# CSC 411: Lecture 17: Ensemble Methods I

Raquel Urtasun & Rich Zemel

University of Toronto

Nov 18, 2015

## Today

- Ensemble Methods
- Bagging
- Boosting

## Ensemble methods

- Typical application: classification
- Ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way to classify new examples
- Simplest approach:
  1. Generate multiple classifiers
  2. Each votes on test instance
  3. Take majority as classification
- Classifiers are different due to different sampling of training data, or randomized parameters within the classification algorithm
- Aim: take simple mediocre algorithm and transform it into a super classifier without requiring any fancy new algorithm

## Ensemble methods: Summary

- Differ in training strategy, and combination method
  - Parallel training with different training sets
    1. Bagging (bootstrap aggregation) – train separate models on overlapping training sets, average their predictions
    2. Cross-validated committees – disjoint subsets of training sets
  - Sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples: boosting
  - Parallel training with objective encouraging division of labor: mixture of experts
- Notes:
  - Also known as meta-learning
  - Typically applied to weak models, such as decision stumps (single-node decision trees), or linear classifiers

## Variance-bias tradeoff

- Minimize two sets of errors:
  1. Variance: error from sensitivity to small fluctuations in the training set
  2. Bias: erroneous assumptions in the model
- Variance-bias decomposition is a way of analyzing the generalization error as a sum of 3 terms: variance, bias and irreducible error (resulting from the problem itself)

## Why do ensemble methods work?

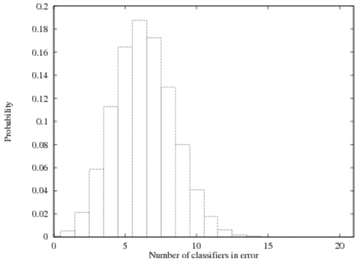
- Based on one of two basic observations:
  1. Variance reduction: if the training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias (e.g., bagging)
     - reduce sensitivity to individual data points
  2. Bias reduction: for simple models, average of models has much greater capacity than single model (e.g., hyperplane classifiers, Gaussian densities).
     - Averaging models can reduce bias substantially by increasing capacity, and control variance by fitting one component at a time (e.g., boosting)

## Ensemble methods: Justification

- Ensemble methods more accurate than any individual members:
  - Accurate (better than guessing)
  - Diverse (different errors on new examples)
- Independent errors: prob $k$ of $N$ classifiers (independent error rate $\epsilon$) wrong:

$$P(\text{num errors} = k) = \binom{N}{k} \epsilon^k (1-\epsilon)^{N-k}$$

- Probability that majority vote wrong: error under distribution where more than $N/2$ wrong



## Ensemble methods: Netflix

- Clear demonstration of the power of ensemble methods
- Original progress prize winner (BellKor) was ensemble of 107 models!
  - *"Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique."*
  - *"We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor."*

## Bootstrap estimation

- Repeatedly draw n samples from $D$
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance
- Bagging: bootstrap aggregation (Breiman 1994)

## Bagging

- Simple idea: generate M bootstrap samples from your original training set. Train on each one to get $y_m$, and average them

$$y^M_{bag}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

- For regression: average predictions
- For classification: average class probabilities (or take the majority vote if only hard outputs available)
- Bagging approximates the Bayesian posterior mean. The more bootstraps the better, so use as many as you have time for
- Each bootstrap sample is drawn with replacement, so each one contains some duplicates of certain training points and leaves out other training points completely

## Boosting (AdaBoost): Summary

- Also works by manipulating training set, but classifiers trained sequentially
- Each classifier trained given knowledge of the performance of previously trained classifiers: focus on hard examples
- Final classifier: weighted sum of component classifiers

## Making weak learners stronger

- Suppose you have a weak learning module (a base classifier) that can always get $(0.5 + \epsilon)$ correct when given a two-way classification task
    - That seems like a weak assumption but beware!
- Can you apply this learning module many times to get a strong learner that can get close to zero error rate on the training data?
    - Theorists showed how to do this and it actually led to an effective new learning procedure (Freund & Shapire, 1996)

## Boosting (ADAboost)

- First train the base classifier on all the training data with equal importance weights on each case.
- Then re-weight the training data to emphasize the hard cases and train a second model.
  - How do we re-weight the data?
- Keep training new models on re-weighted data
- Finally, use a weighted committee of all the models for the test data.
  - How do we weight the models in the committee?

## How to train each classifier

- Input: $\mathbf{x}$, Output: $y(\mathbf{x}) \in \{1, -1\}$
- Target $t \in \{-1, 1\}$
- Weight on example $n$ for classifier $m$: $\mathbf{w}_n^m$
- Cost function for classifier $m$

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(\mathbf{x}^n) \neq t^{(n)}]}_{1 \text{ if error, } 0 \text{ o.w.}} = \sum \text{weighted errors}$$

## How to weight each training case for classifier $m$

- Recall cost function is

$$J_m = \sum_{n=1}^{N} w_n^m \underbrace{[y_m(\mathbf{x}^n) \neq t^{(n)}]}_{1 \text{ if error}, \quad 0 \text{ o.w.}} = \sum \text{weighted errors}$$

- Weighted error rate of a classifier

$$\epsilon_m = \frac{J_m}{\sum w_n^m}$$

- The quality of the classifier is

$$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$$

  It is zero if the classifier has weighted error rate of 0.5 and infinity if the classifier is perfect

- The weights for the next round are then

$$w_n^{m+1} = w_n^m \exp\{\alpha_m [y_m(\mathbf{x}^{(n)}) \neq t^{(n)}]\}$$

## How to make predictions using a committee of classifiers

- Weight the binary prediction of each classifier by the quality of that classifier:

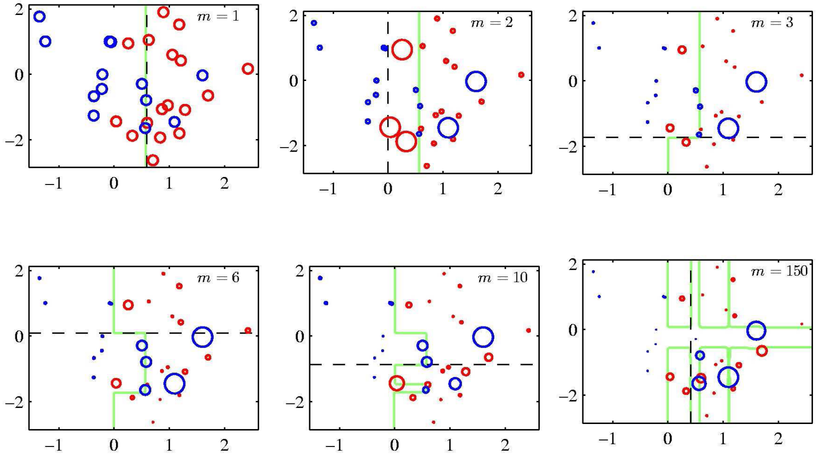
$$y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x})\right)$$

- This is how to do inference, i.e., how to compute the prediction for each new example.

## AdaBoost algorithm

- Input: $\{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^{N}$, and WeakLearn: learning procedure, produces classifier $y(\mathbf{x})$
- Initialize example weights: $w_n^m(\mathbf{x}) = 1/N$
- For m=1:M
  - $y_m(\mathbf{x}) = WeakLearn(\{\mathbf{x}\}, \mathbf{t}, \mathbf{w})$, fit classifier by minimizing
$$J_m = \sum_{n=1}^{N} w_n^m [y_m(\mathbf{x}^n) \neq t^{(n)}]$$
  - Compute unnormalized error rate
$$\epsilon_m = \sum_{n=1}^{N} w_n^m [y_m(\mathbf{x}^n) \neq t^{(n)}]$$
  - Compute classifier coefficient $\alpha_m = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$
  - Update data weights
$$w_n^{m+1} = \frac{w_n^m \exp\{-\alpha_m t^{(n)} y_m(\mathbf{x}^{(n)})\}}{\sum_{n=1^N} w_n^m \exp\{-\alpha_m t^{(n)} y_m(\mathbf{x}^{(n)})\}}$$
- Final model
$$Y(\mathbf{x}) = sign(y_M(\mathbf{x})) = sign(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$$

## AdaBoost example



- Each figure shows the number *m* of base learners trained so far, the decision of the most recent learner (dashed black), and the boundary of the ensemble (green)

## An alternative derivation of ADAboost

- Just write down the right cost function and optimize each parameter to minimize it
  - stagewise additive modeling (Friedman et. al. 2000)
- At each step employ the exponential loss function for classifier $m$

$$E = \sum_{n=1}^{N} \exp\{-t^{(n)} f_m(\mathbf{x}^{(n)})\}$$

- Real-valued prediction by committee of models up to $m$

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{m} \alpha_i y_i(\mathbf{x})$$

- We want to minimize $E$ w.r.t. $\alpha_m$ and the parameters of the classifier $y_m(\mathbf{x})$
- We do this in a sequential manner, one classifier at a time

## Learning classifier m using exponential loss

- At iteration $m$, the energy is computed as

$$E = \sum_{n=1}^{N} \exp\{-t^{(n)} f_m(\mathbf{x}^{(n)})\}$$

with

$$f_m(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} \alpha_i y_i(\mathbf{x}) = \frac{1}{2}\alpha_m y_m(\mathbf{x}) + \frac{1}{2}\sum_{i=1}^{m-1} \alpha_i y_i(\mathbf{x})$$

- We can compute the part that is relevant for the $m$-th classifier

$$\begin{aligned} E_{relevant} &= \sum_{n=1}^{N} \exp\left(-t^{(n)} f_{m-1}(\mathbf{x}^{(n)}) - \frac{1}{2}\alpha_m y_m(\mathbf{x}^{(n)})\right) \\ &= \sum_{n=1}^{N} w_n^m \exp\left(-\frac{1}{2}t^{(n)}\alpha_m y_m(\mathbf{x}^{(n)})\right) \end{aligned}$$

with $w_n^m = \exp\left(-t^{(n)} f_{m-1}(\mathbf{x}^{(n)})\right)$

## Continuing the derivation

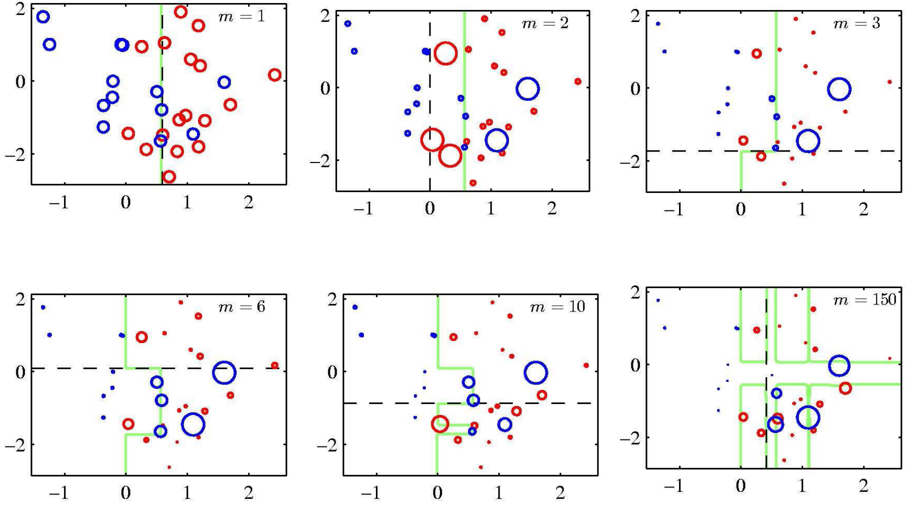
$$
\begin{aligned}
E_{relevant} &= \sum_{n=1}^{N} w_n^m \exp\left(-t^{(n)}\frac{\alpha_m}{2} y_m(\mathbf{x}^{(n)})\right) \\
&= e^{-\frac{\alpha_m}{2}} \sum_{right} w_n^m + e^{\frac{\alpha_m}{2}} \sum_{wrong} w_n^m \\
&= \underbrace{\left(e^{\frac{\alpha_m}{2}} - e^{\frac{-\alpha_m}{2}}\right)}_{multiplicative\ \ constant} \underbrace{\sum_n w_n^m [t^{(n)} \neq y_m(\mathbf{x}^{(n)})]}_{wrong\ \ cases} + \underbrace{e^{-\frac{\alpha_m}{2}} \sum_n w_n^m}_{unmodifiable}
\end{aligned}
$$

- The second term is constant w.r.t. $y_m(\mathbf{x})$
- Thus we minimize the weighted number of wrong examples

## AdaBoost algorithm

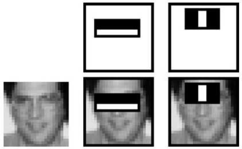
- Input: $\{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^{N}$, and WeakLearn: learning procedure, produces classifier $y(\mathbf{x})$
- Initialize example weights: $w_n^m(\mathbf{x}) = 1/N$
- For m=1:M
  - $y_m(\mathbf{x}) = WeakLearn(\{\mathbf{x}\}, \mathbf{t}, \mathbf{w})$, fit classifier by minimizing
    $$J_m = \sum_{n=1}^{N} w_n^m [y_m(\mathbf{x}^n) \neq t^{(n)}]$$
  - Compute unnormalized error rate
    $$\epsilon_m = \sum_{n=1}^{N} w_n^m [y_m(\mathbf{x}^n) \neq t^{(n)}]$$
  - Compute classifier coefficient $\alpha_m = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$
  - Update data weights
    $$w_n^{m+1} = \frac{w_n^m \exp\{-\alpha_m t^{(n)} y_m(\mathbf{x}^{(n)})\}}{\sum_{n=1^N} w_n^m \exp\{-\alpha_m t^{(n)} y_m(\mathbf{x}^{(n)})\}}$$
- Final model
  $$Y(\mathbf{x}) = sign(y^M(\mathbf{x})) = sign(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$$

## AdaBoost example



## An impressive example of boosting

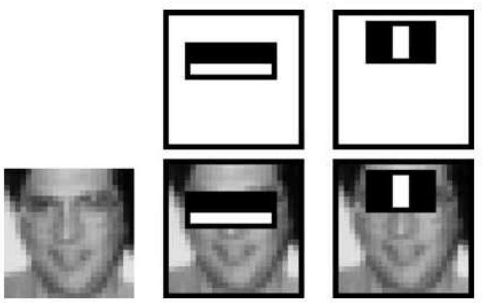
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.



- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image.
  - There is a neat trick for computing the total intensity in a rectangle in a few operations.
    - So its easy to evaluate a huge number of base classifiers and they are very fast at runtime.
  - The algorithm adds classifiers greedily based on their quality on the weighted training cases.

## AdaBoost in face detection

- Famous application of boosting: detecting faces in images
- Two twists on standard algorithm
  - Pre-define weak classifiers, so optimization=selection
  - Change loss function for weak learners: false positives less costly than misses



# AdaBoost face detection results