valor de λ e o modelo utilizado para aquela iteração. (**Dica**: inicialize a variável contendo o erro de validação mínimo como: *minimum_val_error = float("inf")*).
    e. Plote um gráfico mostrando os erros de treinamento e validação versus todos os valores de λ, ou seja, os 1000 valores de λ.
    f. Baseado no menor valor do erro de validação, qual é o valor ótimo para λ?
    g. Dado que você armazenou o modelo que obteve o menor erro de validação, utilize-o para criar um gráfico que mostre a função hipótese (ou seja, o mapeamento do atributos de entrada, x, nos valores de saída, y, através do modelo treinado) e a função objetivo e sua versão ruidosa.
    h. Imprima os valores do pesos obtidos durante o treinamento do modelo que obteve o menor erro de validação (**Dica**: use o atributo ***named_steps*** da classe Pipeline para acessar os objetos que compõem o pipeline: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html.
    i. Repita os passos anteriores para a regressão de Ridge.
    j. O que você percebe com relação aos pesos obtidos com as regressões Ridge e LASSO?

    (**Dica**: Não se esqueça que os parâmetros do escalonamento de atributos, ou seja, média e devio padrão, são encontrados utilizando-se o conjunto de treinamento. Os parâmetros encontrados são utilizados para escalonar o conjunto de validação).
    (**Dica**: Na instanciação da classe Lasso, configure a tolerância para 1, i.e., tol=1.)
    (**Dica**: A documentação do regressor LASSO pode ser acessada através deste link: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html?highlight=lasso#sklearn.linear_model.Lasso)
    (**Dica**: utilize a classe clone para criar uma cópia do modelo que atingiu erro de validação menor do que o valor mínimo atual.

https://scikit-learn.org/stable/modules/generated/sklearn.base.clone.html)

4. Exercício sobre Early stopping. Dada a seguinte versão ruidosa da função objetivo y_noisy = 2 + x + 0.5*x^2 + x^3 + n, onde x é um vetor coluna com M = 100 elementos retirados de uma distribuição aleatória uniformemente distribuída variando entre -3 e 3 e n é o vetor ruído com M elementos retirados de uma distribuição aleatória Gaussiana com média 0 e variância unitária. Utilize um polinômio de ordem 30 como função hipótese, padronização de atributos (ou seja, remoção da média e divisão pelo desvio padrão) e o algoritmo do gradiente descendente em batelada. Utilizando a função "***train_test_split***", divida os exemplos em um conjunto de treinamento e outro de validação com proporção 70% e 30%, respectivamente. Faça o seguinte
    a. Plote um gráfico mostrando a função objetivo e sua versão ruidosa.
    b. Encontre, manualmente, o melhor valor para o passo de aprendizagem.
    c. Execute o treinamento por 1000 épocas.
    d. Para cada época, armazene em um vetor os valores do erro de treinamento e validação.
    e. Para cada época, verifique se o valor do erro de validação atual é menor do que o erro de validação mínimo. Se sim, armazene o modelo utilizado para aquela época, ou seja, os valores dos pesos, e o valor do erro de validação para aquela época. (**Dica**: inicialize a variável contendo o erro de validação mínimo como: *minimum_val_error = float("inf")*).
    f. Plote um gráfico mostrando os erros de treinamento e validação versus o número de épocas.
    g. Dado que você armazenou o modelo que obteve o menor erro de validação, utilize-o para criar um gráfico que mostre a função hipótese (ou seja, o mapeamento do atributos de entrada, x, nos valores de saída, y, através do modelo treinado) e a função objetivo e sua versão ruidosa.
5. Exercício que utiliza validação cruzada. Usando o arquivo ***covid19.csv***, onde onde a primeira coluna são os valores de x (i.e., atributo) representando o número de dias desde o primeiro caso confirmado de COVID-19 e a segunda coluna são os valores de y (i.e., objetivo ou rótulo), representando o número de casos de COVID-19 ativos. Leia o conteúdo do arquivo, ou seja, os vetores x e y, com os seguintes comandos:

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

df = pd.read_csv('./covid19.csv', header=None)

x = df[0].to_numpy()
y = df[1].to_numpy()

x = x.reshape(len(x),1)
y = y.reshape(len(y),1)

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

```
plt.plot(x, y, 'b.')
```

Em seguida, faça o seguinte

a. Plote os valores do arquivo, i.e., um gráfico mostrando o dias desde o primeiro caso versus o número de casos ativos.
b. Encontre uma aproximação polinomial que represente bem os dados do arquivo. Para encontrar a melhor aproximação, utilize os seguintes métodos: validação cruzada holdout (com 80% do conjunto original para treinamento e 20% para validação), validação cruzada k-fold (com k=10 folds), validação cruzada leave-p-out (com p=1) e curvas de aprendizado. Analise polinômios com ordem variando de 1 até 12.
c. Em seguida, de posse da melhor ordem de polinômio que aproxima o modelo gerador, treine o modelo com todos os dados do arquivo csv. Utilize padronização de atributos com a classe StandardScaler da biblioteca SciKit-Learn.
d. De posse do modelo treinado, crie um vetor x variando de 1 a 70 com incrementos de 1 em 1, i.e., número de dias desde o primeiro caso registrado até 70 dias depois, e faça a predição do número de casos ativos até 70 dias após o primeiro caso registrado.
e. Sabendo que o número total de leitos de UTI no Brasil é de 40600 (aqui vamos supor que nenhum leito está ocupado no momento), preveja em quantos dias desde o início do primeiro caso registrado no Brasil (26-02-2020) o número de leitos total seria atingido.

6. Neste exercício você vai utilizar o arquivo ***reg_poli.csv*** onde a primeira coluna são os valores de x (atributo) e a segunda de y (objetivo ou rótulo). O arquivo contém a versão ruidosa da função original, ou seja o modelo gerador ao qual ruído é adicionado. Após, leia o conteúdo do arquivo, ou seja, os vetores x e y, com os seguintes comandos:

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

df = pd.read_csv('./reg_poli.csv', header=None)

x = df[0].to_numpy()
y = df[1].to_numpy()

x = x.reshape(len(x), 1)
y = y.reshape(len(y), 1)

fig = plt.figure(figsize=(10,10))
plt.plot(x, y, 'b.')
```

Em seguida

a. Apresente o gráfico de x versus y, mostrando os **pontos** amostrados do modelo gerador.

b. Encontre uma aproximação polinomial que represente bem os dados do arquivo. Para encontrar a melhor aproximação, utilize os seguintes métodos: validação cruzada holdout (com 70% do conjunto original para treinamento e 30% para validação), validação cruzada k-fold (com k=10 folds), validação cruzada leave-p-out (com p=1) e curvas de aprendizado. Analise polinômios com ordem variando de 1 até 12.
c. Em seguida, de posse da melhor ordem de polinômio que aproxima os dados do arquivo csv, treine o modelo com todos os dados do arquivo csv. Utilize padronização de atributos com a classe StandardScaler da biblioteca SciKit-Learn.
d. Plote um gráfico que mostre os pontos ruidosos do arquivo ***reg_poli.csv*** e os valores encontrados com o modelo para os valores de x vindos do arquivo csv, ou seja, use o modelo para "prever" os valores de y com os valores de x vindos do arquivo.

7. Neste exercício você irá utilizar regressão LASSO para encontrar a ordem de uma função hipótese polinomial que melhor se ajuste aos dados de treinamento fornecidos. Utilize os arquivos training_lasso.csv e validation_lasso.csv para treinar e validar seu modelo, respectivamente.
    a. Plote as amostras de ambos os arquivos.
    b. Utilize uma função polinomial de ordem igual a 40.
    c. Treine um modelo de regressão linear (sem regularização) com os dados de treinamento (utilize padronização para escalonar os atributos)
    d. De posse do modelo treinado, faça a predição com os dados de treinamento e validação e em seguida plote um gráfico comparando os valores obtidos com a predição e os valores dos arquivos de treinamento e validação.
        i. Observando o gráfico, o que você pode concluir que está ocorrendo?
    e. Treine um modelo de regressão linear com regularização LASSO (use a classe Lasso da biblioteca SciKit-Learn) com os dados de treinamento (utilize padronização para escalonar os atributos). Crie um laço de repetição para que você treine modelos de regressão LASSO com diferentes valores de λ, variando de 1e-10 até 100. Armazene o valor do erro quadrático médio (MSE) para cada valor de λ.
    f. Plote um gráfico mostrando o MSE versus o valor de λ.
    g. Encontre o valor de λ que resulta no menor erro de validação, treine um novo modelo de regressão LASSO com este valor, realize a predição com os dados de treinamento e validação e em seguida plote um gráfico comparando os resultados obtidos com a predição e os valores dos arquivos de treinamento e validação.
        i. Observando o gráfico, você ainda observa o mesmo fenômeno que ocorria antes, quando utilizamos apenas a regressão linear sem regularização?
    h. De posse do modelo de regressão LASSO treinado com o valor de λ que resulta no menor erro de validação, crie um laço de repetição para variar o número de

pesos considerados para a predição, varie de 2 (polinômio de ordem 1) até 41 (polinômio de ordem 40). Armazene o MSE de validação para cada uma das iterações.

i. Plote um gráfico mostrando a variação do MSE de validação quando o número de pesos considerados varia de 2 a 41.

j. De posse dos valores de MSE do item anterior, encontre a quantidade de pesos que resulta no menor MSE. Usando apenas a quantidade de pesos que resulta no menor MSE, realize a predição com os dados de treinamento e validação. Em seguida, plote um gráfico comparando os resultados obtidos com a predição e os valores dos arquivos de treinamento e validação. (**Dica**: sempre padronize os atributos)

   i. Observando o gráfico, pode-se dizer que o resultado é similar ao obtido com o modelo de regressão LASSO de ordem 40 da letra (g)?

   ii. Compare com outras quantidades de pesos, por exemplo, 2, 3, e 5, por exemplo.

8. Exercício sobre early stopping. Utilizando a seguinte função geradora ruidosa

$$y = a0 + a1*x + w,$$

onde a0 = 1.0, a1 = 2.0, x é um vetor coluna com Mx1 elementos retirados de uma distribuição uniformemente distribuída entre o intervalo [-1.0, 1.0), w é o ruído adicionado à função geradora e é um vetor coluna com Mx1 elementos retirados de uma distribuição Gaussiana com média zero e variância igual a 0.09 e M = 50. Gere agora um conjunto de validação, utilizando Mtest = 50 valores **linearmente espaçados entre [-1.0, 1.0)** ao invés de valores retirados de uma distribuição uniforme, conforme usamos para gerar os dados de treinamento. De posse dos 2 conjuntos (treinamento e validação), faça o seguinte:

   a. Plote em um gráfico os dados originais e os valores dos conjuntos de treinamento e validação.

   b. Treine um modelo de Regressão Polinomial de ordem igual a 40 (use a classe ***LinearRegression*** da biblioteca SciKit-Learn ou implemente a forma fechada),

   c. Plote em um gráfico os dados originais, os valores dos conjuntos de treinamento e validação e os valores de predição obtidos com o modelo treinado quando se utiliza os conjuntos de treinamento e validação como entrada do modelo.

   d. Após analisar o gráfico da letra (c), o que você conclui? O que está ocorrendo?

   e. Agora, treine um modelo ***polinomial*** de ordem igual a 40 utilizando o ***gradiente descendente em batelada com early stopping***. Configure o número máximo de épocas para 100000 e não utilize critério de parada, deixe o algoritmo treinar as 100000 épocas e sempre armazene os pesos que resultaram no menor erro de validação. (**OBS**.: Não se esqueça de encontrar o melhor valor para o passo de aprendizagem).

   f. Após o treinamento, plote um gráfico mostrando o número de épocas versus os erros de treinamento e validação (MSE).

   g. Após observar o gráfico, o que você conclui? O que está ocorrendo durante o treinamento iterativo do modelo?

h. Plote em um gráfico os dados originais, os valores dos conjuntos de treinamento e validação e os valores de predição obtidos com o modelo treinado iterativamente quando se utiliza os conjuntos de treinamento e validação como entrada do modelo.
i. Após analisar o gráfico da letra (h), o que você conclui? O que está ocorrendo? Qual a diferença entre este gráfico e o obtido na letra (c).
j. Utilizando o último valor dos pesos, ou seja, os pesos obtidos após a época de número 100000, plote em um gráfico os dados originais, os valores dos conjuntos de treinamento e validação e os valores de predição obtidos com o modelo treinado iterativamente quando se utiliza os conjuntos de treinamento e validação como entrada do modelo.
k. Após analisar o gráfico da letra (j), o que você conclui? O que está ocorrendo? Qual a diferença entre este gráfico e o obtido na letra (h).

9. Proponha um exercício sobre Ridge regression. Crie um enunciado para este exercício e apresente a sua solução.
10. Neste exercício veremos que a regressão LASSO pode ser usada para seleção de atributos. Para se ter uma ideia da importância dos atributos, você deve usar o estimador **LassoCV**. Os pesos com o valores absolutos mais altos são considerados os mais importantes e, portanto, indicam os atributos mais importantes. Os pesos são acessados através do atributo **coef_** da classe **LassoCV**. Use o conjunto de dados do arquivo **lasso_feature_selection.csv** e a classe **LassoCV** para selecionar os atributos mais importantes deste conjunto de dados. O conjunto de dados do arquivo possui 100 atributos, mas muitos deles não são importantes e podem ser descartados. O ***conjunto de dados já foi padronizado***. Use o trecho de código abaixo para ler os valores dos atributos e dos valores esperados, i.e., X e y, respectivamente. Em seguida, faça o seguinte

```
df = pd.read_csv('./lasso_feature_selection.csv', header=None)
D = df.to_numpy()

X = D[:,0:D.shape[1]-1]
y = D[:,D.shape[1]-1]
```

1. Use valores de **alpha** entre 0.001 e 0.1. Use a função **linspace** da biblioteca **numpy**, por exemplo, para gerar valores neste intervalo. Lembre-se que para a biblioteca SciKit-Learn **alpha** é o coeficiente de regularização e não o passo de aprendizagem.
2. Faça o número de **folds**, ou seja, o parâmetro **cv**, igual a 5 e treine o modelo.
3. Imprima o valor do MSE para este modelo com o conjunto de dados original.
4. Crie uma figura mostrando o valor **absoluto** dos pesos encontrados. Use a função **bar** da biblioteca **matplotlib**.
5. Crie um subconjunto X apenas com atributos cujos pesos tenham valor **absoluto** maior do que ou igual a 0.1.
6. Qual a quantidade de atributos deste subconjunto?

7. Use novamente a classe **LassoCV** com o subconjunto de dados criado e valores de **alpha** variando **logaritmicamente** de 1e-20 a 1. Use a função **logspace** da biblioteca **numpy**, por exemplo, para gerar valores neste intervalo.
8. Imprima o valor do MSE para este modelo com o subconjunto de dados.
9. Compare os resultados com o conjunto original e o subconjunto, o que pode ser concluído?

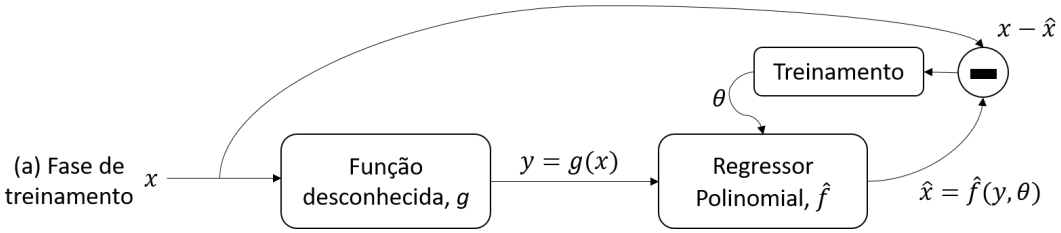
11. Em telecomunicações, podemos encontrar em determinados problemas, variáveis aleatórias que não possuem formas conhecidas para suas funções densidade de probabilidade (FDP). Nestes casos, podemos recorrer à aproximação de funções como uma forma de encontrar uma função que aproxime os valores observados desta variável aleatória. Portanto, neste exercício, iremos usar regressão polinomial para encontrar uma função hipótese polinomial que aproxime o melhor possível os dados observados de uma variável aleatória. As observações da variável aleatória com FDP desconhecida podem ser geradas com o trecho de código abaixo.

```
# Número de amostras da variável aleatória.
M = 10000000

# Número de termos do somatório.
N = 4

# Gera uma das variáveis aleatórias.
f = 1 + (1/(np.sqrt(2)))*(np.random.randn(N,M) + 1j*np.random.randn(N,M))
f = np.abs(f)

# Gera outra variável aleatória.
g = 1 + (1/(np.sqrt(2)))*(np.random.randn(N,M) + 1j*np.random.randn(N,M))
g = np.abs(g)

# Gera variável aleatória com FDP desconhecida.
h = np.sum(f*g, axis=0)

# Número de divisões do histograma.
bins = 300
y, X, p = plt.hist(h, bins=bins, density=True)
plt.title('$y = \sum_{n=1}^{N} |f_n||g_n|$')
plt.show()

X = X[0:len(X)-1].reshape(bins,1)
```

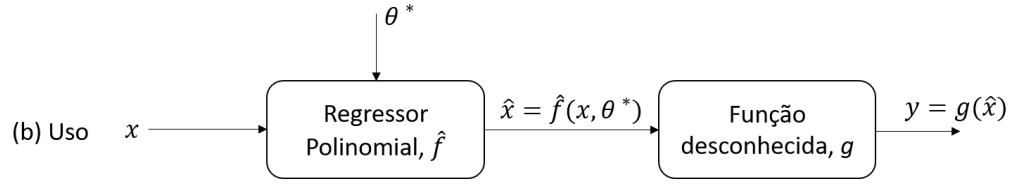
Use as observações geradas para encontrar a ordem ótima de uma função polinomial que aproxime a FDP desta variável. Use as técnicas de validação cruzada (i) Holdout e (ii) K-Fold para determinar a ordem do polinômio. Em seguida, gere uma figura comparando a aproximação encontrada com os valores originais.

12. Neste exercício, iremos inverter uma função não-linear e usar a função invertida para linearizá-la. Em sistemas de telecomunicações, distorção não-linear pode ocorrer entre transmissor e receptor, por exemplo, devido a amplificadores não-lineares no transmissor. Esta distorção pode ser representada por uma função desconhecida *g* que recebe um sinal *x* como entrada e produz uma saída distorcida *y* = *g*(*x*). A maneira convencional de desfazer a distorção é identificar um modelo parametrizado apropriado, então estimar seus parâmetros através de medições e, finalmente, criar uma função inversa com base nas estimativas. Essa abordagem está sujeita à propagação de erros entre as três etapas. Uma abordagem alternativa é treinar um modelo de regressão para inverter diretamente a função, sem exigir modelagem explícita ou estimativa de parâmetros. O aprendizado de máquina pode fornecer melhores resultados do que a abordagem convencional para a linearização de funções.

O procedimento geral para treinar um modelo de aprendizado de máquina para inversão de funções é ilustrado na figura abaixo.



Uma função desconhecida *g* com entrada *x* é invertida usando um modelo $\widehat{f}$ treinando-o para obter $\widehat{f}\left(g(x), \theta^{*}\right) \approx x$, conforme mostrado em (a). O procedimento de treinamento atualizará iterativamente os pesos θ para reduzir gradualmente os erros de aproximação até que convirja para um valor ótimo, θ^{*}.



O modelo treinado em (b) pode ser usado para linearizar a função desconhecida, *g,* sem ter que modelá-la explicitamente e estimar os parâmetros do modelo. Em (b), o que acontece é uma pré-distorção do sinal de entrada *x* o qual quando passado através da função desconhecida *g*, produz em sua saída um sinal linear.

Para realizar o treinamento, precisamos gerar um número N de valores x_n^{train} e passá-los através da função desconhecida, *g*, para medir

$$y_n^{train} = g\left(x_n^{train}\right), \text{ para } n = 1,\ldots, N.$$

Então, y_n^{train} é usado como entrada para o modelo de aprendizado de máquina, enquanto x_n^{train} é a saída desejada.

Exercício: dada a seguinte função *y* = *g*(*x*):

$$y = \frac{\alpha x}{\left(1+\left(\frac{|x|}{v_{sat}}\right)^{2p}\right)^{\frac{1}{2p}}},$$

onde $\alpha = 4, p = 2$e $v_{sat} = 1$. Use um ***modelo de regressão polinomial*** para inverter a função *g*. Para isso, faça o seguinte

1. Plote o gráfico de *x* versus *y*.
2. Treine um modelo de regressão polinomial que inverta a função *g*.
    a. Use validação cruzada, por exemplo, holdout, k-fold, leave-p-out, para encontrar a ordem ótima do polinômio.
    b. Apresente as curvas de erro quadrático médio e desvio padrão, se for o caso, em função das ordens utilizadas na validação cruzada.
3. Após o treinamento, usando a ordem ótima do polinômio, apresente:
    a. Um gráfico comparando o sinal x_n^{train}com o sinal x_n^{pred}(**Dica**: use o próprio sinal de treinamento para obter a predição).
4. O modelo treinado consegue inverter, ou seja, linearizar, a função desconhecida, *g*?
5. Como mostrado em (b), use o modelo treinado para pré-distorcer o sinal de entrada e, em seguida, aplique a saída do modelo à entrada da função *g*.
    a. Plote um gráfico comparando
        i. O sinal de entrada *x* e a saída da função *g*, ou seja, o sinal de saída original e distorcido.
        ii. O sinal de entrada *x* e o sinal pré-distorcido $\widehat{x}$.
        iii. O sinal pré-distorcido $\widehat{x}$ e a saída da função *g* quando sua entrada é $\widehat{x}$.
        iv. O sinal de entrada *x* e o sinal de saída do sistema de pré-distorção, *y*, ou seja, a saída da função *g* quando sua entrada é $\widehat{x}$.
    b. O que ocorre com o sinal de saída, *y*?