Introduction to Data Science

FEATURE SELECTION

BRIAN D'ALESSANDRO

WHY PERFORM FEATURE SELECTION?

The primary motivation for feature selection is to reduce model complexity, the benefits of which are:

- Lower expected model variance
- Easier interpretation of models
- Better scalability (both in training and deployment)
- Lower maintenance costs

Practical note: despite the opportunity *Big Data* provides for building models, we should never by default think it is better to use all of the data at our disposal. This is especially true with features. Having unnecessary features in our model can lead to a lot of downstream problems.

COMMON FEATURE SELECTION TECHNIQUES

- **"Naïve" methods** – pre-filter features based on heuristics
- **Best subset selection** – choose the best subset of k features from p features
- **Stepwise selection** – incrementally add/subtract features until model performance stabilizes
- **Dimensionality Reduction** – take rank-k approximations of X
- **Regularization** – Implicit, based on adding complexity penalties to loss function

NAÏVE SUBSET SELECTION

Choose top k based on a rule:

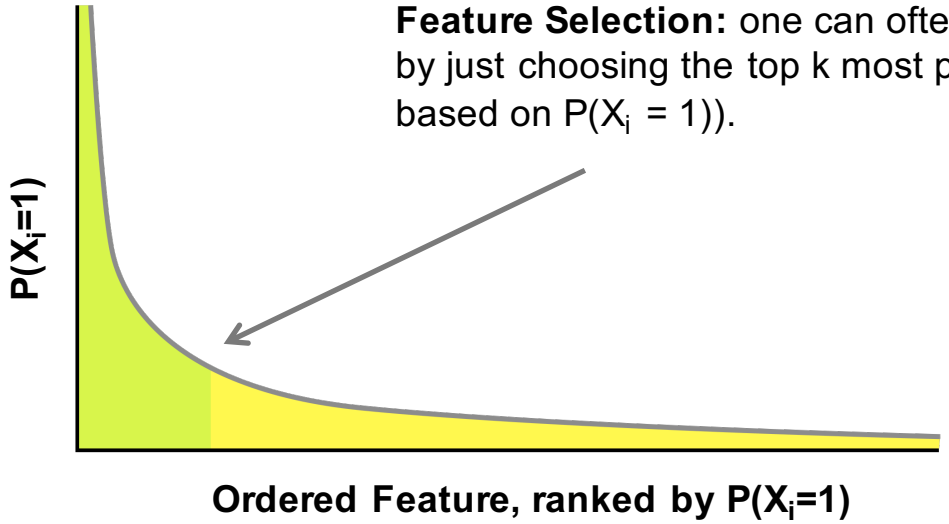
- Has highest Mutual Information/Correlation with Y
- Has the most coverage/support (i.e., % where X!=0)

Choosing k can be learned, but what makes up the top k (once k is chosen) is determined by the heuristic methods, and not standard model selection.

NAÏVE SUBSET SELECTION

Bag-of-words with long tail feature distributions

Some problems involve "the bag-of-words" representation. An instance can be characterized by a set of tokens U = $\{t_1, t_2, t_3, t_4, \ldots t_k\}$. We often convert each token to a binary feature, where the token id is the feature name. This type of transformation often results in very high dimensional, but sparsely populated matrices.



Feature Selection: one can often build good models by just choosing the top k most populated feature (i.e, based on $P(X_i = 1)$).

FORWARD STEPWISE SELECTION

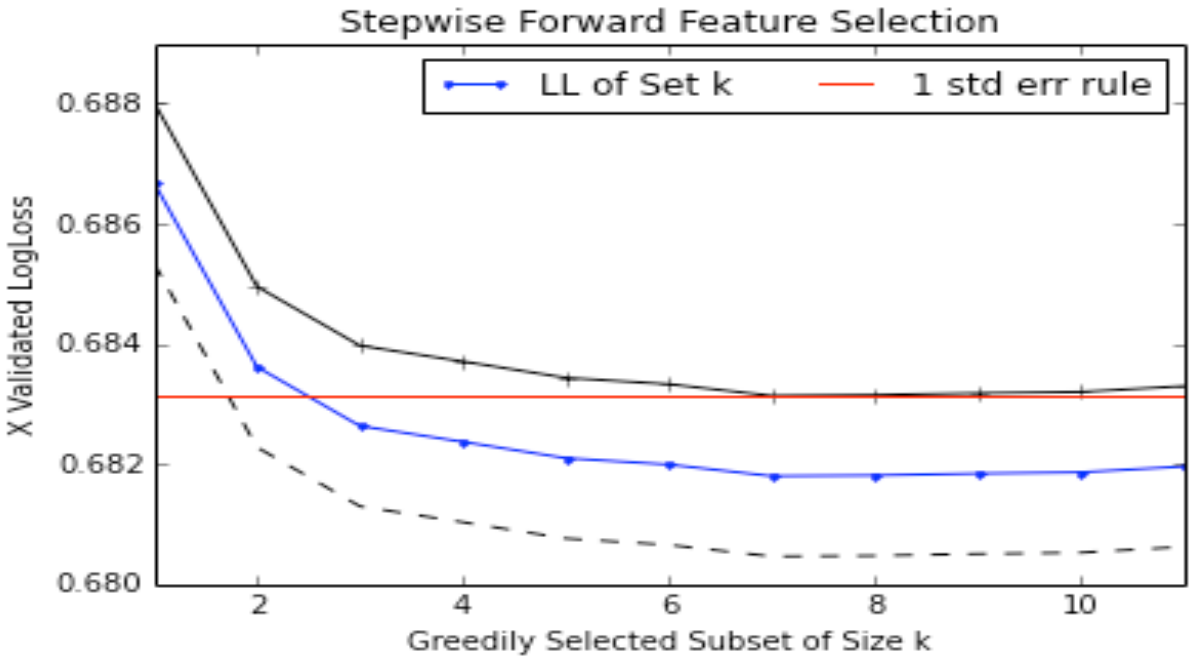
Forward stepwise selection is a greedy algorithm that incrementally selects the kth feature that improves the model after k-1 features have already been chosen.

Algorithm

- 1. Initialize: Curr_Best_Subset = { } (i.e., just the intercept)
- 2. Loop through each feature j:
  - a. Add the feature to the current best subset, train a model
  - b. Get out-of-sample error on the model trained with Curr_Best_Subset+feature j
- 3. Choose the feature with the best out-of-sample improvement in error
- 4. Add this best feature to Curr_Best_Subset, and log the out-of-sample error
- 5. Repeat steps 2 - 4 until stopping criterion is met.

FORWARD STEPWISE SELECTION

This plot shows the cross-validated LogLoss as we iteratively build the feature set, choosing the next best feature as the one that improves the cross-validated error the most. There are multiple ways to define a stopping criterion.



In all cases of feature selection, we should take an empirical view. The method for ranking features and the value *k* should be thought of as model hyper-parameters. With this view we can use standard model selection techniques (i.e. grid search & cross validation) to select the right set of features.

SVD AND THE LOW RANK APPROXIMATION

We can build a matrix X_k that approximates our original matrix by doing the following:

1. Compute the SVD of X
2. Truncate U, Σ and V to take only the k highest columns & singular values
3. Multiply back together to get X_k

Low Rank Approximation – K<<M

$$X_k = U_k \Sigma_k V_k^T$$

NxM *NxK* *KxK* $(MxK)^T$

SVD BASED FEATURE SELECTION

SVD can be used as a dimensionality reduction technique. Like subset methods, we choose k features, but the features are the transformed orthogonal columns of the rank-k approximation of X.

Compute rank-k SVD $X_k = U_k \Sigma_k V_k^T$

We can create an NxK reduced matrix with $X V_k$ This effectively becomes our new "X" matrix, and we do any analysis (clustering, modeling, etc.) with the reduced matrix.

We can project any new data into the reduced space by, $X' V_k$ and then use this in our algorithms.

SVD BASED FEATURE SELECTION: PRACTICAL NOTES

SVD based dimensionality reduction is equivalent to Principle Components based reduction. So you can use either implementation.

SVD can be computationally expensive on large matrices. This method may not scale well on bigger data, especially if you want to test multiple values of *k*

SVD renders the model uninterpretable. The transformed features carry the same information as the original matrix, but not in an explainable way.

SVD dimensionality reduction is unsupervised. While you might only be throwing away a small amount of information, that information may be disproportionately correlated with your target variable.

SVD will preserve features with larger variance more. It is better to normalize the data to unit variance first.

Practical advice: I see many students leverage this technique as a first step when building models on small feature sets (i.e., < 100). For the reasons listed here, this is my least preferred approach for dimensionality reduction. If using SVD/PCA, run a test against another method to fully justify its use.