Ensemble Methods

CV_M12_L01
Ensemble Methods

CS/DSA 5970: Machine Learning Practice
Back to Decision Trees …

- Simple learning algorithm(s)
- Both classification and regression forms
- Classification models easily handle multiple classes
- Models can be intuitive for human experts
  - Naturally give us a sense of the most important features
Decision Tree Challenges

• Splits are most often based on individual features
• Crisp region boundaries
  – Most common regression architecture: end up with a piece-wise constant function (so, it is discontinuous)
• Deep trees are necessary to capture complex models
• Deeper models:
  – > Fewer samples in the leaf nodes
  – > Brittle when it comes to generalization
Sir Francis Galton (1822-1911)

- Meteorology: first weather maps
- Statistics: regression
- Psychology
- Heredity
- …
Weighing a Cow
Weighing a Cow

• Individually, non-experts are generally not good at guessing the weight of a cow
• However, the distribution is ~Normal, with a mean very close to the true weight

Message: Measures from a large set of independent, poor-quality predictors can give us a high-quality prediction
Mixing Many Imperfect “Experts”

Ensemble-based methods:
• Create many models
• Combine the predictions of these models
  – Classifiers: voting
  – Regression: some mechanism for blending the predictions
    (e.g., computing a mean)
Example: Breast Cancer Classification

Levenson et al. (2015), PLOS One
Breast Cancer Classification

Levenson et al:

• Trained individuals to label images of tumors as either malignant or benign
• After 2 weeks, these individuals could classify the images with an accuracy of 85%
• Hard voting classifier: the votes across the individuals were tallied
• Accuracy increased to 99%!
Breast Cancer Classification

Hard voting classifier:

• This improvement in performance requires independence of the individuals

• The law of large numbers: combining a large number of independent random variable samples gives us the correct answer with high probability

• And, the individuals in this case were pigeons…
Ensemble Predictions

• Set of trained classifiers
• Can be different types of classifiers: decision tree, logistic regression, support vector machine…
  – Different model types often capture different trends in the training set
• Combine the labels from the classifiers:
  – Hard voting: crisp answers are counted across the ensemble
  – Soft voting: average class probabilities & select the highest one
Example: Voting Classifier

CV_M12_L02
Example: Voting Classifier

CS/DSA 5970: Machine Learning Practice
sklearn.ensemble: VotingClassifier

• Constructor:
  – List of classifiers
    • We have generally already chosen hyperparameters
  – Hard or soft voting
    • Soft voting requires predict_proba() to be available

• fit() will fit each model in sequence
• predict() will query all models and combine the results
Live demo
Bagging and Pasting

CV_M12_L03
Bagging and Pasting

CS/DSA 5970: Machine Learning Practice
Ensemble Methods

• Success of ensemble methods relies on independence of the individual models
• Can we achieve this if the models are all of the same type?
Forcing Independence

• Train each model instance with a subsample of the training set:
  – Pasting: sample without replacement
  – Bagging (bootstrap aggregation): sample with replacement

• Models can be trained in parallel
Forcing Independence

• Pasting: sample without replacement
  – All ensemble members have different training data
  – Effective training sets may not be large enough

• Bagging (bootstrap aggregation): sample with replacement
  – A single training sample may be used by multiple ensemble members -> less independence
  – But, allows us to have larger training sets
Forcing Independence

After training, a new query is addressed by asking each model to provide an answer

• Classifier: voting
• Regression: average the predictions of the individual models
Example: Bagging Classifier

CV_M12_L04
Example: Bagging Classifier

CS/DSA 5970: Machine Learning Practice
Live demo
Random Subspaces

CV_M12_L05
Random Subspaces

CS/DSA 5970: Machine Learning Practice
Forcing Independence

• So far:
  – Bagging & pasting take random subsets of the training data
  – These are Random Patches of the data

• Sampling features:
  – Random Subspaces: only use a subset of the available features for a given model
  – Support for this also in BaggingClassifier
Live demo
Random Forests

CV_M12_L06
Random Forests

CS/DSA 5970: Machine Learning Practice
Random Forests

Ensemble of Decision Trees
• Can continue to use the Bagging Classifier
• RandomForestClassifier class does the same thing, but is optimized for classifying with decision trees
  – Hyper-parameters for this class include Decision Tree hyper-parameters and the ensemble hyper-parameters
• RandomForestRegressor also optimized for ensemble of regression trees
Forcing Independence

Adding noise to tree construction. For each possible split:

• Random forest: consider only a small subset of the available features
  – This is the Random Subspaces idea!
  – Particularly useful when there are many features possible or many possible questions

• Extra trees: consider only a subset of possible thresholds (or question parameters)
  – ExtraTreesClassifier class
  – Reduces search during each leaf node split
Live demo
Ensemble Methods

• Allow us to combine many *weak learners*
  – Each does not have to perform very well
  – The ensemble model often performs better than the weak learners

• Bigger implications:
  – We can specifically choose simpler models (e.g., trees that are heavily regularized)
  – Cheaper to compute and leaf node predictions are based on a larger number of samples (compared to deeper trees)
Feature Importance

CV_M12_L07
Feature Importance

CS/DSA 5970: Machine Learning Practice
Feature Importance

• Feature Importance:
  – Which of our input features are useful in constructing our models?

• Getting this right can:
  – Help domain scientists focus their models
  – Allow us to more efficiently construct models in the future
  – Refine our data collection / storage processes
Feature Importance

Common approaches:

• Measure the reduction of impurity for questions involving specific features
  – Support built into the RandomForestClassifier
• How often does a feature occur in a tree?
• Where does a feature occur in a tree?
• *Importance sampling*: how does the model perform when an individual feature is corrupted?
Feature Importance

For now, our focus is on the impurity reduction measure …
Live demo…
Boosting

CV_M12_L08
Boosting

CS/DSA 5970: Machine Learning Practice
Forests

So far: training of one tree is handled independently of other trees

• Natural parallelization

• Independence to varying degrees
  – True independence: can easily combine the output of the different models
  – In general:
    • We don’t necessarily achieve true independence
    • If a part of the sample space is not well represented in the training set, then it will often be ignored by all of the constituent models
Boosting

Alternative approach:
• Grow ensemble in sequence
  – One model at a time
• The model currently being learned attempts to repair prediction errors of the prior models
  – Want each new model to solve a new piece of the problem
  – With the set of models, we attempt to cover all of the training set (even the sparsely represented regions of the sample space)
AdaBoost

- Prior algorithms: all training samples have been treated with equal weight in computing the cost function
- In boosting, we adjust these weights depending on how well the current ensemble performs
Example: AdaBoost

CV_M12_L09
Example: AdaBoost

CS/DSA 5970: Machine Learning Practice
Boosting

• Advantage: at each step, we learn a new model that tries to repair problems with the prior model

• The cost: we lose parallelization
Gradient Boosting

CV_M12_L10
Gradient Boosting

CS/DSA 5970: Machine Learning Practice
Gradient Boosting

• Focus: regression
• Learn a sequence of regression models
  – Each model in the sequence: try to predict the errors from the previous model
  – Then, this model’s output is added to the rest of the model outputs
Example: Gradient Boosting

CV_M12_L11
Example: Gradient Boosting

CS/DSA 5970: Machine Learning Practice
Example: Gradient Boosting

GradientBoostingRegressor class

- learning_rate: total contribution by each tree (shrinkage)
- n_estimators: maximum number of trees
- subsample: fraction of the number of training samples to use for a given tree
- validation_fraction: fraction of samples to hold out to detect overfitting

- Can overfit the training data
  - Cut-off training at some number of trees based on performance
  - We can do this after the fact or dynamically
Live example
Stacking

CV_M12_L12
Stacking

CS/DSA 5970: Machine Learning Practice
Stacking

So far: we have combined the outputs of the individual models through some fixed method

• Voting, averaging …
• Ignores the fact that some models are better than others

• Exception: Boosting
Stacking

We can ask another model to do this combination

• Split the training set
• First training set:
  – Train the individual models
• Second training set:
  – Each model makes predictions for the samples in the 2\textsuperscript{nd} training set
  – New learner (\textit{the blender} or \textit{meta-learner}):
    • Inputs: predictions made by the individual models
    • Outputs: outputs from the 2\textsuperscript{nd} training set
Example: Stacking

CV_M12_L13
Ensemble Methods

CS/DSA 5970: Machine Learning Practice
Stacking

Live example
Boosting