Lecture 2: Assignment 03
Polynomial Regression, Bias Variance Tradeoff, Regularization

EECE-6822 Machine Learning
Cris Ababei
Electrical and Computer Engr., Marquette University

1. Objective

The objectives of this activity include: (1) run several given code examples that investigate bias-variance tradeoff; and (2) investigate the use of regularization.

2. Prerequisite Readings

Murphy
- Bias-variance trade-off: Murphy 4.7.6
- Ridge regression: Murphy 11.3-11.4

Mostafa
- Generalization: 2.2-2.3
- Overfitting; regularization: 4.1-4.2

Geron
- Ch.4 (Polynomial Regression Section, up to Logistic Regression) talks about polynomial regression, ridge regression (adds regularization term to MSE cost function), Lasso regression (also adds regularization term to MSE cost function, but uses l1 norm of the weight vector, not l2), Elastic Net regression. It also talks about overfitting; early stopping.

Raschka
- Ch. 9: Discussion of regression models to predict target variables on a continuous scale. It talks about exploring and visualizing datasets. Several approaches to implement linear regression models. Training regression models to be robust to outliers. Fitting regression models to nonlinear data.

3. Code Examples

Example 1: Bias-Variance Demo - Python, NumPy

Open your google colab and upload the notebook, provided with the files for this lecture:
code/bias_variance_demo.ipynb
This notebook provides an example that illustrates how for an $\eta(x)=E[Y|X=x]$ (i.e., ground truth) a polynomial of degree 5, we get underfitting with a degree-3 polynomial linear regression (large bias small variance) and overfitting with a degree-12 polynomial linear regression (small bias, large variance). The example also defines $P(Y|X=x)=N(x,\sigma^2)$.
Note that the example introduces "expected test error", which is called "true error": **Ltrue=E[Ltest]**

- "test error" is an unbiased estimate of the "true error"
- "true error" is unobservable (we cannot compute it, given finite samples)
- yet, we care the most about the "true error"
- so, we use "test error" as a surrogate or an approximation for the "true error"

Note that regression is implemented directly in Python, using matrix operations to calculate: **w_hat=$(X^T X)^{-1} X^T y$**
Go through all the steps of the notebook. Make sure to understand it in its entirety.
Takeaways:
- Degree 3 fit has very high bias; and moderate variance.
- Degree 12 has very low bias; and high variance.

- Degree 5 has very low bias; and small variance.

Example 2: Regularization; ridge regression - Python, NumPy, SciKit-Learn

Open your google colab and upload the notebook, provided with the files for this lecture:
code/ridge_lasso_kaggle.ipynb
Go through the steps of the notebook **until and including Ridge Regression**.
Note that this notebook uses **Ames Iowa Housing Prices** Dataset, which is also included in the files provided; so, do not forget to upload it to your colab.

This notebook provides an example where 23 features (out of 80?) from the dataset are selected; then, simple linear regression is done for each input feature individually using **np.linalg.lstsq**.

Dataset of length 1459 is split into train and test portions (80/20); it preprocesses the data to standardize it (remove mean and standardize variance).

It uses linear regression (**LinearRegression** from SciKit) on train portion.

Then, it runs ridge regression (**Ridge** from SciKit) for different regularization values lambda.

Example 3: Polynomial Regression – SciKit-Learn – Geron example

In this activity, we use code examples from:
[*B3-Geron] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.
We continue to look at the examples from Chapter 4. The corresponding notebook is at:
https://github.com/ageron/handson-ml3/blob/main/04_training_linear_models.ipynb
Launch that in your google colab and go through the steps of **Polynomial Regression** and **Ridge Regression**.

In this notebook, regression is done on a small array (size m=100) with randomly generated X and y as a linear degree-1 polynomial:

X = 2 * np.random.rand(m, 1) # column vector

y = 4 + 3 * X + np.random.randn(m, 1) # column vector

Try to understand the code; pay attention to:

- The use of lin_reg = LinearRegression(), its training/fitting, and use for prediction: y_new = lin_reg.predict(X_new_poly)
- The plot (Figure 4-14 in Geron's book) titled High-degree polynomial regression. Look at this plot and discuss it in terms of bias variance tradeoff!
- The use of learning_curve() of SciKit-Learn used to train and evaluate the model using cross-validation; it shows that LinearRegression() underfits the model (which is quadratic); it then shows how **PolynomialFeatures**(degree=10) overfits the model.
- The use of Ridge() of SciKit-Learn for ridge regression

Example 4: Polynomial Regression – SciKit-Learn – Raschka example

This activity uses code example from:
[*B3-Raschka] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili, Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python, Packt Publishing, 2022.

We will continue the notebook from Ch. 9, which we started in previous assignment.
So, work locally on your computer directly using the Jupyter notebook (or the equivalent Python code), or launch it in your google colab (see previous assignment document on how to do that).
Open or upload into Google Colab the notebook for Ch. 9, located at:
https://github.com/rasbt/machine-learning-book/blob/main/ch09/ch09.ipynb
and go through all the steps concomitantly with reading the chapter.
Focus on the **polynomial regression** part.

Recall that the examples in this chapter are applied on the **Ames Housing Dataset**. Continue with the rest of the notebook from where we stopped in previous assignment: using regularized methods for regression: regularization is one approach to tackling the problem of overfitting by adding additional information and thereby shrinking the parameter values of the model to induce a penalty against complexity.
Then, study polynomial regression for dealing with nonlinear relashionships.
Optionally, continue even further with Random forests, and Decision trees.

Example 5: Bias Variance Tradeoff, Polynomial Regression – mlm tutorials and code examples

In this part, we will look at several tutorials from **machinelearningmastery (mlm)**, which has several tutorials on bias-variance tradeoff. Read the following tutorials and run the code where applicable:

Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning
https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/

How to Reduce Variance in a Final Machine Learning Model
https://machinelearningmastery.com/how-to-reduce-model-variance/

How to Calculate the Bias-Variance Trade-off with Python
https://machinelearningmastery.com/calculate-the-bias-variance-trade-off/

How to Use Polynomial Feature Transforms for Machine Learning
https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/

4. Assignment

--(a)Create a new Jupyter notebook (or work directly as a Python program) that starts with the code from the **Polynomial Regression** Section of Example #3 above. You will need actually only the first part of that Section; you can ignore the code that generates figures 4-13 and 4-14.

--(b)Add your own code to the newly created notebook to import and work with the Ames Iowa Housing Prices Dataset. It is the dataset from Example #2 above. As done in that example, use only the 23 selected features.

--(c)Identify what are the 3 features (out of all 23) that individually give the top three lowest MSE values. Basically, here you must run 23 individual liner regressions - similarly to the first part of the example **linear_regression_demo.ipynb**, provided with the .zip archive for this lecture. Feel free to use that code for this step. Once you found those three most relevant/important features, keep those three columns only in X. Also, split now the three-column reduced dataset into train and test portions (80/20%). Go to the next step.

--(c)Run a number of six (6) simulations to investigate different degree-p polynomial regressions. Each such simulation would be essentially a re-run of the code from part (a). Try values for p={1,2,3,5,8,10}. Note that here, we use PolynomialFeatures() and LinearRegression() and not any manual coding of the polynomial regression.

--(d)For each of the above six simulation experiments, calculate train and test errors. Save them and plot them at the end in order to generate a plot similar to the one at the end of the example in **demo_polynomial.ipynb** (provided in the .zip archive for this lecture). That plot looks like the one in the figure below:

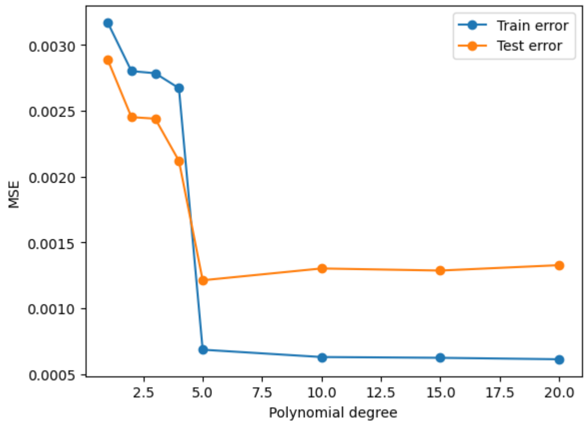

Figure 1: Plot that illustrates variation of train and test errors as a function of model complexity.

--(e)Present a discussion of your findings.

5. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**hw3_report_LastName.pdf**". You should also create a .zip archive with all your code and implementations of all parts of the assignment. Upload also this archive .zip file with the name "**hw3_implementation_code_LastName.zip**" to D2L. Hence, your D2L should contain two items: the report and the .zip file. **Do not include the report inside the .zip and upload only the .zip. They should be two separate items!**

The report should include the following sections and subsections:

1) Title + course info + your name
2) **Summary.** Describe in one paragraph what the objective of the assignment is.
3) **Description.** Describe the investigation you did to find polynomial degrees that underfit, good fit, and overfit. Include plots to aid your description of the results you obtained. All plots in your figures should have axes labels, and all figures should be numbered and have short captions describing what the figures present.
4) **Conclusion.** Present your conclusions; highlight what are your main takeaways that you learned from this lecture and assignment. Describe what issues you encountered and how you solved them.
5) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them! If your report has References that are not cited in the report, points will be deducted!