Asmt 6: Regression

Turn in through Gradescope by 11:59 PM on Wednesday, March 29:
100 points

Overview

In this assignment, you will explore regression techniques on high-dimensional data. **I am intentionally not providing you a link to the documentation to make you familiarize yourself with navigating the `sickit-learn` documentation.**

Report implementation only where asked.

Note: Homework assignments are intended to help you learn the course material, and successfully solve mid-term and final exams that will be done on paper in person.

As usual, it is recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. The latex source for this homework is available on Canvas. I recommend that you drop the two latex files (.tex and .sty) in a new Overleaf project, and compile/edit the .tex file there.

Submissions that are not uploaded on Gradescope will get 10% penalty.

1 Regression, Regularization, and Cross-Validation (100 points)

You will explore regression to model the diabetes progression in one year. You will use a dataset of $N = 442$ examples (diabetes patients). Each example includes $n = 10$ input variables x_i: age, sex, body mass index, average blood pressure, and six blood serum measurements, as well as the output variable y which is a quantitative measure of disease progression after one year. The entire data can be loaded as follows:

```
from sklearn.datasets import load_diabetes
X, y = load_diabetes(return_X_y=True, as_frame=False)
```

Each row of X represents one patient's data and the corresponding row in y is that patient's progression.

We need a train-test split of the data to be able to both calculate the weights w and evaluate our solution on a held-out test set. To get a train-test split we change the data above as follows:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,shuffle=False)
```

A (20 points): Go over the class `LinearRegression` from the module `sklearn.linear_models`. Use the default parameters of `LinearRegression`. Estimate the weights w of linear regression calculated using `X_train`, `y_train` above. Be careful that you use the training data.

(i) Report the formula for your model with the learned weights w. Watch out for w_0, don't miss it.

(ii) Provide your implementation.

Answer (i):
Answer (ii):

B (10 points):

(i) Report the error of the model $h(x; w) = w^T x$ you learned in **A** on the train data, so $E(w|(X_{\text{train}}, y_{\text{train}}))$.

(ii) Report $E(w|(X_{\text{test}}, y_{\text{test}}))$

(iii) Why is the test error larger?

(iv) Provide your implementation of the error function.

Answer (i):
Answer (ii):
Answer (iii):
Answer (iv):

C (20 points): Go over the class `PolynomialFeatures` from the module `sklearn.preprocessing`. Calculate the degree-2 polynomial features of your input training data, $\Phi(X_{\text{train}})$. Use the default parameters of `PolynomialFeatures`. Use `LinearRegression`, *polynomial* features, and the training data to estimate the weights w of polynomial regression. Be careful that you use the training data.

(i) Report the learned weights w (don't need to report the formula, it's too long). Watch out for w_0, don't miss it.

(ii) Provide your implementation.

Answer (i):
Answer (ii):

D (10 points):

(i) Report the error of the model $h(x; w) = w^T \Phi(x)$ you learned in **C** on the train data, so $E(w|(\Phi(X_{\text{train}}), y_{\text{train}}))$.

(ii) Report $E(w|(\Phi(X_{\text{test}}), y_{\text{test}}))$.

(iii) Compared to the test error in **A**, what do you conclude about the polynomial model?

Answer (i):
Answer (ii):
Answer (iii):

E (20 points): Go over the class `Ridge` from the module `sklearn.linear_models`. Use the regularization factor $\alpha \in \{0, 0.001, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 1, 10\}$ (called λ in class) and the degree-2 polynomial features of your input training data to estimate the weights w of Ridge regression. For each α, calculate $E(w|(X_{\text{test}}, y_{\text{test}}))$.

(i) Report your implementation of this Ridge regression - please don't report the implementation for every α, just one.

(ii) Report a plot with the regularization factor, α, on the x-axis and the error, $E(w|(X_{\text{test}}, y_{\text{test}}))$, on y-axis.

Answer (i):
Answer (ii):

E (20 points): Repeat **E** but instead of calculating one $E(w|(X_{\text{test}}, y_{\text{test}}))$, use 5-fold Cross-Validation and for every α calculate the *average* of five test errors calculated on five test splits. To create data folds from the original entire data, you can use this code:

```
from sklearn.model_selection import KFold

n_splits = 5
kf = KFold(n_splits=n_splits)
kf.get_n_splits(X)

for i, (train_index, test_index) in enumerate(kf.split(X)):
    fold_train_X = X[train_index]
    fold_train_y = y[train_index]
    fold_test_X = X[test_index]
    fold_test_y = y[test_index]
```

(i) Report a plot with the regularization factor, α, on the x-axis and the *average* of five test errors calculated on five test splits on the y-axis.

(ii) When we pick the best α, what else we would ideally do to report a better estimate of our model on unseen instances?

Answer (i):
Answer (ii):

2 Bonus: Matching Pursuit (5 points)

Files `M.csv` and `W.csv` are available on Canvas. Consider a linear equation `W = M*S` where `M` is a measurement matrix filled with random values $\{-1, 0, +1\}$ (although now that they are there, they are no longer random), and `W` is the output of the sparse signal `S` when measured by `M`.

Use Matching Pursuit (as described in the book as **Algorithm 5.5.1**) to recover the non-zero entries from `S`. Record the order in which you find each entry and the residual vector after each step.

Answer: