第一次上机作业

注意事项

1. 上机作业所需上传的文件，应打包并命名为"学号-姓名-上机#序号"，如"18000xxxxx-张三-上机1"，并在截止时间前上传到教学网。截止时间后仍开放提交，但会酌情扣分。
2. 上机作业为半自动评分。上传的压缩包，会根据压缩包名称自动分类，并提取压缩包名称中的有效信息。请尽可能保持压缩包命名符合规范。压缩包名中的短横线'-'可替换为下划线'_'、加号'+'，但不能是空格' '、长横线'—'。
3. 上机作业中的代码会自动测试。要求代码中读入的外部数据文件存放在代码文件上一级目录下的data文件夹下，生成的图表、数据文件等应在代码文件所在目录下。

```
root
  ├─workdir (Compress this directory and upload it)
  |     ├─code.py (Required)
  |     ├─data.csv (Required)
  |     ├─chart.png (Optional)
  |     └─report.pdf (Required)
  └─data (DO NOT upload this directory)
        └─dataset.xlsx
```

4. 题目中要求生成的图表和计算的数据均列在报告中。

1. 在用机器学习算法训练一个模型时，如果选取的特征数较多但样本量有限，直接在训练集上训练模型，模型的泛化性往往较差，出现过拟合。为了提升模型的泛化性，降低模型的复杂度，可以引入正则化，使模型兼顾好的预测能力和泛化性。

（1）以`train.dat`中数据为训练集，以`test.dat`中数据为测试集。数据文件的第一列为x，第二列为y。用岭回归拟合数据，以$x, x^2, x^3, \cdots$为基（拟合至4阶，需拟合常数项），**绘制合适的图表**，表示在训练集和测试集上的均方根误差RMS及拟合系数（$C_0, C_1, C_2, C_3, \cdots$）随正则化参数λ的变化关系（横轴用对数坐标或用$\ln \lambda$作为标度）

多项式岭回归示例代码：

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 9/28/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''


# Imports
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error

N = 1000   # Sample size
np.random.seed(20200928)   # Set random seed
```

```
X = np.arange(0., 10., .1)   # Generate X data series in range [0.0, 10.0) with
step 0.1
Y = np.sin(X) + np.random.normal(0.0, 0.5, X.size)   # Generate Y data series by
adding gaussian noise with μ = 0.0 and σ = 0.5 to sin(X)

# Generate polynomial features for fitting
# With degree=4 and include_bias=False, train_X will be transformed to 4-feature
data series (without constant term)
# Got train_X like this:
# [[X0, X0^2, X0^3, X0^4],
#  [X1, X1^2, X1^3, X1^4],
#  ...
#  [Xn-1, Xn-1^2, Xn-1^3, Xn-1^4]]
poly_features = PolynomialFeatures(degree=4, include_bias=False)
train_X = poly_features.fit_transform(X.reshape(-1, 1))

lambda_ = 1.0   # Regularization factor
train_model = linear_model.Ridge(alpha=lambda_, fit_intercept=True)   # Create a
ridge regression model
train_model.fit(train_X, Y)   # Training

# Get model coefficients
# C[0] - intercept (constant term)
# C[1], C[2], C[3], C[4] - unpack model coefficients
C = [train_model.intercept_, *train_model.coef_]
print("="*20, "# Model Parameters", "="*20, sep='\n')
for i, v in enumerate(C):
    print("C%d = %.4f" % (i, v))

pred_Y = train_model.predict(train_X)   # Predicting
# with squared=False, mean_squared_error calculates RMSE
# Otherwise, mean_squared_error calculates RME
train_rmse = mean_squared_error(Y, pred_Y, squared=False)
print("="*20, "# Predict", "="*20, sep='\n')
print("Train RMSE = %.3f" % train_rmse)
"""
====================
# Model Parameters
====================
C0 = 0.4519
C1 = 1.4958
C2 = -1.0236
C3 = 0.1851
C4 = -0.0099
====================
# Predict
====================
Train RMSE = 0.507
"""
```

（2）根据（1）中生成的曲线图，简要分析应当怎样选取正则化参数λ。**给出你认为最优的λ，以及在此λ下拟合得到的模型，与此模型在训练集和测试集上的RMS。**

（3）用岭回归拟合数据，以$\cos x, \cos^2 x, \cos^3 x, \cdots$为基（拟合至4阶，需拟合常数项），**绘制合适的图表**，表示在训练集和测试集上的均方根误差RMS及拟合系数（$C_0, C_1, C_2, C_3, \cdots$）随正则化参数λ的变化关系（横轴用对数坐标或用$\ln \lambda$作为标度）

（4）根据（3）中结果，**给出你认为最优的λ，以及在此λ下拟合得到的模型，与此模型在训练集和测试集上的RMS。**

（5）已知原数据的模式为

$$y = C_0 + x + C_1 \cos x + C_2 \cos^2 x + C_3 \cos^3 x + C_4 \cos^4 x$$

给出拟合系数以及相应的均方根误差RMS。

（6）将文件打包上传，压缩包中应有文件

- Python源代码`ridge_regression.py`
- 分析报告`*.doc/*.docx/*.pdf`
- 必要的说明文件`*.txt`，图片文件`*.jpg/*.png/*.tiff`等

（如果使用Markdown写报告，请将报告转为pdf格式）

重要提示：

上机作业中，参考模板类合理组织代码，在编写代码时尽可能保持代码良好的可阅读性

模板代码：

```
#!/usr/bin/env python
#coding=utf-8

"""
Author:        Fanchong Jian, Pengbo Song
Created Date:  2021/09/28
Last Modified: 2021/09/29
"""

from warnings import warn
from sklearn import linear_model

class MLChemLab1(object):
    """Template class for Lab1 -*- Linear Regression -*-

    Properties:
        model: Linear model to fit.
        featurization_mode: Keyword to choose feature methods.
    """

    def __init__(self):
        """Initialize class with empty model and default featurization mode to
identical"""
        self.model = None
        self.featurization_mode = "identical"

    def fit(self, x, y, featurization_mode : str):
        """Feature input X using given mode and fit model with featurized X and
y

        Args:
            x: Input X data.
            y: Input y data.
            featurization_mode: Keyword to choose feature methods.

        Returns:
```

```
            Trained model using given X and y, or None if no model is specified
        """

        # Catch empty model, a model should be added earlier
        if (self.model is None):
            warn("No model to fit. Nothing Returned.")
            return None
        self.featurization_mode = featurization_mode
        featurized_x = self.featurization(x)
        self.model.fit(featurized_x, y)
        return self.model

    def add_model(self, model : str, **kwargs):
        """Add model before fitting and prediction"""
        if model == "ridge":
            # Put your Ridge Regression model HERE
            self.model = ...
        elif model == "lasso":
            # Put your Lasso Regression model HERE
            self.model = ...
        elif model == "naive":
            # Put your naive model HERE
            self.model = ...
        else:
            # Catch incorrect keywords
            raise NotImplementedError("Got incorrect model keyword " + model)

    def featurization(self, x):
        """Feature input X data using preset mode"""
        if self.featurization_mode == "poly":
            # Put your polynomial featurization code HERE
            return ...
        elif self.featurization_mode == "poly-cos":
            # Put your polynomial cosine featurization code HERE
            return ...
        elif self.featurization_mode == "identical":
            # Do nothing, returns raw X data
            return x

    def predict(self, x):
        """Predict based on fitted model and given X data"""
        x = self.featurization(x)
        y_predict = self.model.predict(x)
        return y_predict

    def evaluation(self, y_predict, y_label, metric : str):
        """Eavluate training results based on predicted y and true y"""
        if metric == "RMS":
            # Returns RMS value
            return ...
        else:
            # Catch incorrect keywords
            raise NotImplementedError("Got incorrect metric keyword " + model)
```