

CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Zoom Link

Recap

- Bias variance analysis studies the sources of e_{avg} . It gives quantitative insight into i_{bias} .
- We can decompose the EGE of any learning algorithm into three components: n_{noise} , v_{variance} and b_{bias} .
- What does the first component describe?
- What does the second component describe?
- What does the third component describe?
- For an arbitrary loss function, the main prediction is the label for each point that $m_{\text{minimizes}}$ the expected loss with respect to the $p_{\text{distribution}}$ l_{loss} (in expectation over t_{training} s_{samples}).
- Using this, the bias error is the loss of the \hat{y} with respect to the y . The variance error is the loss of the \hat{y} with respect to \bar{y} . The noise error is the loss of the y with respect to the \bar{y} .
- We must try to balance b_{bias} and v_{variance} . But there is a tradeoff because b_{bias} implies v_{variance} .
- High b_{bias} leads to $o_{\text{overfitting}}$. But controlling for $o_{\text{overfitting}}$ introduces b_{bias} .

Today

- Feature Selection and Dimensionality Reduction

Problem Statement

- Given: A set of examples (\mathbf{x}_i, y_i) over D features
- Do: Find a *subset* S of features ($|S| \leq D$) so that, for any other subset $S' \neq S$, a learned concept that uses S *generalizes better* than a learned concept that uses S'

Issues

- Optimal solution requires a combinatorial optimization
 - Not practical, must be approximated somehow
- This is a bit of a chicken and egg problem

Approach 1: Filter methods

- Decouples feature selection from learning
 - Ignores the chicken and egg problem
- First apply some procedure to prune the feature set
- Then use the selected features to learn a classifier

Feature Pruning procedure

- General approach: “score” each feature using some criterion
- Keep the k highest scoring features

Scoring criterion

- Often use information theoretic scoring criteria
- E.g. mutual information or information gain

$$IG(X) = H(Y) - H(Y | X) = I(X; Y)$$

- Need to be aware of calibration issues as before

Issue

- Just using gain often won't work
- Gain checks *relevance*, but not *redundancy*
- So we need to adjust the criterion to penalize adding multiple correlated features
- In order to account for redundancy, must move to a sequential selection procedure

Redundancy-aware Filtering

- Start with an empty feature set
- At each point, pick a feature that maximizes

$$Score(X | S) = I(X; Y | S)$$

- Where Y is the label and S the currently selected set of features
- Continue until k features selected

Issues

- In general, computing $I(X; Y|S)$ is difficult on a finite sample as S grows
- Also, beyond these criteria, we may want “stable” feature selection methods
 - i.e., the set of chosen features should not change a lot if the data is perturbed a little

Joint Mutual Information

- A good selection criterion that seems to work well in many cases and is practical to compute is the “JMI” criterion (Yang and Moody 99):

$$Score(X | S) = \sum_{X_j \in S} I(X, X_j; Y)$$

- But, as always, no single best technique is possible (ask for paper with detailed empirical study)

Causal Feature Selection

- Often, we want to interpret features as “causes” of the label
 - Leads to greater understanding of the task
- In such cases, we can try to extend the information theoretic approaches to causal approaches, using probabilistic graphical models (CSDS 491/442/600 ask for paper)

Pros and cons

- Filter methods are generally computationally very efficient
 - Often used as a quick and dirty first step
- But may produce suboptimal results
 - Strongly dependent on heuristic scoring function
 - Ignore the “chicken and egg” problem: the learning algorithm that will actually do the classification
 - May require additional ad-hoc assumptions if missing data present

Approach 2: Wrapper methods

- Features need to be selected *in the context of the learner* for best results
- Different features may work well for learners with different inductive biases
- Wrapper methods search through the feature set *using the learner's performance* as a guide

Search space

- The search space consists of all possible subsets of features
- It is intractable to search this exhaustively, so heuristic strategies (e.g. greedy search) are generally employed

Forward Feature Selection

- Set up feature selection as a *search for optimization* problem
- Initial search state: empty feature set
- Search operators: add a single feature
- Scoring function: Generalization error of classifier trained on the new set of features
- Termination condition: Generalization error does not improve

Forward Feature Selection

- $F = \{\}$
- While *generalization error* improves
 - For each feature f not yet added to F
 - $F' = F \cup f$
 - Train a classifier with F' and evaluate it on a validation sample
 - Pick the f that yields the best error metric
 - If this error rate is better than the error rate using F , store F' as the new F

Need validation
set/internal CV

Methodological note

- It is very important to remember that feature selection must be done *within* cross validation
 - i.e. we must not use the test data to select features!
- This means that *different sets of features may be selected in each fold* during learning
- Also means that the evaluation step in Forward FS must be done with an *internal* cross validation loop

Methodological note

- Sometimes the following strategy is employed when deploying a classifier
- First, select features per fold and evaluate as normal
- After evaluation is complete, build a *consensus* feature set (e.g., features that were used by nearly all folds) and retrain on entire dataset
 - This classifier will be used on future data

Backward Feature Selection

- $F = \{\text{All features}\}$
- While generalization error improves
 - For each feature f in F
 - $F' = F \setminus f$
 - Train a classifier with F' and evaluate it on a test sample
 - Remove the f that yields the best error rate
 - If this error rate is better than the error rate using F , store F' as the new F

Boosted feature selection

- Instead of a heuristic search, we can employ *boosting* as a wrapper to select features
- Here, the base learners are generally “decision stumps”: decision trees with a single node
 - Remember this uses Information Gain to pick a feature to classify examples

Boosted feature selection

- The first feature is the one with max IG score
- Because of the way boosting works, the next feature will be the one that is best at classifying *the mistakes of the first feature*
 - The subsequent one will be the best at classifying the mistakes of the first two, etc.
- So each feature provides new, complementary information
 - Good at picking a “diverse” feature set with low redundancy, expected to generalize well
 - Found to work very well in practice

Pros and cons

- Wrapper methods generally provide very good performance because they are paired with a learner
 - Solve the chicken and egg problem iteratively
- Also robust to data issues like missing data (pass on to learner)
- But, need lots of data to work well (multiple cv loops)
- Also generally computationally very expensive (lots of training loops)
- So need a fast learner to work, or some way to heuristically estimate or update the scores
- Boosting provides a very good middle ground in this area

Approach 3: Embedded methods

- Embedded methods alter the objective functions of learning algorithms to *simultaneously* select features and learn the classifier
 - A good way to incorporate the learner into feature selection, but requires modifying learning algorithms

Modifying objectives

- Many learning algorithms optimize a loss function on the training sample:


$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

- Embedded methods add terms to this objective to encourage “sparseness”
 - i.e., encourage many zero w_i ’s

Modifying objectives

- An objective function that does this might be:

$$L(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + \lambda \|\mathbf{w}\|_0$$



The zero-norm is the number of elements in a vector that are non-zero

- This isn't continuous, but it indicates connections to overfitting control

Feature Selection and Overfitting Control

- We also modified learning objectives to control overfitting
- In fact, FS and OC have similar objectives
 - Both are trying to “simplify” the learned model and encourage generalization
 - Selecting a good set of features should help with overfitting control
 - Conversely, a classifier that is robust to overfitting should be using a good set of features