CS5841/EE5841 Machine Learning

Lecture 7: Ensemble methods

Evan Lucas

Overview

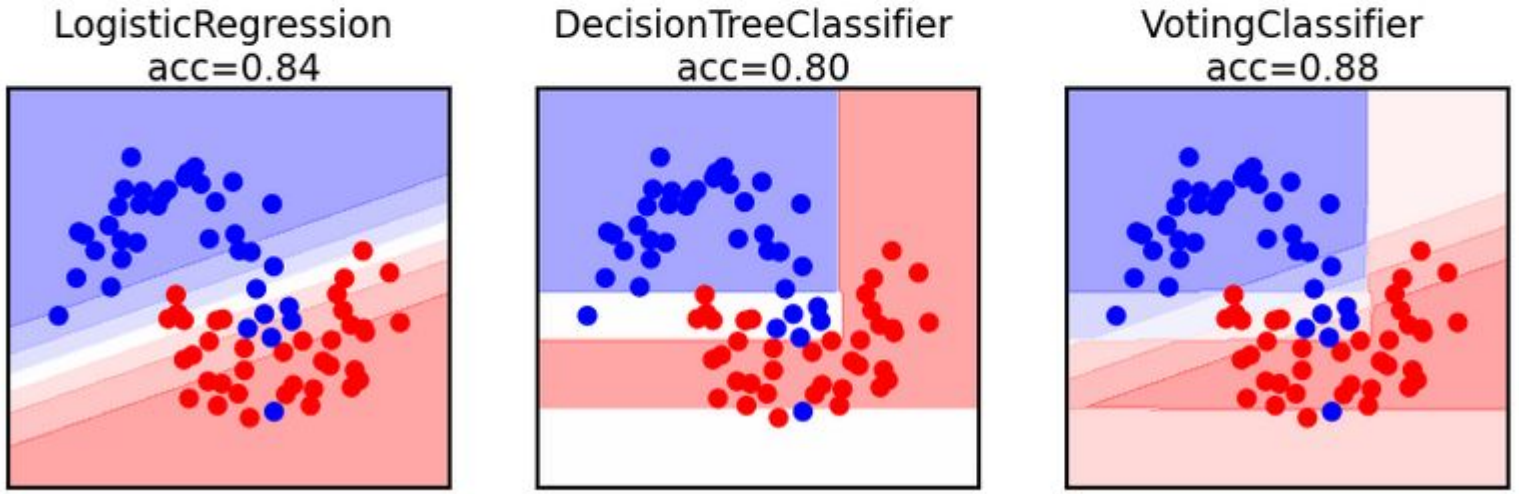
- Course updates
- Ensemble Methods

Class updates

- Course zoom link on Canvas page

Ensemble learning

- If different models make different mistakes, can we simply average the predictions?
- Voting Classifier: gives every model a *vote* on the class label
  - Hard vote: majority class wins (class order breaks ties)
  - Soft vote: sum class probabilities $p_{m,c}$ over M models: $\underset{c}{\operatorname{argmax}} \sum_{m=1}^{M} w_c p_{m,c}$
  - Classes can get different weights w_c (default: $w_c = 1$)



Why does this work?

- Different models may be good at identifying utility of different features of data

Bias-variance approach to understanding model combining

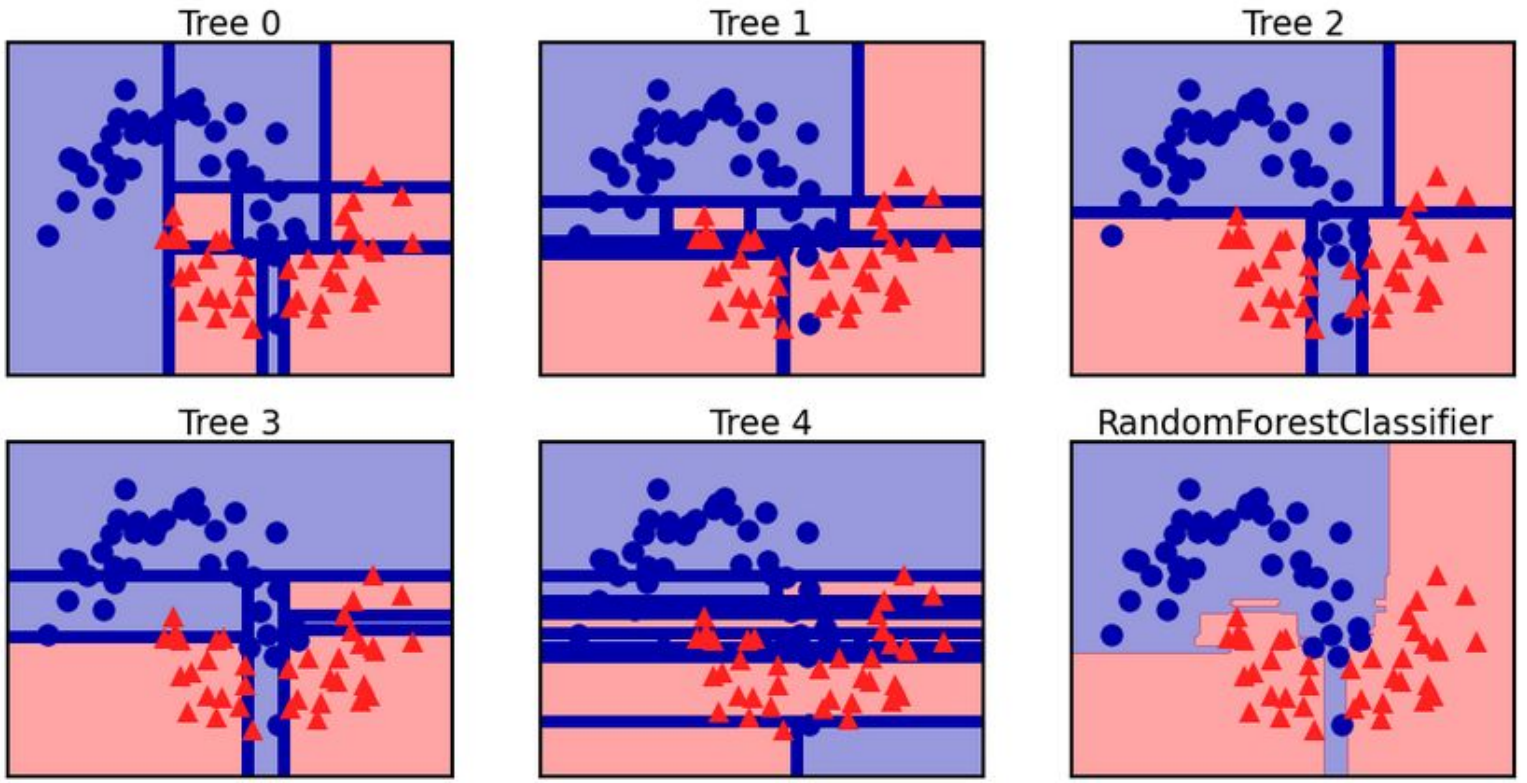
- If model underfits (high bias, low variance): combine with other low-variance models
  - Need to be different, consider them to be 'experts' on different aspects of feature space
  - Reduces bias
  - **Boosting** - technique for combining these models
- If model overfits (low bias, high variance): combine with other low-bias models
  - Need to be different, mistakes have to be different
  - Reduces variance
  - **Bagging** - how we combine these models
- **Stacking** - learned method of combining predictions

Bagging (bootstrap aggregating)

- Build different models
  - Train same model architecture on different training samples
  - Base models *should* be unstable
- Predict by averaging predictions of multiple base models
  - Soft voting for classification
  - Mean value for regression
- Random forests - common example of bagging

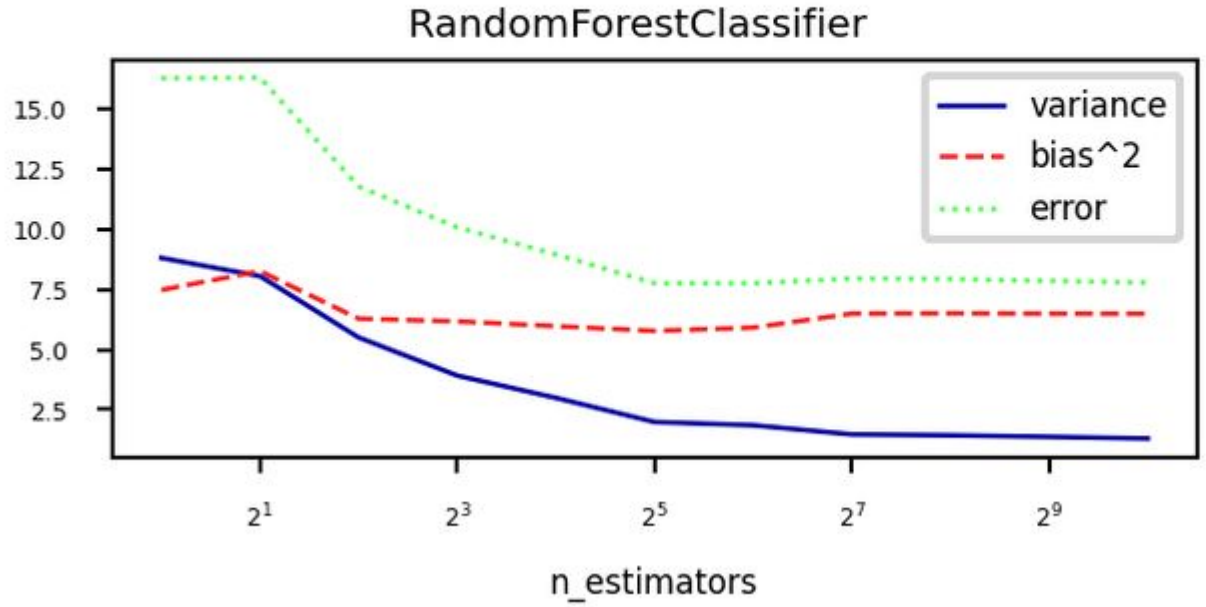
Random Forests

- Uses *randomized trees* to make models even less correlated (more unstable)
  - At every split, only consider `max_features` features, randomly selected
- Extremely randomized trees: considers 1 random threshold for random set of features (faster)



Effect on bias and variance

- Increasing the number of models (trees) decreases variance (less overfitting)
- Bias is mostly unaffected, but will increase if the forest becomes too large (oversmoothing)



Strength and weaknesses

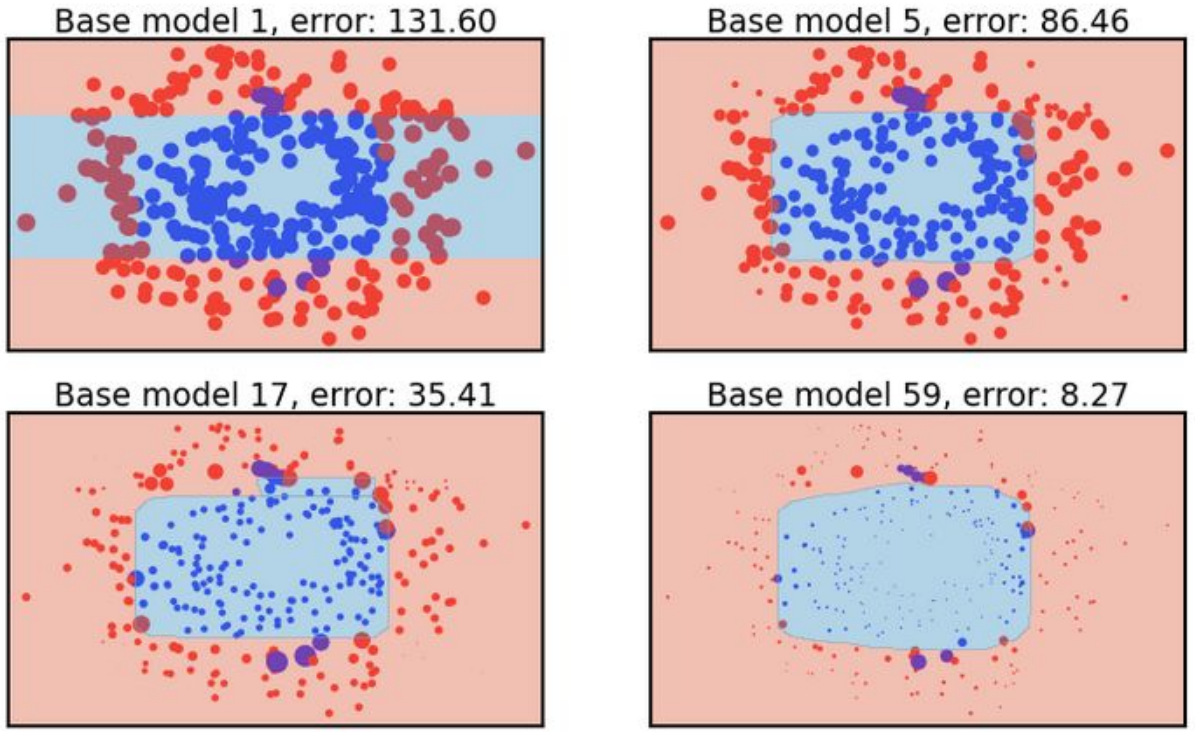
- RandomForest are among most widely used algorithms:
    - Don't require a lot of tuning
    - Typically very accurate
    - Handles heterogeneous features well (trees)
    - Implictly selects most relevant features
- Downsides:
    - less interpretable, slower to train (but parallellizable)
    - don't work well on high dimensional sparse data (e.g. text)

Boosting

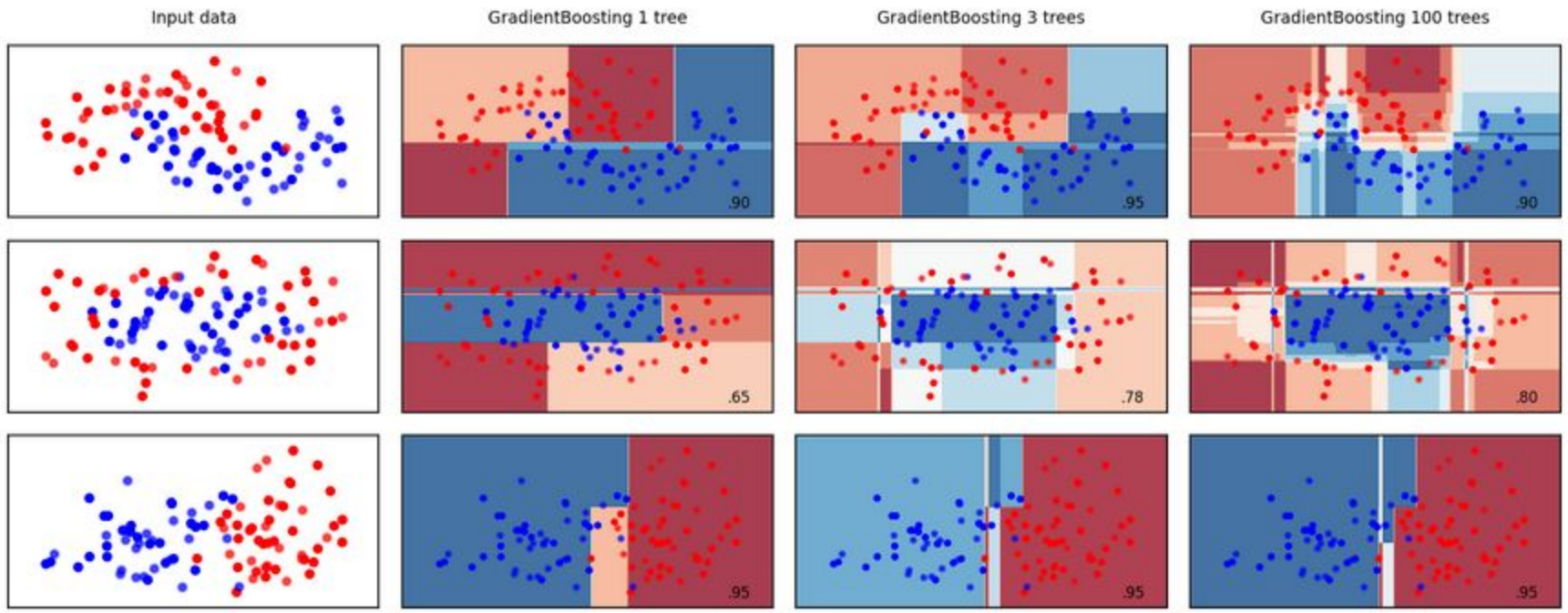
- Weak learners are combined in series
- Four main methods
  - AdaBoost (adaptive boosting) - reweights data points iteratively to weight incorrectly classified points higher for next classifier in sequence
  - Gradient Boosting (GB) - optimizes loss function to create a new base model that is better than the past one
  - XGBoost (extreme gradient boosting) GB that's been parallelized
  - Others
    - LightGBM - gradient based sampling
    - CatBoost - for categorical variables

GB in action

- Size of the samples represents the residual weights: most quickly drop to (near) zero

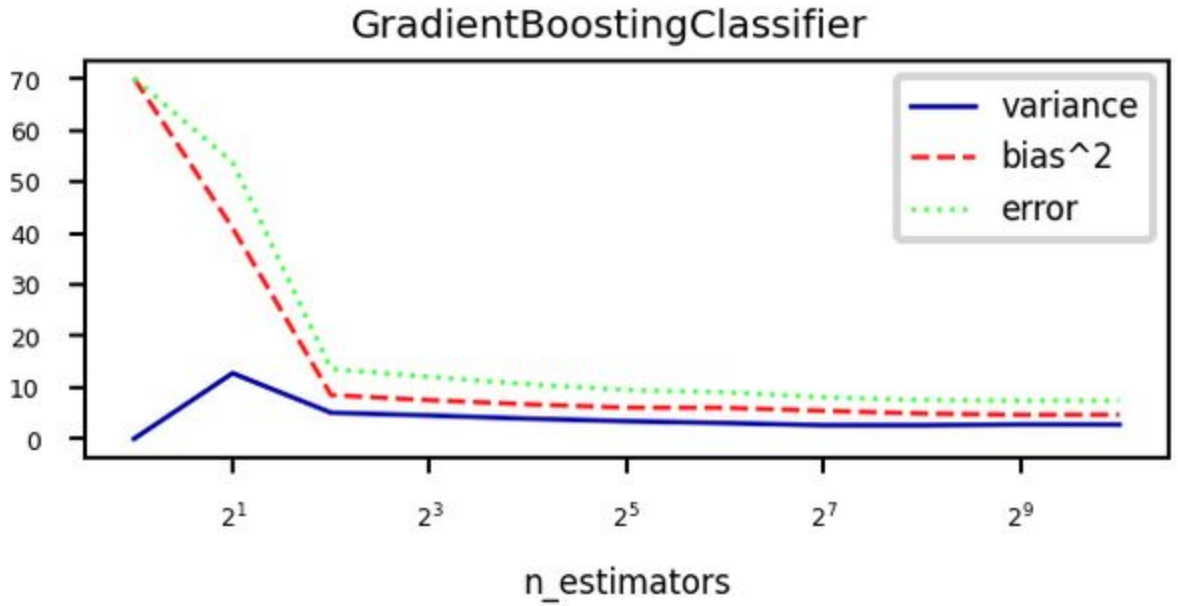


Examples


Input data
GradientBoosting 1 tree
GradientBoosting 3 trees
GradientBoosting 100 trees
.90
.95
.90
.65
.78
.80
.95
.95
.95

Bias-variance analysis

- Gradient Boosting is very effective at reducing bias error
- Boosting too much will eventually increase variance



Gradient Boosting: strengths and weaknesses

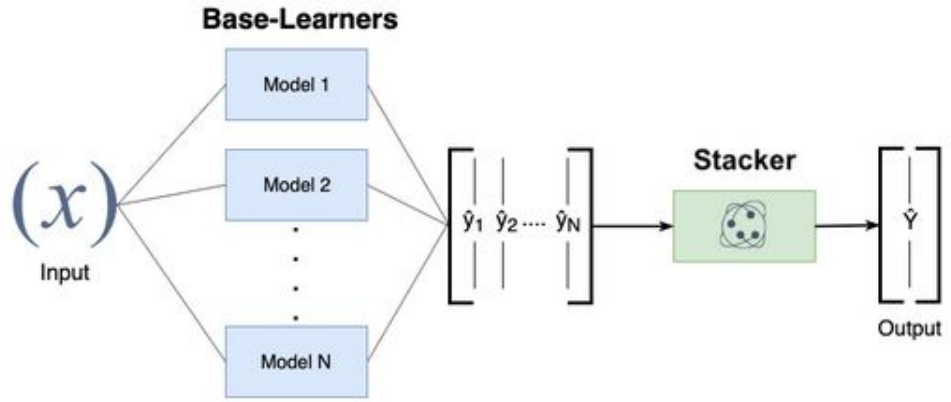
- Among the most powerful and widely used models
- Work well on heterogeneous features and different scales
- Typically better than random forests, but requires more tuning, longer training
- Does not work well on high-dimensional sparse data

Main hyperparameters:

- `n_estimators` : Higher is better, but will start to overfit
- `learning_rate` : Lower rates mean more trees are needed to get more complex models
    - Set `n_estimators` as high as possible, then tune `learning_rate`
    - Or, choose a `learning_rate` and use early stopping to avoid overfitting
- `max_depth` : typically kept low (<5), reduce when overfitting
- `max_features` : can also be tuned, similar to random forests
- `n_iter_no_change` : early stopping: algorithm stops if improvement is less than a certain tolerance `tol` for more than `n_iter_no_change` iterations.

Stacking

- Choose M different base-models, generate predictions
- Stacker (meta-model) learns mapping between predictions and correct label
    - Can also be repeated: multi-level stacking
    - Popular stackers: linear models (fast) and gradient boosting (accurate)
- Cascade stacking: adds base-model predictions as extra features
- Models need to be sufficiently different, be experts at different parts of the data
- Can be *very* accurate, but also very slow to predict



Other ensembling methods

- Hyper-ensembles: same basic model but with different hyperparameter settings
    - Can combine overfitted and underfitted models
- Deep ensembles: ensembles of deep learning models
- Bayes optimal classifier: ensemble of all possible models (largely theoretic)
- Bayesian model averaging: weighted average of probabilistic models, weighted by their posteri probabilities
- Cross-validation selection: does internal cross-validation to select best of M models
- Any combination of different ensembling techniques

Questions + Comments?

Slide source

https://ml-course.github.io/master/05%20-%20Ensemble%20Learning.slides.html