Take-Home Assignment 1

Introduction to Machine Learning (BPINFOR-113)
Grading scheme: 20 points (0-20)
Weight for final grade: 10%

Responsible: Decebal Mocanu, Christopher Gadzinski

Release date: October 3, 2024.
Deadline: October 28, 2024, 23:59.

Submit a report (PDF or HTML) with your solutions and submit it through Moodle. A template project capable of generating an HTML report is available at `https://github.com/cgadski/iml_template/`.

Requirements

- All groups should have three members. Write your names in the report.
- Include the code you wrote to solve the problems. If all your code is in a notebook, an export of your notebook is enough. Otherwise, attach source files to your submission.

A. Supervised Learning: Regression (8 points)

For this exercise you'll use `powerplant.csv`, a dataset collected at a combined cycle power plant. You get two attributes:

- X: ambient air temperature outside the power plant, measured in Celsius.
- Y: energy output of the plant, measured in megawatts.

Your goal is to model Y a function of X.

A.I. Linear Regression (5 points)

1. (0.5 points) Where n is your group number, take the 20 data points starting at index $20n$. You will use these 20 data points for regression. Graph them.
2. (0.5 points) Perform some transformation of your data. (Suggestions: normalize ranges, normalize Z-scores, etc.). Explain why this transformation may be useful.
3. (0.5 point) Compute the coefficients of a linear regression model $Y \approx aX + b$ using the closed-form solution for the least squares problem. Print them. Plot the regression line against the data points.
4. (1 point) Choose a cost function for a linear regression model. Motivate your choice. Find the partial derivatives of the cost function with respect to your regression parameters.
5. (0.5 point) Choose arbitrary initial values for your regression parameters. Plot the corresponding regression line against the data and compute the cost function. Perform one iteration of gradient descent and check that the cost has decreased.
6. (1 point) Find a good step size experimentally and iterate gradient descent until convergence. Plot the cost function over all iterations. What happens if your step size is too large or too small?
7. (1 points) When solving a least squares problem, why might we prefer an iterative method over a "closed form" solution?

A.II. Polynomial Regression (3 points)

Now you'll fit a quadratic model

$$h_\theta(x) = \theta_2 x^2 + \theta_1 x + \theta_0$$

with parameters $\theta = (\theta_0, \theta_1, \theta_2)$. Your cost function will be

$$J(\theta) = \frac{1}{4n} \sum_{i=1}^{n} (y^{(i)} - h_\theta(x^{(i)}))^4.$$

1. (1 points) Differentiate the cost function with respect to regression parameters and write the update rule you'll use for gradient descent.
2. (1.5 points) Optimize your model with gradient descent. Plot the cost function over all iterations, and plot the function you obtained.
3. (0.5 points) Explain how polynomial regression compares to linear regression. If we use least squares as our cost function, is there a closed-form solution for the optimal parameters of a polynomial regression model?

B. Supervised Learning: Classification (4 points)

1. (0.5 points) Generate an artificial dataset for testing binary classification methods. It should have two continuous independent attributes, one binary "class" attribute, and 40 data points in each class. Plot it.
2. (0.5 points) Split your data into a training and a test set and add four outliers (two per class) to the training set. Why would we need a test set? How do we choose its size?
3. (1 point) Use scikit-learn to train the following classification models:
   - k-nearest neighbors
   - naive Bayes
   - a decision tree
   - a random forest

   In each case, report training accuracy, test accuracy, and confusion matrices on test data.
4. (1 point) Graph the decision boundaries produced by each model. (See `https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`.)
5. (1 point) What model performed the best? Describe a different classification problem where one of the alternatives would have been better.

C. Unsupervised Learning (8 points)

C.I. k-means Clustering (4.5 points)

1. (0.5 points) Study k-means and briefly describe how this model is trained. (Aim for 3 sentences.)
2. (0.5 points) Use scikit-learn to run k-means on the dataset in `iris.csv`. Produce scatter plots of each pair of attributes, colored by the k-means clusters. (You can use `seaborn.pairplot`.)
3. (0.5 points) Compute the average silhouette score for each cluster. Explain what this measures.
4. (1.0 point) Compare the clusters inferred by k-means with the data labels using a confusion matrix. Did k-means accurately distinguish the three species?
5. (1.0 point) `unknown_species.csv` contains data on 6 flowers of unknown species. Use your trained k-means model to predict which cluster these new flowers belong to.
6. (1.0 point) Implement k-means from scratch and test your implementation on the iris dataset.

C.II. DBScan and PCA (3.5 points)

1. (0.5 points) Study DBScan and briefly describe how this algorithm works. (Aim for three sentences.)
2. (0.5 points) Read the dataset in `ulu.csv` and apply scikit-learn's implementation of DBScan to infer clusters. (Keep parameters at their defaults.) How many clusters did you get?
3. (0.5 points) Plot some projections of the data into two dimensions. Color using the clusters inferred by DBScan. What shapes do you recognize?
4. (1 point) Use PCA to find a two-dimensional projection of maximum variance and plot the data under this projection. Informally describe what this transformation does. For 3-dimensional data, what does it do geometrically?
5. (0.5 points) Now run k-means instead of DBScan. Compare their silhouette scores.
6. (0.5 points) Would silhouette scores be a good cost function to learn the interpretable clusters in this data?

Bonus Problem (4 points)

Conventional wisdom says that a model with too many parameters will "overfit," meaning that it will learn to explain the training data without learning the underlying distribution. However, in some situations, we can scale the number of parameters (even to infinity!) without overfitting. This behavior is typical of deep neural networks. In these last two questions, you can explore why an overparameterized model may (or may not) overfit the test data.

These problems are optional—your grade will be capped at 20 points.

1. (2 points): Set up a regression problem with 30 noisy two-dimensional data points (x_i, y_i). Compare the best fit of a model with 3 parameters to the best fit of a model with 30. (Suggestion: solve a least squares problem for a linear combination of sinusoids.)
2. (2 points): (Challenge.) Now consider a model

$$f(x) = \sum_{i=1}^{N} \theta_i f_i(x)$$

where the N "basis functions" f_i are random generated. Suggestion: use

$$f_i(x) = R_1 \,\mathrm{ReLU}(R_2(x - Z))$$

where R_1 and R_2 are Rademacher variables (-1 or 1 with equal probability) and Z is uniformly random over your domain. Initialize $\theta = 0$, set $N = 1000$, and train this model on your 30 data points using gradient descent. Does this model overfit?

Success!