COMPSCI 589
Lecture 7: Ensembles and Classification

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Outline

Ensembles

- An *ensemble* is simply a collection of models that are all trained to perform the same task.

Ensembles

- An *ensemble* is simply a collection of models that are all trained to perform the same task.
- An ensemble can consist of many different versions of the same model, or many different types of models.

Ensembles

- An *ensemble* is simply a collection of models that are all trained to perform the same task.
- An ensemble can consist of many different versions of the same model, or many different types of models.
- The final output for an ensemble of classifiers is typically obtained through a (weighted) average or vote of the predictions of the different models in the ensemble.

Ensembles

- An *ensemble* is simply a collection of models that are all trained to perform the same task.
- An ensemble can consist of many different versions of the same model, or many different types of models.
- The final output for an ensemble of classifiers is typically obtained through a (weighted) average or vote of the predictions of the different models in the ensemble.
- An ensemble of different models that all achieve similar generalization performance often outperforms any of the individual models.

Ensembles

- An *ensemble* is simply a collection of models that are all trained to perform the same task.
- An ensemble can consist of many different versions of the same model, or many different types of models.
- The final output for an ensemble of classifiers is typically obtained through a (weighted) average or vote of the predictions of the different models in the ensemble.
- An ensemble of different models that all achieve similar generalization performance often outperforms any of the individual models.
- **Question:** How is this possible?

Ensemble Intuition

- Suppose we have an ensemble of binary classification functions $f_k(\mathbf{x})$ for $k = 1, ..., K$.

Ensemble Intuition

- Suppose we have an ensemble of binary classification functions $f_k(\mathbf{x})$ for $k = 1, ..., K$.
- Suppose that on average they have the same expected error rate $\epsilon = E_{p(x,y)}[y \neq f_k(\mathbf{x})] < 0.5$, but that the errors they make are *independent*.

Ensemble Intuition

- Suppose we have an ensemble of binary classification functions $f_k(\mathbf{x})$ for $k = 1, ..., K$.
- Suppose that on average they have the same expected error rate $\epsilon = E_{p(x,y)}[y \neq f_k(\mathbf{x})] < 0.5$, but that the errors they make are *independent*.
- The intuition is that the majority of the K classifiers in the ensemble will be correct on many examples where any individual classifier makes an error.

Ensemble Intuition

- Suppose we have an ensemble of binary classification functions $f_k(\mathbf{x})$ for $k = 1, ..., K$.
- Suppose that on average they have the same expected error rate $\epsilon = E_{p(x,y)}[y \neq f_k(\mathbf{x})] < 0.5$, but that the errors they make are *independent*.
- The intuition is that the majority of the K classifiers in the ensemble will be correct on many examples where any individual classifier makes an error.
- A simple majority vote can significantly improve classification performance by *decreasing variance* in this setting.

Ensemble Intuition

- Suppose we have an ensemble of binary classification functions $f_k(\mathbf{x})$ for $k = 1, ..., K$.
- Suppose that on average they have the same expected error rate $\epsilon = E_{p(x,y)}[y \neq f_k(\mathbf{x})] < 0.5$, but that the errors they make are *independent*.
- The intuition is that the majority of the K classifiers in the ensemble will be correct on many examples where any individual classifier makes an error.
- A simple majority vote can significantly improve classification performance by *decreasing variance* in this setting.
- **Question:** How can we come up with such an ensemble?

Independent Training Sets

- Suppose we collect multiple independent training sets $Tr_1, ..., Tr_K$ and use each of these training sets to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.

Independent Training Sets

- Suppose we collect multiple independent training sets $Tr_1, ..., Tr_K$ and use each of these training sets to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.
- Classifiers trained in this way are guaranteed to make independent errors on test data.

Independent Training Sets

- Suppose we collect multiple independent training sets $Tr_1, ..., Tr_K$ and use each of these training sets to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.
- Classifiers trained in this way are guaranteed to make independent errors on test data.
- If the expected error of each classifier is less than 0.5, then the weighted majority vote is guaranteed to reduce the expected generalization error.

Independent Training Sets

- Suppose we collect multiple independent training sets $Tr_1, ..., Tr_K$ and use each of these training sets to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.
- Classifiers trained in this way are guaranteed to make independent errors on test data.
- If the expected error of each classifier is less than 0.5, then the weighted majority vote is guaranteed to reduce the expected generalization error.
- **Question:** What is the weakness of this approach?

Outline

Bagging

- Bootstrap aggregation or *Bagging* is an approximation to the previous method that takes a single training set Tr and randomly sub-samples from it K times (with replacement) to form K training sets $Tr_1, ..., Tr_K$.

Bagging

- Bootstrap aggregation or *Bagging* is an approximation to the previous method that takes a single training set Tr and randomly sub-samples from it K times (with replacement) to form K training sets $Tr_1, ..., Tr_K$.
- Each of these training sets is used to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.

Bagging

- Bootstrap aggregation or *Bagging* is an approximation to the previous method that takes a single training set Tr and randomly sub-samples from it K times (with replacement) to form K training sets $Tr_1, ..., Tr_K$.
- Each of these training sets is used to train a different instance of the same classifier obtaining K classification functions $f_1(\mathbf{x}), ..., f_K(\mathbf{x})$.
- The errors won't by totally independent because the data sets aren't independent, but the random re-sampling usually introduces enough diversity to decrease the variance and give improved performance.

Bagging and Random Forests

- Bagging is particularly useful for high-variance, high-capacity models.

Bagging and Random Forests

- Bagging is particularly useful for high-variance, high-capacity models.
- Historically, it is most closely associated with decision tree models.

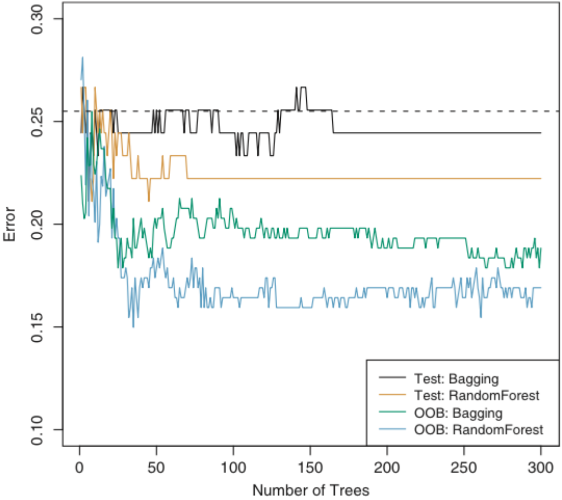
Bagging and Random Forests

- Bagging is particularly useful for high-variance, high-capacity models.
- Historically, it is most closely associated with decision tree models.
- A very successful extension of bagged trees is the random forest classifier.

Bagging and Random Forests

- Bagging is particularly useful for high-variance, high-capacity models.
- Historically, it is most closely associated with decision tree models.
- A very successful extension of bagged trees is the random forest classifier.
- The random forests algorithm further decorrelates the learned trees by only considering a random sub-set of the available features when deciding which variable to split on at each node in the tree.

Example: Bagging vs Random Forests



Outline

Boosting

- Boosting is an ensemble method based on iteratively re-weighting the data set instead of randomly resampling it.

Boosting

- Boosting is an ensemble method based on iteratively re-weighting the data set instead of randomly resampling it.
- The main idea is to up-weight the importance of data cases that are missclassified by the classifiers currently in the ensemble, and then add a next classifier that will focus on data cases that are causing the errors.

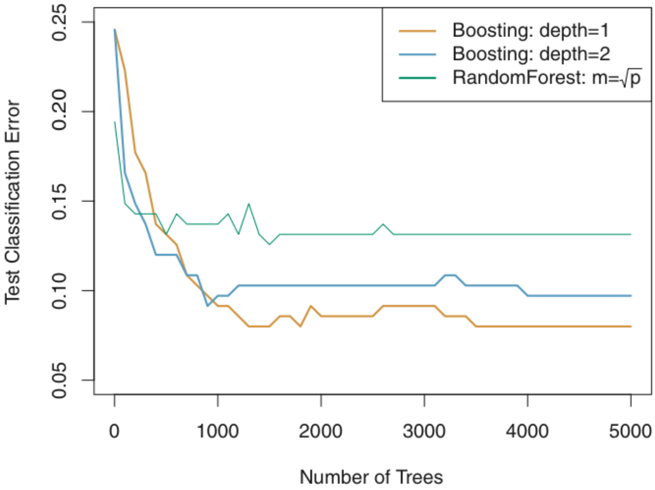
Boosting

- Boosting is an ensemble method based on iteratively re-weighting the data set instead of randomly resampling it.
- The main idea is to up-weight the importance of data cases that are missclassified by the classifiers currently in the ensemble, and then add a next classifier that will focus on data cases that are causing the errors.
- Assuming that the base classifier can always achieve an error of less than 0.5 on any data sample, the boosting ensemble can be shown to decrease error.

AdaBoost Algorithm

1. **Input:** $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, Number of Iterations T
2. **Initialize:** $d_n^{(1)} = 1/N$ for all $n = 1, \ldots, N$
3. **Do for** $t = 1, \ldots, T$,
   (a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathfrak{L}(S, \mathbf{d}^{(t)})$
   (b) Calculate the weighted training error ε_t of h_t:
   $$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ ,$$
   (c) Set:
   $$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
   (d) Update weights:
   $$d_n^{(t+1)} = d_n^{(t)} \exp\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\} / Z_t \ ,$$
   where Z_t is a normalization constant, such that $\sum_{n=1}^{N} d_n^{(t+1)} = 1$.
4. **Break if** $\varepsilon_t = 0$ or $\varepsilon_t \geq \frac{1}{2}$ and set $T = t - 1$.
5. **Output:** $f_T(\boldsymbol{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(\boldsymbol{x})$

Example: AdaBoost



Outline

Stacking (or Blending)

- Unlike bagging and boosting, stacking is an algorithm for combining several different types of models.

Stacking (or Blending)

- Unlike bagging and boosting, stacking is an algorithm for combining several different types of models.
- The main idea is to form a train-validation-test split and train many classifiers $f_k(\mathbf{x})$ on the training data.

Stacking (or Blending)

- Unlike bagging and boosting, stacking is an algorithm for combining several different types of models.
- The main idea is to form a train-validation-test split and train many classifiers $f_k(\mathbf{x})$ on the training data.
- The trained classifiers are used to make predictions on the validation data set and a new feature representation is then created where each data case consists of the vector of predictions of each classifier in the ensemble $\tilde{\mathbf{x}} = [f_1(\mathbf{x}), \ldots, f_K(\mathbf{x})]$.

Stacking (or Blending)

- Unlike bagging and boosting, stacking is an algorithm for combining several different types of models.
- The main idea is to form a train-validation-test split and train many classifiers $f_k(\mathbf{x})$ on the training data.
- The trained classifiers are used to make predictions on the validation data set and a new feature representation is then created where each data case consists of the vector of predictions of each classifier in the ensemble $\tilde{\mathbf{x}} = [f_1(\mathbf{x}), ..., f_K(\mathbf{x})]$.
- Finally, a meta-classifier called a *combiner* is trained to minimize the validation error given the data $\{(y_i, \tilde{\mathbf{x}}_i)|i = 1, .., N\}$.

Stacking (or Blending)

- Unlike bagging and boosting, stacking is an algorithm for combining several different types of models.
- The main idea is to form a train-validation-test split and train many classifiers $f_k(\mathbf{x})$ on the training data.
- The trained classifiers are used to make predictions on the validation data set and a new feature representation is then created where each data case consists of the vector of predictions of each classifier in the ensemble $\tilde{\mathbf{x}} = [f_1(\mathbf{x}), ..., f_K(\mathbf{x})]$.
- Finally, a meta-classifier called a *combiner* is trained to minimize the validation error given the data $\{(y_i, \tilde{\mathbf{x}}_i) | i = 1, .., N\}$.
- The extra layer of combiner learning can deal with correlated classifiers as well as classifiers that perform poorly.

Example: Netflix Prize (2009)

Netflix Prize COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43

Example: Netflix Prize (2009)

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43

Winning team used stacked predictor of 450+ different models.

Outline

Classification Wrap-Up

- We've covered a good mix of classical and state-of-the-art classifiers including KNN, decision trees, naive Bayes, LDA, logistic regression, SVMs, neural networks and ensembles.

Classification Wrap-Up

- We've covered a good mix of classical and state-of-the-art classifiers including KNN, decision trees, naive Bayes, LDA, logistic regression, SVMs, neural networks and ensembles.
- We covered three of the most important meta issues in classification: generalization assessment, capacity control, and hyperparameter selection.

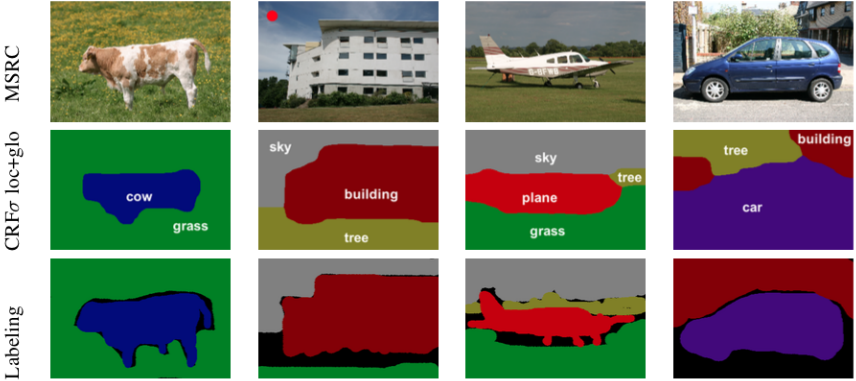
Classification Wrap-Up

- We've covered a good mix of classical and state-of-the-art classifiers including KNN, decision trees, naive Bayes, LDA, logistic regression, SVMs, neural networks and ensembles.
- We covered three of the most important meta issues in classification: generalization assessment, capacity control, and hyperparameter selection.
- Things we didn't cover: feature selection, feature engineering, dealing with class imbalance, covariate shift, cost of errors, classifier evaluation beyond accuracy, structured prediction, sequential decisions...

Classifier Evaluation

		actual value		
		p	*n*	total
prediction outcome	*p'*	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
	total	P	N	

Image Segmentation



Mesh Segmentation

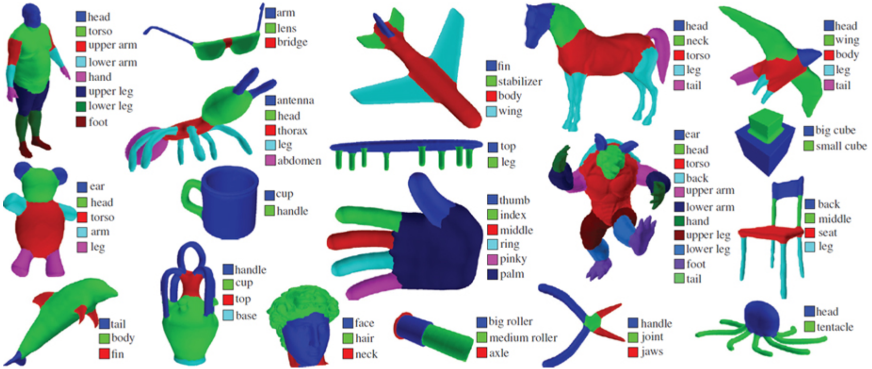


Image to Text

Image to Text

A very cute looking cat in a hat

Playing Atari Breakout

https://www.youtube.com/watch?v=EfGD2qveGdQ