Model Combination

Tong Zhang

Rutgers University

Standard model for supervised learning

- Goal: want to predict Y based on X
  - prediction function $f(x)$
- Data: (X, Y) are randomly drawn from an underlying distribution D
- Training data: iid samples from D:

$$S_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$$

- Learning: construct prediction function f from training data
  - want to minimize future loss over D:

$$\mathbf{E}_{(X,Y)\sim D} L(f(X), Y)$$

Learning Algorithm

- Learning algorithm $\mathcal{A}$
  - given training data $S_n = \{(X_i, Y_i)\}_{i=1,\ldots,n}$.
  - learn prediction rule $\hat{f} = \mathcal{A}(S_n)$ from S_n
- Examples: linear logistic regression, SVM, ridge regression, etc
  - learning linear prediction $f(x) = \beta^T x$ from training data.

Learning Algorithm

- Learning algorithm $\mathcal{A}$
  - given training data $S_n = \{(X_i, Y_i)\}_{i=1,\ldots,n}$.
  - learn prediction rule $\hat{f} = \mathcal{A}(S_n)$ from S_n
- Examples: linear logistic regression, SVM, ridge regression, etc
  - learning linear prediction $f(x) = \beta^T x$ from training data.
- Model selection:
  - many algorithms, which one to choose?
  - goal: good interpretability and/or better performance

Learning Algorithm

- Learning algorithm $\mathcal{A}$
  - given training data $S_n = \{(X_i, Y_i)\}_{i=1,\ldots,n}$.
  - learn prediction rule $\hat{f} = \mathcal{A}(S_n)$ from S_n
- Examples: linear logistic regression, SVM, ridge regression, etc
  - learning linear prediction $f(x) = \beta^T x$ from training data.
- Model selection:
  - many algorithms, which one to choose?
  - goal: good interpretability and/or better performance
- Model averaging/combination:
  - many algorithms, how to combine?
  - goal: better performance

Model Combination

Also called ensemble or blending or averaging

- Equally weighted averaging (voting).
- Exponentially weighted model averaging.
- Weight optimization using stacking.
- Bagging.
- Additive model and boosting

Equally weighted averaging (voting)

- Models $\{M_j\}_{j=1,\ldots,L}$; prediction functions $\hat{f}_j(x)$ from training data.
- The voting ensemble is

$$\bar{f}(x) = \frac{1}{L}\sum_{j=1}^{L} \hat{f}_j(x).$$

- For convex loss functions: reducing model estimation variance.
- Least squares:

$$\underbrace{\mathbf{E}_{X,Y}\left(\frac{1}{L}\sum_{j=1}^{L}\hat{f}_j(X) - Y\right)^2}_{\text{ensemble performance}} = \underbrace{\frac{1}{L}\sum_{j=1}^{L}\mathbf{E}_{X,Y}(\hat{f}_j(X) - Y)^2}_{\text{average single model performance}}$$

$$\underbrace{- \frac{1}{L}\sum_{j=1}^{L}\mathbf{E}_{X,Y}(\hat{f}_j(X) - \bar{f}(X))^2}_{\text{model diversity}}$$

Model Selection versus Averaging

- Model selection: works better if one model is significantly more accurate than other models
  - no ambiguity of which single model is better
- Equally weighted averaging: works better if all models have similar prediction accuracy, but are different
  - some ambiguity of which single model is better

Model Selection versus Averaging

- Model selection: works better if one model is significantly more accurate than other models
  - no ambiguity of which single model is better
- Equally weighted averaging: works better if all models have similar prediction accuracy, but are different
  - some ambiguity of which single model is better
- Key to the success of model ensemble (averaging):
  - all models are reasonably high performance
  - models are diverse (they have different predictions)

Ideas to Generate Diverse Models

In competitions, to improve performance, one often creates a diverse set of models with similar performance.

How to do so?

Ideas to Generate Diverse Models

In competitions, to improve performance, one often creates a diverse set of models with similar performance.

How to do so?

- different learning algorithms
- tuning parameter variation
- different variations of features
- randomization (e.g. bagging)

Average diverse models nearly always help a little

- related reason: when models are ambiguous, model combination is more stable than model selection

Example

- Winning strategy in a recent predictive modeling competition over more than 200 teams.
- Performance is measured by a certain error metric: the smaller the better – test data size: $60,000$
- Ensemble: generate 62 models using randomization and variations of ensemble trees

Example

- Winning strategy in a recent predictive modeling competition over more than 200 teams.
- Performance is measured by a certain error metric: the smaller the better – test data size: $60,000$
- Ensemble: generate 62 models using randomization and variations of ensemble trees
- Test error:
  - single best: 0.688 (the performance already wins the competition)
  - combination of five models: 0.685 (0.4% error reduction)
  - combination of thirty models: 0.682 (0.9% error reduction)
  - combination of sixty-two models: 0.680 (1.2% error reduction)
- Larger gains on some other data: create a more diverse ensemble

Exponentially Weighted Model Averaging

- Observe data (X_i, Y_i) and want to find a combination to minimize a certain error $\text{ERROR}(f)$
- Consider model averaging with model outputs $\hat{f}_1(x), \ldots, \hat{f}_L(x)$, and

$$\hat{f}(x) = w_1 \hat{f}_1(x) + \cdots w_L \hat{f}_L(x)$$

- Exponentially weighted model averaging:

$$w_j = \frac{e^{-\lambda \, \text{ERROR}(\hat{f}_j)}}{\sum_{i=1}^{L} e^{-\lambda \, \text{ERROR}(\hat{f}_j)}}$$

- Give larger weight to smaller error (error can be either on training or validation data)
  - $\lambda = \infty$: model selection (find the one with smallest error)
  - $\lambda = 0$: equally weighted model averaging
- Good theoretical property; related to Bayesian model averaging

Stacking

- Split data into training + validation.
- L models $\hat{f}_j(x)$ obtained from training data.
- On validation data define VALIDATION ERROR$(\hat{f}_j)$, find combination:

$$f(x) = \sum_{j=1}^{L} w_j \hat{f}_j(x), \quad w_j \geq 0, \sum_j w_j = 1$$

to minimize the validation error:

$$\min_w \left[\text{VALIDATION ERROR} \left(\sum_{j=1}^{L} w_j \hat{f}_j \right) \right].$$

more complex than exponentially weighted model averaging

Stacking

- Split data into training + validation.
- L models $\hat{f}_j(x)$ obtained from training data.
- On validation data define VALIDATION ERROR$(\hat{f}_j)$, find combination:

$$f(x) = \sum_{j=1}^{L} w_j \hat{f}_j(x), \quad w_j \geq 0, \sum_j w_j = 1$$

to minimize the validation error:

$$\min_w \left[\text{VALIDATION ERROR} \left(\sum_{j=1}^{L} w_j \hat{f}_j \right) \right].$$

more complex than exponentially weighted model averaging

- More generally: treat $\{\hat{f}_j(x)\}$ as features and use any learning algorithm to train a combination model with features $\{\hat{f}_j(x)\}$ on validation data.

Example

- Winning strategy of an old competition: CoNLL 2003 English named entity recognition.
- Performance measured by F-measure (similar to accuracy): the higher the better
- Training four classifiers using different algorithms
- Best single: 89.94
- Equally weighted averaging: 91.56
- Stacking: training a linear combination of the classifiers: 91.63

Generating Diverse Ensemble: Bagging

- How to generate ensemble candidates from a single learning algorithm $\mathcal{A}$?

Generating Diverse Ensemble: Bagging

- How to generate ensemble candidates from a single learning algorithm $\mathcal{A}$?
- Answer: bagging
  - build $\hat{f}_i$ from bootstrapped training samples:
    - drawn n samples $\hat{S}_n^i$ from training data S_n, with replacement.
    - $\hat{f}_i = \mathcal{A}(\hat{S}_n^i)$.
  - form voted classifiers: $\hat{f} = \frac{1}{L}\sum_{i=1}^{L} \hat{f}_i$.

Generating Diverse Ensemble: Bagging

- How to generate ensemble candidates from a single learning algorithm $\mathcal{A}$?
- Answer: bagging
  - build $\hat{f}_i$ from bootstrapped training samples:
    - drawn n samples $\hat{S}_n^i$ from training data S_n, with replacement.
    - $\hat{f}_i = \mathcal{A}(\hat{S}_n^i)$.
  - form voted classifiers: $\hat{f} = \frac{1}{L}\sum_{i=1}^{L} \hat{f}_i$.
- When does it help.
  - unstable classifier: $\hat{f}_i$ very different from each other.
  - performance of $\hat{f}_i$ are similar.

Generating Diverse Ensemble: Bagging

- How to generate ensemble candidates from a single learning algorithm $\mathcal{A}$?
- Answer: bagging
  - build $\hat{f}_i$ from bootstrapped training samples:
    - drawn n samples $\hat{S}_n^i$ from training data S_n, with replacement.
    - $\hat{f}_i = \mathcal{A}(\hat{S}_n^i)$.
  - form voted classifiers: $\hat{f} = \frac{1}{L}\sum_{i=1}^{L} \hat{f}_i$.
- When does it help.
  - unstable classifier: $\hat{f}_i$ very different from each other.
  - performance of $\hat{f}_i$ are similar.
- How does it help.
  - variance reduction: ensemble is more stable although each classifier uses less data.

Other Ideas to Generate Ensemble Candidates

- Use boosting: can be either regarded as a single model or as an ensemble method

Other Ideas to Generate Ensemble Candidates

- Use boosting: can be either regarded as a single model or as an ensemble method
- Use different learning algorithms (different loss functions, regularizations)

Other Ideas to Generate Ensemble Candidates

- Use boosting: can be either regarded as a single model or as an ensemble method
- Use different learning algorithms (different loss functions, regularizations)
- Use different variations of data-processing
  - different transformations of features
  - different subsets (selection) of features – unstable so choose from multiple ensemble helps

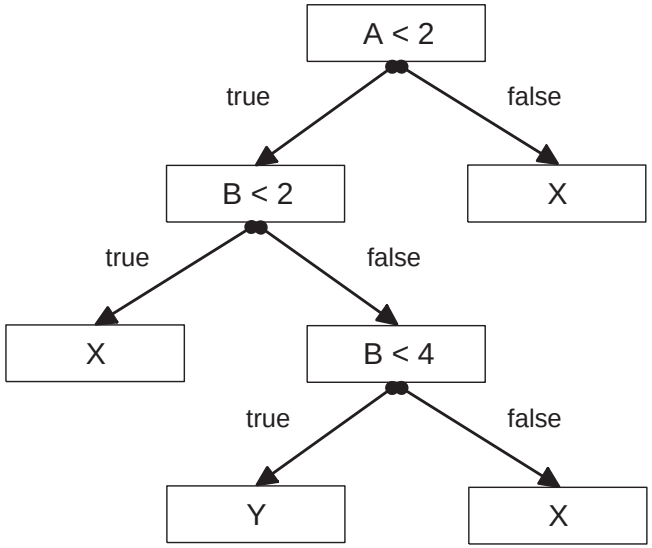
Other Ideas to Generate Ensemble Candidates

- Use boosting: can be either regarded as a single model or as an ensemble method
- Use different learning algorithms (different loss functions, regularizations)
- Use different variations of data-processing
  - different transformations of features
  - different subsets (selection) of features – unstable so choose from multiple ensemble helps
- Use different tuning parameter configurations:
  - e.g. varying λ for ridge regression or K for nearest neighbor

Other Ideas to Generate Ensemble Candidates

- Use boosting: can be either regarded as a single model or as an ensemble method
- Use different learning algorithms (different loss functions, regularizations)
- Use different variations of data-processing
  - different transformations of features
  - different subsets (selection) of features – unstable so choose from multiple ensemble helps
- Use different tuning parameter configurations:
  - e.g. varying λ for ridge regression or K for nearest neighbor
- Use randomization
  - try to randomly select training data (bagging)
  - try to randomly select features or randomly select basis functions
  - try to randomization the algorithm especially if it contains some combinatoric elements (e.g. random forest for decision tree building)

Decision Tree



- *A* and *B*: features
- *X* and *Y*: predicted class labels
- Training: greedily test features to split at each node

Remarks on Decision Tree

- Advantages:
  - interpretable
  - handle non-homogeneous features easily
  - finds non-linear interactions
- Disadvantage:
  - usually not the most accurate classifier by itself

Improving Decision Tree Performance

Improve accuracy through tree ensemble learning:

- bagging
  - generate bootstrap samples.
  - train one tree per bootstrap sample.
  - take equally weighted average of the trees.
- random forest
  - bagging with additional randomization.
- boosting

Bagging and Random Forest

- Bagging
  - build $\hat{f}_i$ from bootstrapped training samples:
  - Form voted classifiers: $\hat{f} = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i$.

Bagging and Random Forest

- Bagging
  - build $\hat{f}_i$ from bootstrapped training samples:
  - Form voted classifiers: $\hat{f} = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i$.
- Random forest
  - generate bootstrap samples.
  - build one tree per bootstrap sample
  - increase diversity via additional randomization: randomly pick a subset of features to split at each node
  - take equally weighted average of the trees.

Why Bagging and RF work

- Tree unstable, thus different random trees are very different
- May generate deep trees, overfitting the data with each tree
- Equally weighted averaging reduces the variance.
- Related to Bayesian model averaging over all possible trees.

Summary

- Model selection: find the best single model
  - better interpretability but may be unstable under ambiguity
- Model averaging/combination:
  - find the best combination of models
  - various methods

Summary

- Model selection: find the best single model
  - better interpretability but may be unstable under ambiguity
- Model averaging/combination:
  - find the best combination of models
  - various methods
- Model averaging is often more effective than model selection
  - ensemble can improve stability: ensemble is stable even if individual models are not
- Key issue in model averaging: generate diverse models with good performance