An Introduction to Statistical Machine Learning - Ensembles -

Samy Bengio

bengio@idiap.ch

Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP)
CP 592, rue du Simplon 4
1920 Martigny, Switzerland
http://www.idiap.ch/~bengio

Ensemble Models

1. Basics of Ensembles
2. Bagging
3. AdaBoost
4. Mixture of Experts (already seen!)

Basics of Ensembles

- When trying to solve a problem, we generally make some choices:
  - family of functions, range of the hyper-parameters
  - input representation and preprocessing
  - precise dataset
  - etc
- Idea: instead of making these choices, let us provide not one but many solutions to the same problem, and let us combine them
- Why should this be a good idea?
  - These choices imply a variance in the expected performance (implicit capacity).
  - In general, combining estimates → reduces the variance → enhances expected performance.

Ensemble - Why Does it Work?

- It has been shown that the expected risk of the average of a set of models is better than the average of the expected risk of these models
- Let us consider the simplest ensemble g over models f_i:

$$g(x) = \sum_i \alpha_i f_i(x) \text{ with } \sum_i \alpha_i = 1$$

- The MSE risk of f_i at x is $e_i(x) = E_y[(y - f_i(x))^2]$
- The average risk of a model is $\bar{e}(x) = \sum_i \alpha_i e_i(x)$
- The average risk of the ensemble is $e(x) = E_y[(y - g(x))^2]$
- Let us define diversity $d_i(x) = (f_i(x) - g(x))^2$
- The average diversity is $\bar{d}(x) = \sum_i \alpha_i d_i(x)$
- It can then be shown that $e(x) = \bar{e}(x) - \bar{d}(x)$

Bagging

- Bagging: bootstrap aggregating
- Underlying idea: part of the variance is due to the specific choice of the training data set
- Let us create many similar training data sets,
- For each of them, let us train a new function
- The final function will be the average of each function outputs.
- How similar? using bootstrap.

Bootstrap

- Given a data set D_n with n examples drawn from $p(Z)$
- A bootstrap B_i of D_n also contains n examples:
- For $j = 1 \rightarrow n$, the j^{th} example of B_i is drawn independently with replacement from D_n
- Hence,
  - some examples from D_n are in multiple copies in B_i
  - and some examples from D_n are not in B_i
- Hypothesis: the examples were iid drawn from $p(Z)$
- Hence, the datasets B_i are as plausible as D_n, but drawn from D_n instead of $p(Z)$.

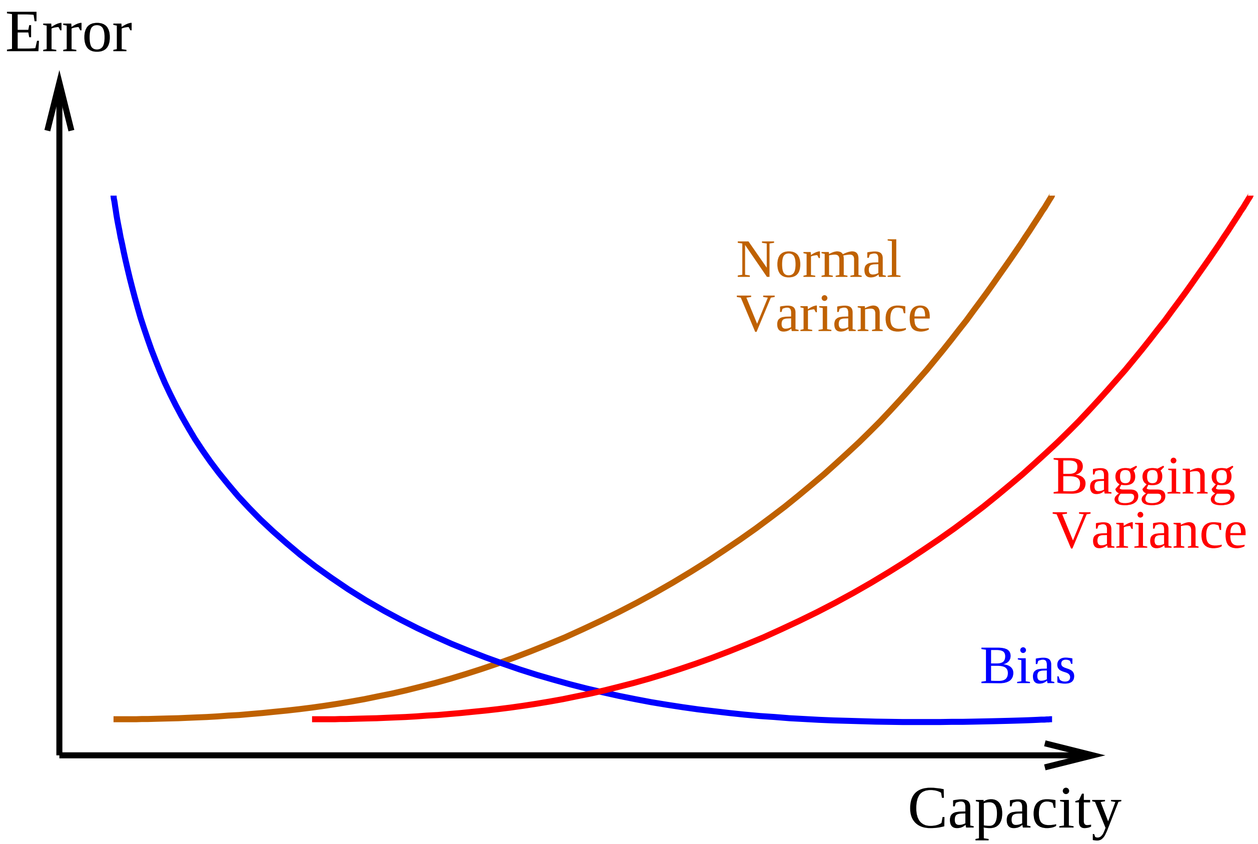
Bagging - Algorithm

- Training:
  1. Given a training set D_n, create T bootstraps B_i of D_n
  2. For each bootstrap B_i, select $f^*(B_i) = \arg\min_{f \in \mathcal{F}} \hat{R}(f, B_i)$
- Testing:
  - Given an input x, the corresponding output $\hat{y}$ is:

$$\hat{y} = \frac{1}{T} \sum_{i=1}^{T} f^*(B_i)(x)$$

- Analysis: if generalization error is decomposed into bias and variance terms then bagging reduces variance.

Bias + Variance for Bagging



AdaBoost

- Most popular algorithm in the family of boosting algorithms
- Boosting: the performance of simple (weak) classifiers is boosted by combining them iteratively.
- General combination classifier:

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

- Simplest framework: binary classification, targets $= \{-1, +1\}$
- What can we do with the following simplest requirement: each weak classifier f_t should perform better than chance

AdaBoost - Concepts

- AdaBoost is an iterative algorithm: select f_t given the performance obtained by previous weak classifiers $f_1 \rightarrow f_{t-1}$.
- At each time step t,
  - Modify training sample distribution in order to favor difficult examples (according to previous weak classifiers).
  - Train a new weak classifier
  - Select the new weight α_t by optimizing a global criterion
- Stop when impossible to find a weak classifier satisfying the simplest condition (being better than chance)
- Final solution is the weighted sum of all weak classifiers

AdaBoost - Algorithm

1. inputs: $D_n = \{(x_1, y_1), \cdots, (x_n, y_n)\}$
2. initialize: $w_i^{(1)} = \frac{1}{n}$ for all $i = 1, \cdots, n$
3. for $t = 1, \cdots, T$
   (a) $D^{(t)}$: sample n examples from D_n according to weights $w^{(t)}$
   (b) train classifier f_t using $D^{(t)}$
   (c) calculate weighted training error ϵ_t of f_t:
   $$\epsilon_t = \sum_{i=1}^{n} w_i^{(t)} I(y_i \neq f_t(x_i))$$
   where $I(z) = 1$ if z is true, 0 otherwise
   (d) calculate weight α_t of weak classifier f_t:
   $$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

AdaBoost - Algorithm

(e) update weights of examples for next iteration:

$$w_i^{(t+1)} = w_i^{(t)} \frac{\exp(-\alpha_t y_i f_t(x_i))}{Z_t}$$

where Z_t is a normalization factor such that $\sum_i w_i^{(t+1)} = 1$.

(f) if $\epsilon_t = 0$ or $\epsilon_t \geq \frac{1}{2}$, break: $T = t - 1$.

4. Final output:

$$g(x) = \sum_t \frac{\alpha_t}{\sum_r \alpha_r} f_t(x)$$

AdaBoost - Analysis

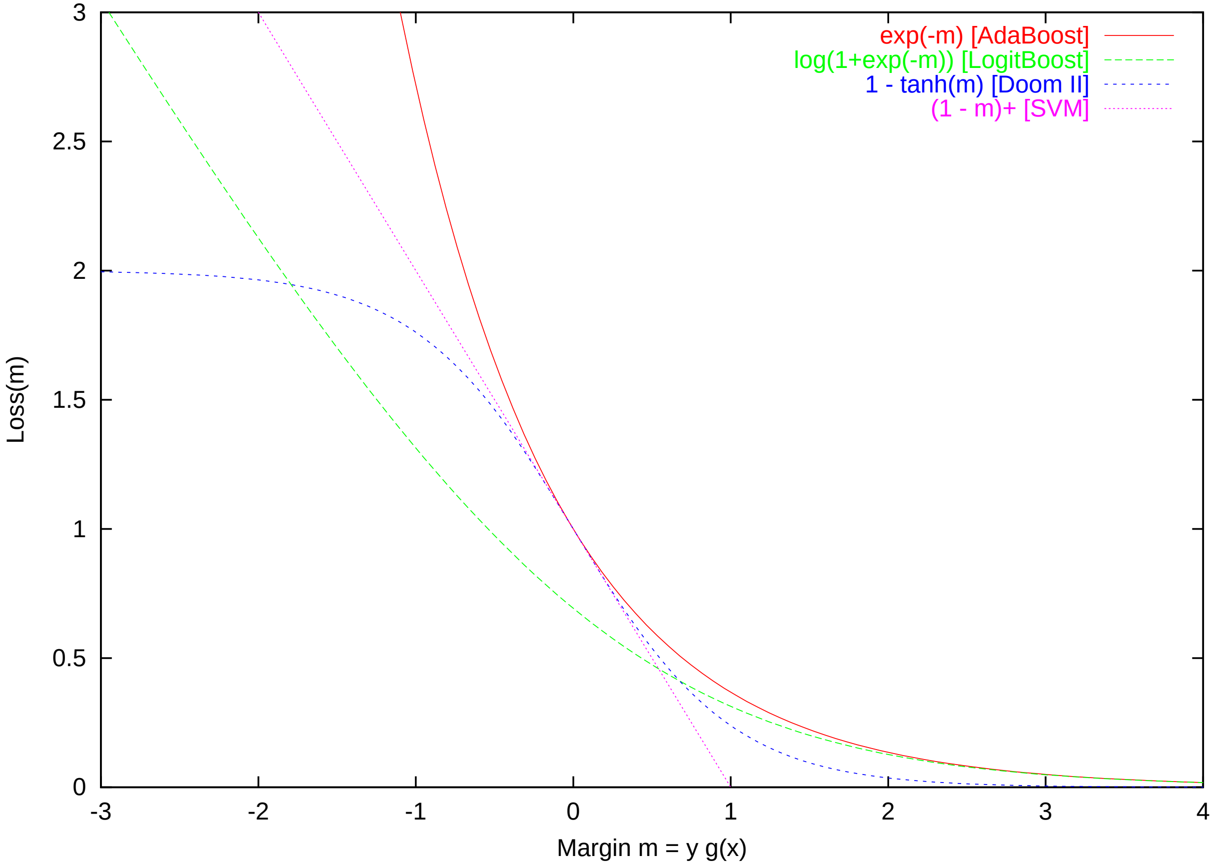
- Selection of α_t comes from minimizing

$$\alpha_t^* = \arg\min_{\alpha_t} \sum_{i=1}^{n} \exp\left(-y_i\left(\alpha_t f_t(x_i) + \sum_{s=1}^{t-1} \alpha_s f_s(x_i)\right)\right)$$

- Other cost functions have been proposed (such as logitboost or arcing)
- Sampling can often be replaced by weighting
- If each weak classifier is always better than chance, then AdaBoost can be proven to converge to 0 training error
- Even after training error is 0, generalization error continues to improve: the margin continues to grow
- Early claims: AdaBoost does not overfit! This is false of course...

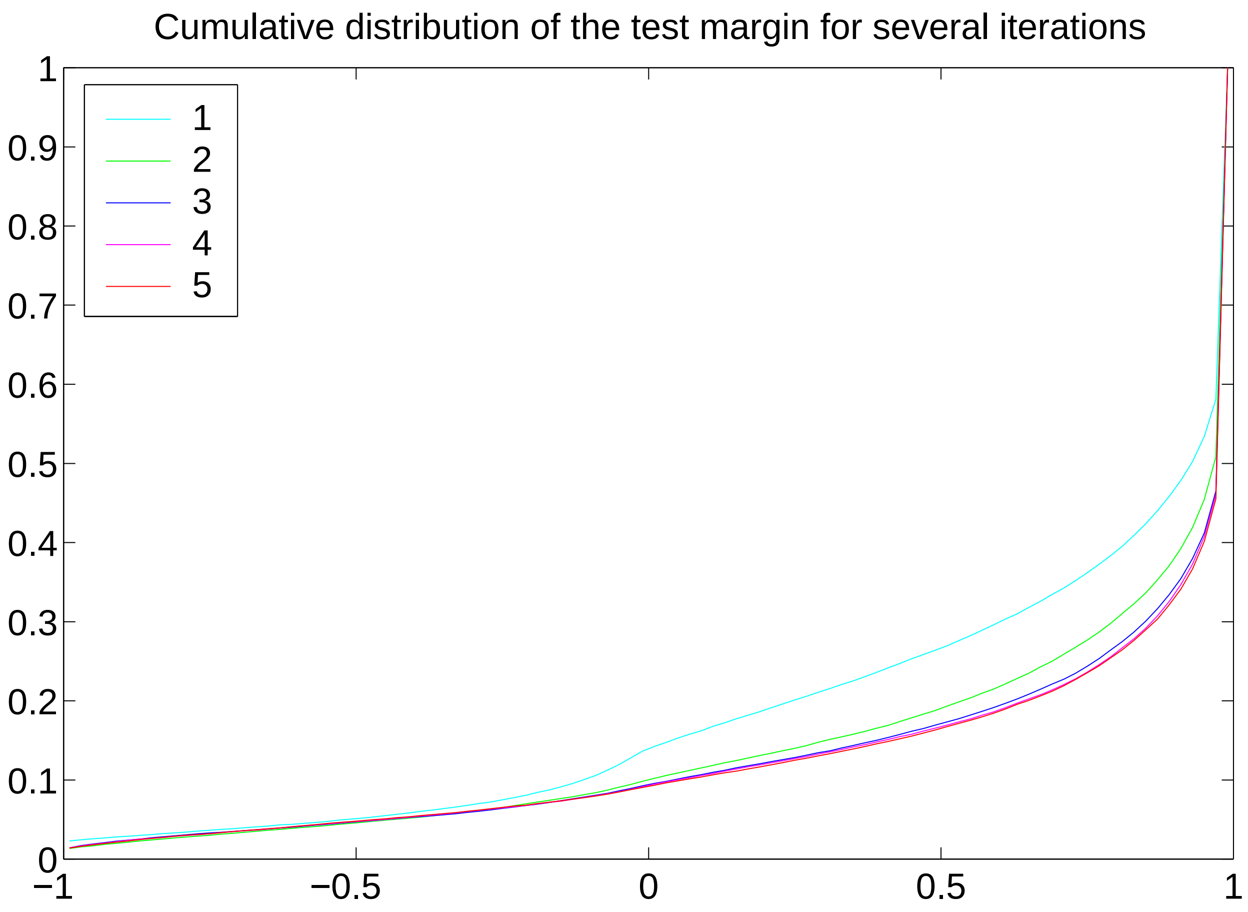
AdaBoost - Cost Functions

- Comparison of various cost functions related to AdaBoost:



AdaBoost - Margin

- The AdaBoost margin is defined as the distribution of $y \cdot g(x)$:



AdaBoost - Extensions

- Multi-class classification
- Single-class classification: estimating quantiles
- Regression: transform the problem into a binary classification task
- Localized Boosting: similar to mixtures of experts

$$g(x) = \sum_{t=1}^{T} \alpha_t(x) \cdot f_t(x)$$

- Examples of weak classifiers:
  - Decision trees and stumps
  - neural networks