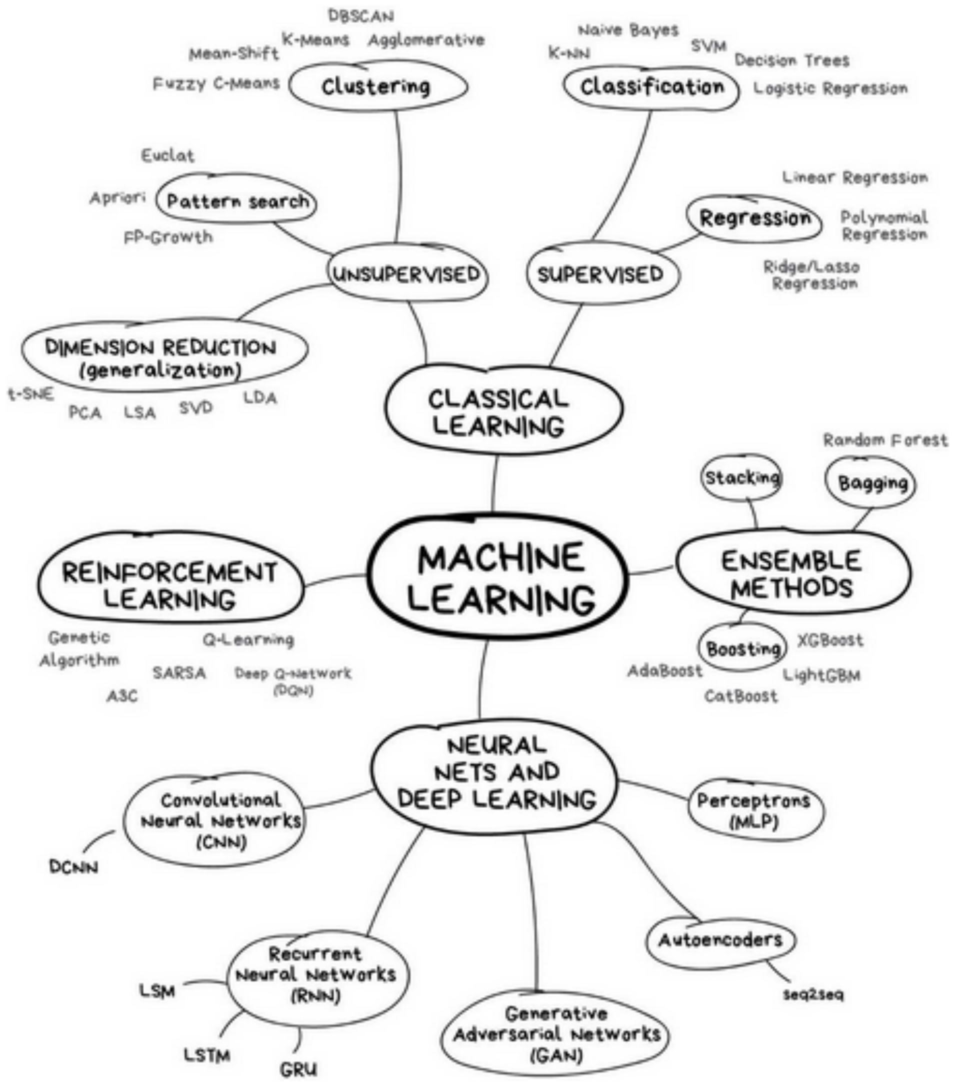
What Machine learning is



Ensembles

Based on Ray Mooney CS 391L

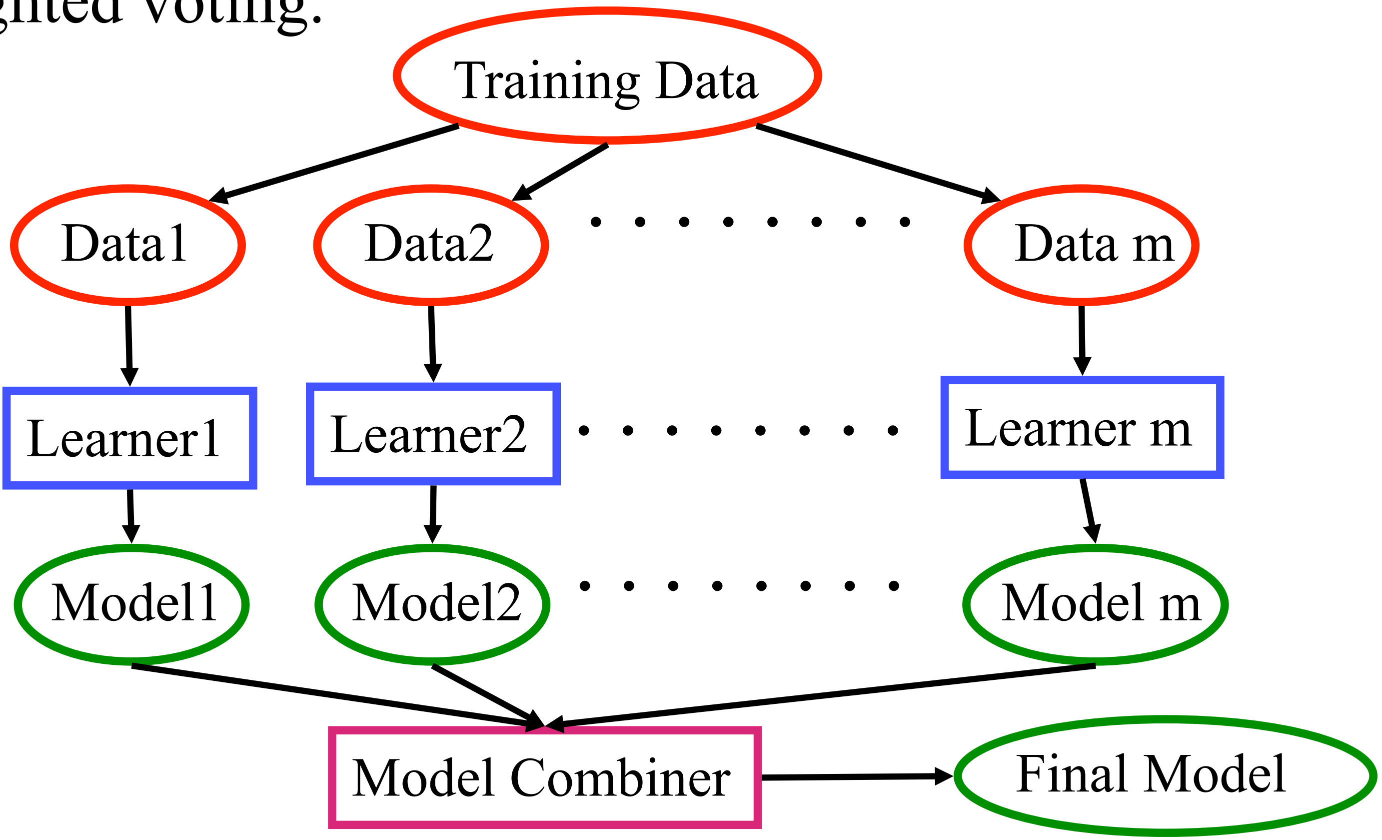
University of Texas at Austin

Bias-variance dilemma

- bias–variance dilemma: a low-complexity model suffers less from variability due to random variations in the training data, but
- may introduce a systematic bias that even large amounts of training data can't resolve;
- Example(s):
- on the other hand,
- a high-complexity model eliminates such bias but can suffer non-systematic errors due to variance.
- Example(s):

Learning Ensembles

- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



Value of Ensembles

- When combing multiple ***independent*** and ***diverse*** decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Expert friend vs. audience vote.

Stacking

- considers heterogeneous weak learners
- learns them in parallel and
- combines them by training a meta-model to output a prediction based on the different weak models predictions
- meta-learner – any
- Actually, stacking is a kind of general model for ensemble learning

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

Bagging

- Create ensembles by repeatedly randomly resampling the training data (Brieman, 1996).
- Given a training set of size n, create m samples of size n by drawing n examples from the original data, ***with replacement***.
  - Each ***bootstrap sample*** will on average contain 63.2% of the unique training examples, the rest are replicates.
- Combine the m resulting models using simple majority vote.
- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.

Bagging : Algorithms

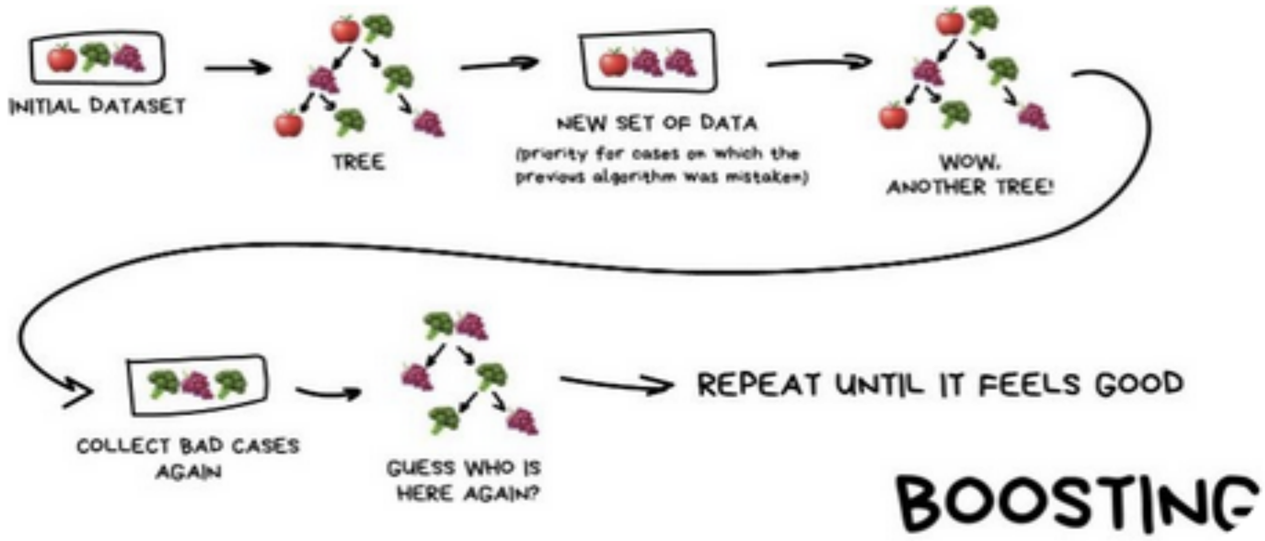
Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

```
for t = 1 to T do
    build a bootstrap sample D_t from D by sampling |D| data points with replacement;
    run 𝒜 on D_t to produce a model M_t;
end
return {M_t | 1 ≤ t ≤ T}
```

Boosting



- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 (Schapire, 1990; Goedel Prize)

Boosting

- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).

- Examples are given weights. At each iteration, a new hypothesis is learned and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.

Boosting: Basic Algorithm

- General Loop:

  Set all examples to have equal uniform weights.
  For t from 1 to T do:
  Learn a hypothesis, h_t, from the weighted examples
  Decrease the weights of examples h_t classifies correctly
- Base (weak) learner must focus on correctly classifying the most highly weighted examples while strongly avoiding over-fitting.
- During testing, each of the T hypotheses get a weighted vote proportional to their accuracy on the training data.

AdaBoost Pseudocode

TrainAdaBoost(D, BaseLearn)

For each example d_i in D let its weight w_i=1/|D|

Let H be an empty set of hypotheses

For t from 1 to T do:

Learn a hypothesis, h_t, from the weighted examples: h_t=BaseLearn(D)

Add h_t to H

Calculate the error, ε_t, of the hypothesis h_t as the total sum weight of the examples that it classifies incorrectly.

If $\varepsilon_t > 0.5$ then exit loop, else continue.

Let $\beta_t = \varepsilon_t \ / \ (1 - \varepsilon_t\ )$

Multiply the weights of the examples that h_t classifies correctly by β_t

Rescale the weights of all of the examples so the total sum weight remains 1.

Return H

TestAdaBoost(ex, H)

Let each hypothesis, h_t, in H vote for ex's classification with weight log(1/ β_t)

Return the class with the highest weighted vote total.

Note on ensemble construction

- Ensemble construction can be defined as a learning problem
- given the predictions of some base classifiers as features, learn a meta-model that best combines their predictions.
- E.g. in Bagging, what classifiers to use and with what weights (weighted voting)
- In Boosting we could learn the weights rather than deriving them from each base model's error rate.

Random Forests

- an ensemble of classification or regression random trees.
- each Random tree is constructed by a
  - different bootstrap sample from the original data
  - with a subset of features
- 1/3 of all samples are left out (a cause of bootstrap) – OOB (out of bag) data – for classification error estimation
- majority voting, = a variant of bagging

Random Forest

Algorithm RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

Input : data set D; ensemble size T; subspace dimension d.
Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

for $t = 1$ to T **do**
  build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
  select d features at random and reduce dimensionality of D_t accordingly;
  train a tree model M_t on D_t without pruning;
end
return $\{M_t | 1 \leq t \leq T\}$

Learning with Weighted Examples

- Generic approach is to replicate examples in the training set proportional to their weights (e.g. 10 replicates of an example with a weight of 0.01 and 100 for one with weight 0.1).
- Most algorithms can be enhanced to efficiently incorporate weights directly in the learning algorithm so that the effect is the same (e.g. implement the WeightedInstancesHandler interface in WEKA).
- For decision trees, for calculating information gain, when counting example i, simply increment the corresponding count by w_i rather than by 1.
- For kNN and other learners?

Experimental Results on Ensembles (Freund & Schapire, 1996; Quinlan, 1996)

- Ensembles have been used to improve generalization accuracy on a wide variety of problems.
- On average, Boosting provides a larger increase in accuracy than Bagging.
- Boosting on rare occasions can degrade accuracy.
- Bagging more consistently provides a modest improvement.
- Boosting is particularly subject to over-fitting when there is significant noise in the training data.

Issues in Ensembles

- Parallelism in Ensembles: Bagging is easily parallelized, Boosting is not.
- Variants of Boosting to handle noisy data.
- How "weak" should a base-learner for Boosting be?
- What is the theoretical explanation of boosting's ability to improve generalization?
- Exactly how does the diversity of ensembles affect their generalization performance.
- Combining Boosting and Bagging.

Ensembles and bias-variance dilemma

- Bagging decreases variance
  variance -> variance/num_of_ensembleMembers

- Boosting decreases bias
  (as hypothesis complexity is increasing)

Ensembles and Active Learning

- Ensembles can be used to actively select good new training examples.
- Select the unlabeled example that causes the most disagreement amongst the members of the ensemble.
- Applicable to any ensemble method:
  - QueryByBagging
  - QueryByBoosting