Problem Set - Cost Functions

October 29, 2021

Problem 1: Regularization

A linear model such as logistic regression or linear regression might overfit the data when there is a large number of features. One way to address this issue is to regularize or shrink the contribution of some of the features, by reducing the magnitude of their corresponding weights or coefficients. By doing so, we can reduce the variance of the predicted values, and we can enhance the interpretability of the linear model by determining a smaller subset of the most important features.This is achieved by adding a penalty or regularization term to the cost function, which forces the learning algorithm to not only fit the data but also to keep the weights of the model as small as possible.

Regularized versions of linear regression

One way to regularize the linear regression is to modify the cost function as follows:

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\beta_0+\sum_{i=1}^{p}\beta_j x_j^{(i)}-y^{(i)}\right)^2}_{\text{MSE}}+\lambda\sum_{j=1}^{p}\beta_j^2$$

$\lambda \sum_{j=1}^{p} \beta_j^2$ (also known as the ℓ_2-norm of the weight vector) is the regularization term or shrinkage penalty. λ is the regularization hyperparameter that you can use to control how much you want to regularize the model. Note that it is important to scale the data before performing regularization.

1. Regularization hyperparameter:
    - What is the difference between a model's parameter and a model's hyperparameter?
    - As we incease λ from 0, how will the training MSE (mean-squared error) change? How will the test MSE change? Sketch the bias-variance trade-off.
    - Suppose you used regularized linear regression and noticed that the training and validation errors are almost equal and high. Should you increase or decrease the regularizarion hyperparameter λ?
2. Regularizing linear regression using the ℓ_2-norm is known as the ridge regression. There is another version of regularized linear regression knowsn as the Lasso regression, defined as follows:

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\left(\beta_0+\sum_{i=1}^{p}\beta_j x_j^{(i)}-y^{(i)}\right)^2}_{\text{MSE}}+\lambda\sum_{j=1}^{p}|\beta_j|$$

$\lambda \sum_{j=1}^{p} |\beta_j|$ is known as the ℓ_1-norm of the weight vector. An important characteristc of the Lasso regression is that it can force some of the weights to be exactly equal to zero when the hyperparameter λ is high enough. On the other hand, ridge regression shrinks all of the weights toward zero but does not set any of them to zero.

- Load the boston dataset using `load_boston`.
- Split the data into 80% for training and 20% for testing using `train_test_split`. Set the `random_state` to 23.
- *Part 1*: Use the ridge regression implementation in scikit-learn and fit several ridge regression models to the training data. Use a different λ (`alpha` in scikit-learn) for each of these models (you can try values in `np.arange(0, 100, 0.1)`). For each fitted model extract its coefficients and compute their norms. How is the norm of the coefficients changing by varying the hyperparameter λ. Make sure to scale the data before fitting each model.
- *Part 2*: Perform cross validation on the training dataset, to select the best model between: linear regression, lasso, and ridge regression. For each of the ridge regression and lasso models, try different values for λ (or alpha in sklearn), and then choose a final model and evaluate it on the test set. Note: to perform cross-validation (5-fold), you can use `cross_validate` and specify the scoring as `neg_mean_squared_error` or `mean_squared_error`.

Regularized version of logistic regression: Logistic regression can be also extended by including a regularization term to the cost function of logistic regression. The regularization term can be in terms of the ℓ_2-norm or ℓ_1-norm of the weight vector.

Problem 2: Class Weighted Algorithm

We've seen that the cost functions used in training the machine models can be written as:

$$\sum_{i=1}^{N} Loss(y_{pred}^{(i)}, y_{true}^{(i)})$$

ML models used for classification assume that there is an equal number of samples observed from each class, and that all losses incured due to missclassification are the same. However, this is not always the case, as the data might be imbalanced and where the minority class is the class of interest for us.

One way to mitigate the class imbalances is to modify the cost function to the following:

$$w_1 \sum_{i \in \text{class 1}} Loss(y_{pred}^{(i)}, y_{true}^{(i)}) + w_0 \sum_{i \in \text{class 0}} Loss(y_{pred}^{(i)}, y_{true}^{(i)})$$

w_1 and w_0 are weights assigned to each class, where w_1 is chosen in order to draw the attention of the learning algorithm to the minoriy class. When the weighted version of the cost function is implemented, we refer to the modified version of the algorithm as the class weighted algorithm or weighted algorithm.

- If class 1 is the minority class, how w_1 should be chosen with respect to w_2?
- Load the `creditcard.csv` data.
  - The last column represents the class label. Check the proportion of each class and comment on the severity of the class imbalance.
  - Split the data into training and testing (20% for testing).

- Process the training set by scaling the features using `StandardScaler`, scale the testing set in the same way.
- Fit two models on the training data: logistic regression and weighted logistic regression (by setting `class_weight` to "balanced").
- Test the two models on the testing set using the recall score.