Data, Environment and Society:
Lecture 21: Measuring Classifier Performance

Instructor: Duncan Callaway
GSI: Salma Elmallah

November 12, 2019

Announcements

- Today: Elinor Benami guest lecture
- HW9 due today
- HW10 posts today, due in two weeks
- Thursday: Exam review
- Lab next week: Cancelled, prep for exam!
- Tuesday next week: Exam
    - Covers up to Lecture 19
    - Covers up to HW9

Objectives for today

- Methods for measuring performance of classifiers
- Reminders on parameter, hyperparameter tuning

Remidner: classification error rate

Remidner: classification error rate

The typical error, RSS $= \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ won't work.

Alternatives? Let's start by defining

$$p_{mk} = \text{ fraction of observations belonging to class } k \text{ in region } m.$$

Then a simple measure is:

Classification error rate = how many training observations don't fall into the assigned class.

Within-region this is simply:

$$E_m = 1 - \max_k(\hat{p}_{mk})$$

The confusion matrix

- A “confusion matrix” C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j.
- Numbers on the diagonal of the matrix are correct predictions
- Rows sum to the number of actual observations in a category
- Columns sum to the number of predicted observations in a category

	Predict: True	Predict: False
Actual: True	True pos (TP)	False neg (FN)
Actual: False	False pos (FP)	True neg (TN)

Precision and recall

	Predict: True	Predict: False
Actual: True	True pos (TP)	False neg (FN)
Actual: False	False pos (FP)	True neg (TN)

- **Precision:** Correct number of positives divided by total number of positive predictions. (“True positive rate”)

$$= \frac{TP}{TP + FP}$$

- **Recall:** Correctly number of positives divided by total number of true positives. $(1-$ false positive rate)

$$= \frac{TP}{TP + FN}$$

Reminder / clarification for cross validation: Models have two different classes of parameters

1. Parameters that enter as decision variables for minimizing loss function:

$$\begin{aligned}
\text{Shrinkage: } \beta^* \quad &= \arg\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot R(\beta) \\
\text{CART: } \{j^*, s^*\} \quad &= \arg \min_{j \in J, s \in X_j} \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2
\end{aligned}$$

2. “Hyperparameters”: parameters that constrain how you solve the loss function. These generally prevent overfit.

Reminder / clarification for cross validation: Models have two different classes of parameters

1. Parameters that enter as decision variables for minimizing loss function:

$$\begin{aligned} \text{Shrinkage: } \beta^* &= \arg\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot R(\beta) \\ \text{CART: } \{j^*, s^*\} &= \arg \min_{j\in J, s\in X_j} \sum_{i:x_i\in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i\in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \end{aligned}$$

2. "Hyperparameters": parameters that constrain how you solve the loss function. These generally prevent overfit.
   - λ in shrinkage methods
   - How deep to grow a classification tree?
   - $\sum_{i=1}^{n} \epsilon_i$ in SVM

Ways to minimize the loss function

Ways to minimize the loss function

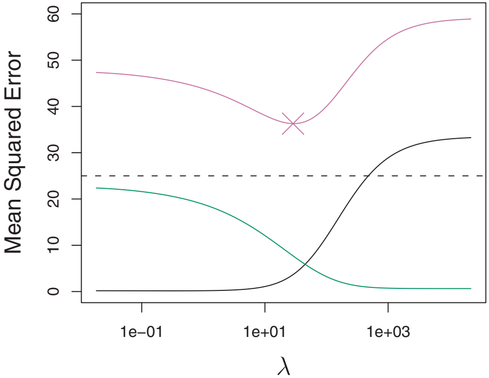
- Closed form solution – e.g. normal equations
- Gradient search

In both cases, we're relying on the condition that the gradient of the loss function approaches zero as we approach the optimal solution

Ways to choose hyperparameters

Ways to choose hyperparameters

- **Grid search**: This is what we've done with shrinkage methods, when there is just one parameter to tune (λ)
- **Randomized parameter search**: This is what you're doing in HW10. Works well when you have lots of hyperparameters to tune.



In both cases, all we're doing is

- Creating a list of candidate hyperparameters (or sets of hyperparameters)
- Training the model (with the training data) for each hyperparameter in the list
- Choosing the hyperparameter with the best cross-validated error.