Lecture 15

A generalized penalty term, also known as a regularization term, is a term added to the regression model that penalizes certain types of model complexity. This is used to prevent overfitting and improve the model's predictive accuracy on new data.

There are two common types of regularization methods used in regression models: L_1 regularization (Lasso) and L_2 regularization (Ridge).

L_1 regularization adds a penalty term to the regression model equal to the absolute value of the coefficients. This leads to sparse models with only a few important features having non-zero coefficients, and the remaining coefficients being set to zero. This type of regularization is useful when the number of features is high and we want to select only the most important features for the model.

L_2 regularization adds a penalty term to the regression model equal to the square of the coefficients. This leads to all the features having non-zero coefficients, but with smaller values. This type of regularization is useful when we want to reduce the impact of large coefficients on the model output.

Both regularization methods allow us to control the complexity of the model by adjusting the penalty term, which can be done using cross-validation or other optimization techniques. The regularization term can be added to any regression model, including linear regression, logistic regression, and others.

Other kinds of penalty terms are possible, but less common.

The pros of using a generalized penalty term include:

- Helps prevent overfitting
- Improves the model's predictive accuracy on new data
- Allows for the selection of important features
- Can be used with any regression model

The cons of using a generalized penalty term include:

- Makes the model more complex to interpret
- Requires choosing an appropriate penalty parameter, which can be difficult
- May not work well with highly correlated variables
- May not work well with non-linear relationships between variables.

There are other types of penalties used in regression models besides Ridge and LASSO. Some of them are:

Elastic Net: Elastic Net is a combination of LASSO and Ridge penalties, which involves both L_1 and L_2 regularization. The penalty term is a combination of the absolute value of the coefficients and the squared value of the coefficients.

Least Absolute Shrinkage and Selection Operator (LASS): LASS is a type of regularization that reduces the impact of less important features in the model. It shrinks the coefficients towards zero and makes the less important features exactly zero.

Smoothly Clipped Absolute Deviation (SCAD): SCAD is another type of regularization that is similar to LASSO. It also shrinks the coefficients towards zero and sets the less important features to zero. However, it has a smoother transition than LASSO, which avoids the variable selection bias that LASSO can produce.

Adaptive LASSO: Adaptive LASSO is a modification of LASSO, which adapts the penalty to the different levels of importance of the predictors. It puts less penalty on the more important predictors and more penalty on the less important predictors.

Bridge Regression: Bridge regression is a type of regularization that combines the L_1 and L_2 penalties. It can be seen as a generalization of both LASSO and Ridge regression, and it allows for both variable selection and shrinkage.

These penalties are used to avoid overfitting and improve the generalization performance of the model. They are useful in situations where the number of predictors is large relative to the sample size, and the predictors are highly correlated. When we look at non-parametric regression, or other kinds of regression in machine learning, these penalties can be more common, and in principal, can be more varied, or based on other features besides the coefficients such as the number of coefficients, the size of the tree, etc.

Some examples include:

Group Lasso: A penalty that encourages sparsity within groups of variables. This can be useful in situations where there are many variables that can be grouped together based on some prior knowledge or domain expertise.

Total Variation: A penalty that encourages a smooth solution by penalizing the differences between adjacent values in a sequence. This can be useful in image and signal processing applications.

Fused Lasso: A penalty that encourages a sparse and piecewise-constant solution by penalizing both the L1 norm of the coefficients and the differences between adjacent coefficients. This can be useful in applications where the underlying signal is expected to have sharp transitions.

Nuclear Norm: A penalty that encourages low-rank solutions by penalizing the sum of the singular values of a matrix. This can be useful in matrix completion and collaborative filtering applications.

Validation:

There are several methods for validating models in machine learning, including:

Hold-out validation: This involves splitting the dataset into a training set and a validation set. The model is trained on the training set and then evaluated on the validation set.

Cross-validation: This involves dividing the dataset into k equal parts (or folds). The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold used as the validation set once. The performance of the model is then averaged over the k runs.

Bootstrap validation: This involves resampling the dataset with replacement to create multiple training and validation sets. The model is trained on each training set and evaluated on the corresponding validation set. The performance of the model is then averaged over all the runs.

Leave-one-out cross-validation (LOOCV): This is a special case of k-fold cross-validation where k is equal to the number of observations in the dataset. The model is trained on all but one observation and evaluated on the left-out observation. This process is repeated for each observation in the dataset. Monte Carlo cross-validation: This involves randomly partitioning the dataset into training and validation sets multiple times and averaging the performance of the model over all the runs.

These methods can be used to assess the performance of a model, compare the performance of different models, and tune the hyperparameters of a model.

In machine learning, data is typically divided into three sets: training set, validation set, and test set. The training set is used to fit the model, the validation set is used to tune hyperparameters, and the test set is used to evaluate the performance of the final model.

The process of selecting the test and train sets involves randomly partitioning the available data into two separate sets, ensuring that they are representative of the whole data. Typically, a split of 70/30 or 80/20 is used, with the larger portion being the training set.

It's important to ensure that the test set is independent of the training set and that there is no overlap between them. This is to prevent the model from memorizing the training data and performing poorly on new, unseen data. If you are going to normalize your values, you need to do it after splitting, otherwise, values that appear in the test set will have some affect on the normalization, which can make the test procedures less valid.

In some cases, where the dataset is small, cross-validation techniques may be used to evaluate the model's performance. In this case, the data is partitioned into multiple folds, with each fold being used as a test set in turn, and the remaining data used as the training set. This helps to get a more reliable estimate of the model's performance on unseen data.

K-fold cross-validation is a method used to evaluate the performance of a machine learning model. The basic idea behind this method is to divide the data set into K-folds or subsets of equal size. Then, for each fold, the model is trained on K-1 subsets and tested on the remaining subset. This process is repeated K times so that each subset is used once as the validation data, and the remaining K-1 subsets are used for training the model. If the data set size is small and the value of k relatively large, the test sets may be too small to be useful.

The performance of the model is then evaluated by averaging the performance over the K folds. This provides an estimate of how well the model will generalize to new data that it has not been trained on. K-fold cross-validation is a popular method because it can provide a more accurate estimate of model performance than traditional validation methods, especially when the data set is small or there is a high degree of variability in the data. It is also helpful in avoiding overfitting the model to the training data.

The sampling method can affect cross-validation results. Cross-validation involves splitting the data into subsets, and the sampling method used to create these subsets can impact the performance of the

model. For example, if the data is imbalanced, meaning one class is underrepresented, using a random sampling method may lead to unequal representation of the classes in the training and testing sets. In this case, stratified sampling may be used to ensure that each class is proportionally represented in each subset. Additionally, if the data contains spatial or temporal autocorrelation, it may be important to use a sampling method that takes this into account, such as spatial or temporal clustering.

There are different sampling methods used in cross-validation, and the choice of sampling method depends on the data and the research question. Here are a few commonly used sampling methods:

K-fold cross-validation: This method divides the data into k equally sized folds, trains the model on k-1 folds and tests it on the remaining fold. This process is repeated k times, with each fold serving as the test set exactly once.

Leave-one-out cross-validation (LOOCV): This method uses all but one observation to train the model and tests the model on the left-out observation. This process is repeated for all observations, so each observation serves as the test set once. This method seems to work best in a classification setting.

Stratified cross-validation: This method ensures that the distribution of the target variable is maintained in each fold. This is particularly useful when the target variable is imbalanced, meaning there are more observations in one class than in the other.

Time-series cross-validation: This method is used when the data is time-dependent, and it ensures that the model is trained and tested on data that is chronologically consistent.

There are other sampling methods used in cross-validation, such as Monte Carlo cross-validation, which randomly samples observations to create training and test sets. The choice of sampling method depends on the data and the research question, and it is important to choose a sampling method that reflects the data-generating process and avoids introducing biases into the model evaluation.

Resources:

1. http://www.sthda.com/english/articles/36-classification-methods-essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/
2. https://daviddalpiaz.github.io/r4sl/regularization.html
3. http://faculty.washington.edu/yenchic/18W_425/Lec11_Reg03.pdf
4. https://statweb.stanford.edu/~jtaylo/courses/stats203/notes/penalized.pdf
5. https://appen.com/blog/machine-learning-model-validation/
6. https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7
7. https://datatron.com/what-is-model-validation-and-why-is-it-important/
8. https://medium.com/analytics-vidhya/what-is-model-validation-257686d0253e
9. https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data
10. https://machinelearningmastery.com/k-fold-cross-validation/