Contents

1 Introduction to Ensemble Models 2
  1.1 Notation 2
  1.2 Bagging 2

2 Model Averaging and Stacking 8
  2.1 Model Selection 8
  2.2 Model Averaging 8
  2.3 Stacking 9

3 Ensemble Models 14
  3.1 Boosting Preview 14
  3.2 Constructing Ensemble Models 15
1 Introduction to Ensemble Models

Ensemble models combine predictions from several individual models (individual models are also called base learners).

1.1 Notation

- Observed data:
  - \( D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \)
  - There are \( n \) observations
  - Regression: \( y_i \in \mathbb{R} \)
  - Classification: \( y_i \in \mathcal{G} \)

- Base learners (i.e., individual models)
  - \( \hat{g}_1(x), \hat{g}_2(x), \ldots, \hat{g}_M(x) \)
  - There are \( M \) base models

- Ensemble Model
  - \( \hat{f}(x) = F(\hat{g}_1(x), \hat{g}_2(x), \ldots, \hat{g}_M(x)) \)
  - \( F \) is generic notation for methods of combining, aggregating, or using the information from all \( M \) models to make a prediction.

- Summary: Ensemble approaches differ in which base models are used how they are combined

- Benefits:
  - Collective Knowledge of Crowds / Mixture of Experts
  - Bagging: variance reducer
  - Boosting: bias reducer

1.2 Bagging

Bagging fits the same base model to bootstrap samples of the observed data and averages the predictions from each model.

- The base model, \( g(x) \), is usually a tree (or other high variance model)

- The predictions from the base models are averaged:

\[
\hat{f}(x) = \frac{1}{M} \sum_{k=1}^{M} \hat{g}(x \mid D_k^*)
\]

- \( M \) is the number of bootstrap samples (and thus base models)
- \( D_k^* \) is the \( k^{th} \) bootstrap sample
- \( \hat{g}(x \mid D_k^*) \) is the model fit to the \( k^{th} \) bootstrap sample
Bagging Illustration

Bagging

Bootstrap Samples

fit/train models

Combine/aggregate

Original Data

\[ D \]

\[ \hat{g}_1(x) \]

\[ \hat{g}_2(x) \]

\[ \vdots \]

\[ \hat{g}_m(x) \]

\[ f(x) = \frac{1}{M} \sum_{k=1}^{M} \hat{g}_k(x) \]
1.2.1 Bagging Variations

- **Random Forest** models fit trees to bootstrap data, but with the extra de-correlation step of only considering a subset of features for each split.

- **Sub-bagging**: $D_k^*$ is a sub-sample (less than $n$) without replacement

- **Bragging**: use the median instead of the mean to combine predictions

$$\hat{f}(x) = \text{median}(\hat{g}(x \mid D_1^*), \hat{g}(x \mid D_2^*), \ldots, \hat{g}(x \mid D_M^*))$$

- **Bumping**: Like bagging, but choose best model instead of averaging.

$$\hat{f}(x) = \hat{g}(x \mid D_{\text{opt}}^*)$$

  - where $\text{opt} = \arg \min_k \sum_{i=1}^n L(y_i, \hat{g}(x_i \mid D_k^*))$
  - Include the original data $D$ in the comparison
  - Thus, only a single (potentially bagged) dataset is being used for the final model

- **Cross-Validation Committee**: instead of using bootstrap samples, use cross-validation to make the different training sets

$$\hat{f}(x) = \frac{1}{M} \sum_{k=1}^M \hat{g}(x \mid D \setminus D_k)$$

- $D \setminus D_k$ are all the observations not included in the $k^{\text{th}}$ fold.
  - In this notation, there are $M$ folds
  - Special case of sub-bagging!

- The special case of leave-one-out:

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \hat{g}^{-i}(x)$$

- Same tuning parameters used for all base models
10-fold cross-validation sampling
2 Model Averaging and Stacking

The basic idea of model averaging and stacking is simple to represent:

\[ \hat{f}(x) = \sum_{k=1}^{M} \hat{w}_k \hat{g}_k(x) \]

- The estimated weight \( \hat{w}_k \) determines how much the final aggregate model is influenced by model \( \hat{g}_k(x) \)
- Bagging uses equal weights \( w_k = 1/M \) (i.e., nothing is estimated).

2.1 Model Selection

Model selection is the approach of choosing the single best model.

- In this setting, \( \hat{w} = [\hat{w}_1, \ldots, \hat{w}_M] \) is one-hot
  - All \( \hat{w}_k = 0 \), except one is 1.
  - \( \hat{w}_k \in \{0, 1\} \), \( \sum_{k=1}^{M} \hat{w}_k = 1 \)

The best model is selected by: cross-validation (and repeated train/test), AIC/BIC, GCV, LOO-CV, OOB error, etc.

- We have done a type of model selection in choosing the optimal tuning parameters in ridge/lasso/best subsets.

Ensemble Motivation

We may be able to obtain better predictions if we combine all the models instead of just picking the best.

2.2 Model Averaging

Let there be \( M \) candidate models.

- Assume one of the \( M \) models is correct (i.e., one model generated the data)
  - Let \( \mathcal{M} \) denote the true model that generated the data
- Let \( \pi_k = \Pr(\mathcal{M} = k) \) is the prior probability that model \( k \) is the true model
- \( p(D | \mathcal{M} = k) = \int_{\Theta} p(D | \theta_k, \mathcal{M} = k) f(\theta_k | \mathcal{M} = k) d\theta_k \)

The posterior probability of model \( k \) is

\[ \Pr(\mathcal{M} = k | D) = \frac{p(D | \mathcal{M} = k) \cdot \pi_k}{\sum_m p(D | \mathcal{M} = m) \cdot \pi_m} \]

The best prediction (under a squared error loss) is

\[ f^*(x) = \mathbb{E}[Y | X = x, D] \]

\[ = \sum_{k=1}^{M} \Pr(\mathcal{M} = k | D) \cdot \mathbb{E}[Y | X = x, D, \mathcal{M} = k] \]

\[ = \sum_{k=1}^{M} w_k \cdot g_k(x) \]

- \( g_k(x) = \mathbb{E}[Y | X = x, D, \mathcal{M} = k] \) is the best prediction from model \( k \).
- \( w_k = \Pr(\mathcal{M} = k | D) \) is the posterior probability that model \( k \) is the correct model
2.2.1 BIC/AIC

Recall that we considered AIC and BIC for model selection (along with cross-validation).

\[ \text{AIC}(k) = -2 \log L(\hat{\theta}_k) + 2d(k) \]
\[ \text{BIC}(k) = -2 \log L(\hat{\theta}_k) + \log n \cdot d(k) \]

- \( L(\hat{\theta}_k) = \max_{\theta \in \Theta_k} p(D \mid \theta, M = k) \) is the maximized likelihood for model \( k \)
- \( d(k) \) is the effective degrees of freedom for model \( k \) (under MLE)

It turns out that under certain settings (a bit beyond the scope of this course) that BIC is a good estimate of

\[ -2 \log p(D \mid M = k) \approx \log p(D \mid \hat{\theta}_k, M = k) - \log n \cdot d(k)/2 \]

\[ p(D \mid M = k) \approx e^{-\frac{1}{2} \text{BIC}(k)} \]

If we fill this into the posterior and set equal priors (\( \pi_k = 1/M \)) we get

\[ \text{Pr}(M = k \mid D) = \frac{e^{-\frac{1}{2} \text{BIC}(k)}}{\sum_m e^{-\frac{1}{2} \text{BIC}(m)}} \]

Thus for any model where AIC/BIC can be calculated (i.e., there is a likelihood and estimated degrees of freedom) we can use the following ensemble:

\[ \hat{f}(x) = \sum_{k=1}^{M} \hat{w}_k \hat{g}_k(x) \]

- where \( \hat{g}_k(x) \) is the prediction from model \( k \)
- And the weights are:

<table>
<thead>
<tr>
<th>BIC Version</th>
<th>AIC Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{w}_k = \frac{e^{-\frac{1}{2} \text{BIC}(k)}}{\sum_m e^{-\frac{1}{2} \text{BIC}(m)}} )</td>
<td>( \hat{w}_k = \frac{e^{-\frac{1}{2} \text{AIC}(k)}}{\sum_m e^{-\frac{1}{2} \text{AIC}(m)}} )</td>
</tr>
</tbody>
</table>

2.3 Stacking

A stacking model combines base models as a weighted sum

\[ \hat{f}(x) = \sum_{k=1}^{M} \hat{w}_k \hat{g}_k(x) \]

- Strictly speaking, this is a bit more general that model averaging as the weights aren’t constrained to sum to 1 or even be non-negative. (Although it is essentially the same idea.)
- Stacking is popular in prediction contests as it is a great way to combine models from teammates
- Notice that each model \( \hat{g}_k(x) \) and the weights \( \hat{w} = (\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_M) \) must be estimated.
Your Turn #1

1. In the best subsets and step-wise approaches, model \( \hat{g}_k(x) \) is the best linear model with \( k \) predictors. What are the optimal weights if selected according to least squares (using the training data):

\[
\hat{w} = \arg \min_w \sum_{i=1}^{n} (y_i - \sum_{k=1}^{M} w_k \hat{g}_k(x_i))^2
\]

2. In lasso/ridge regression, model \( \hat{g}_k(x) \) is the model corresponding to \( \lambda_k \). What are the optimal weights if selected according to least squares (using the training data):

\[
\hat{w} = \arg \min_w \sum_{i=1}^{n} (y_i - \sum_{k=1}^{M} w_k \hat{g}_k(x_i))^2
\]

3. What would be a better way to select \( \hat{w} \)?
2.3.1 Stacking (Single Hold-Out)

The main idea behind stacking is to find the weights using out-of-sample predictions.

**Algorithm: Single Hold-out Stacking**

1. Partition the data into a training and testing set $D = [D_{\text{train}}, D_{\text{test}}]$
2. Fit each model with the data from the training set and make predictions for the data in test set
   - Let $\hat{g}_k(x_i \mid D_{\text{train}})$ denote the prediction for test observation $i$ using the training data $D_{\text{train}}$
3. The optimal weights are selected as:
   \[
   \hat{w} = \arg\min_w \sum_{i \in D_{\text{test}}} L \left( y_i, \sum_{k=1}^M w_k \hat{g}_k(x_i \mid D_{\text{train}}) \right)
   \]
4. (optional) The final prediction is made by fitting each model with all the data
   \[
   \hat{f}(x) = \sum_{k=1}^M \hat{w}_k \hat{g}_k(x \mid D)
   \]
   - Note: cross-validation is only used to estimate the weights

Because the prediction $\hat{y}_i|x_i$ is made from models that aren’t trained with $(x_i, y_i)$, the stacking weights are fairly adjusted for different model complexities.

- E.g., a model that is too complex (overfits) will not make good estimates on the hold-out data and hence should receive a low weight

**Stacking Features**

Another way to view stacking is that each model creates a set of new features (feature engineering):

$Z_{ik} = \hat{g}_k(x_i \mid D_{\text{train}})$

- where $x_i \in D_{\text{test}}$

and uses a simple model (e.g., linear regression or logistic regression) to estimate the weights:

\[
\hat{w} = \arg\min_w \sum_{i \in D_{\text{test}}} L \left( y_i, \sum_{k=1}^M w_k Z_{ik} \right)
\]

- E.g., using linear regression, $\hat{w} = (Z^T Z)^{-1} Z^T Y$
- Note: we could also use constrained optimization to force the weights to be non-negative and sum to one (model averaging)
- Add in the original feature $X_{\text{test}}$ as interactions if certain regions features space are better predicted by certain models
  - i.e., weights vary over feature space
  - trees may be good ensemble models for this if there is sufficient hold-out data.
2.3.2 Stacking (Cross-validated)

The main idea behind stacking is to find the weights using out-of-sample predictions.

**Algorithm: Cross-Validated Stacking**

1. Partition the data into $V$-folds ($D_1, D_2, \ldots, D_V$)
2. Fit each model with the data from all folds except fold $v$ and make predictions for the data in fold $v$
   - Repeat for all $V$ folds
   - Let $\hat{g}_k(x_i \mid D \setminus D_{v_i})$ denote the prediction for observation $i$ using all the data except the data in the same fold as $i$ (i.e., $D_{v_i}$ is the data in the same fold as observation $i$)
3. The optimal weights are selected as:
   $$\hat{w} = \arg \min_w \sum_{i=1}^{n} L \left( y_i, \sum_{k=1}^{M} w_k \hat{g}_k(x_i \mid D \setminus D_{v_i}) \right)$$
4. (optional) The final prediction is made by fitting each model with *all* the data
   $$\hat{f}(x) = \sum_{k=1}^{M} \hat{w}_k \hat{g}_k(x \mid D)$$
   - Note: cross-validation is only used to estimate the weights
Because the prediction $\hat{y}_i|x_i$ is made from models that aren’t trained with $(x_i, y_i)$, the stacking weights are fairly adjusted for different model complexities.

- E.g., a model that is too complex (overfits) will not make good estimates on the hold-out data and hence should receive a low weight.

### Stacking Features

Another way to view stacking is that each model creates a set of new features (feature engineering):

$$Z_{ik} = \hat{g}_k(x_i \mid D \setminus D_{ei})$$

and uses a simple model (e.g., linear regression or logistic regression) to estimate the weights:

$$\hat{w} = \arg \min_w \sum_{i=1}^{n} L(y_i, \sum_{k=1}^{M} w_k Z_{ik})$$

- E.g., using linear regression, $\hat{w} = (Z^T Z)^{-1} Z^T Y$
- Note: we could also use constrained optimization to force the weights to be non-negative and sum to one (model averaging)

### Stacking Illustration

- [Diagram or image related to stacking features]

- [Diagram or image related to stacking illustration]
3 Ensemble Models

3.1 Boosting Preview

So far, we have focused on fitting the base models in *parallel*. In boosting, the base models are fit *sequentially*.

The general idea of *boosting* is to fit models sequentially, where each model depends on the combination of previous models.

There are two primary approaches:

1. Gradient Boosting: Sequentially fit models to the (pseudo) residuals, where the residuals are larger for observations that are poorly predicted.

2. AdaBoost: Sequentially fit to re-weighted data, where the weights are larger for observations that are poorly predicted.

Boosting is primarily a bias reducer

- The base models are often simple/weak (low variance, but high bias) models (like shallow trees)
3.2 Constructing Ensemble Models

Ensemble methods differ on (i) which base models are included and (ii) how the base models are combined to form a final prediction.

Here are a few thoughts on different ensemble configurations

- Think about how these impact the overall bias and variance (including model correlation) trade-off

3.2.1 Fitting Base Learners

1. Use same base learners (with different data/initialization) or different base learners.
   - Bagging and RF uses the same base learners, but fit with different (bootstraped) data
   - Better predictions may be achieved by using very different base learners (e.g., random forest, xgboost, GAM, ANN)

2. Use different data to train each model.
   - Bagging/RF uses bootstrap data to build different models
   - Boosting sequentially uses re-weighted or modified/residuals data

3. Use different sets of features to train each model.
   - RF randomly selects a sub-set of features for making each split
   - Has the potential to decrease correlation between base learners

4. Use different transformations of outcome variable to build models.
   - E.g., fit models to $y$ and also $y' = \log y$ (and then backtransform)
   - E.g., one-vs-rest for classification
   - Gradient Boosting sequentially fits models to current residuals

5. Use randomness in model fitting.
   - use different initializations
   - RF uses random subset of features for each split
   - Average multiple stochastic models (of same family/tuning) from different seeds

3.2.2 Combining Models/Predictors

The base models can be combined in many different ways

1. Weighted sum/average
   - Model Averaging
   - Stacking

2. Choose the best one
   - Model Selection
   - (all weights are zero except 1 weight is one)

3. Use the median prediction (Bragging)
4. Parallel vs. Sequential

- Bagging is parallel
- Boosting is sequential