ENSEMBLES IN MACHINE LEARNING

What is Ensemble Classification?

- Use multiple learning algorithms (classifiers)
- Combine the decisions
- Can be more accurate than the individual classifiers
- Generate a group of base-learners
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations (Modalities)
  - Training sets

Why should it work?

- Works well only if the individual classifiers disagree
  - Error rate < 0.5 and errors are independent
  - Error rate is highly correlated with the correlations of the errors made by the different learners

Bias vs. Variance

- We would like low bias error and low variance error
- Ensembles using multiple trained (high variance/low bias) models can average out the variance, leaving just the bias
  - Less worry about overfit (stopping criteria, etc.) with the base models

Combining Weak Learners

- Combining weak learners
  - Assume *n* independent models, each having accuracy of 70%.
  - If all *n* give the same class output then you can be confident it is correct with probability $1-(1-.7)^n$
  - Normally not completely independent, but unlikely that all *n* would give the same output
    - Accuracy better than the base accuracy of the models by using the majority output.

– If *n1* models say class 1 and *n2*<*n1* models say class 2, then

P(class1) = 1 – Binomial(*n*, n2, .7)

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Ensemble Creation Approaches

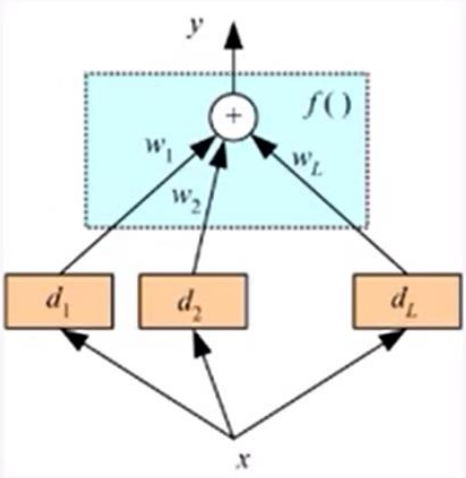
- Get less correlated errors between models
  - Injecting randomness
    - initial weights (eg, NN), different learning parameters, different splits (eg, DT) etc.
  - Different Training sets
    - Bagging, Boosting, different features, etc.
  - Forcing differences
    - different objective functions
  - Different machine learning model

Ensemble Combining Approaches

- Unweighted Voting (e.g. Bagging)
- Weighted voting – based on accuracy (e.g. Boosting), Expertise, etc.
- Stacking - Learn the combination function

Combine Learners: Voting

- Unweighted voting
- Linear combination (weighted vote)
  - weight ∝ accuracy
  - weight ∝ $^1/_{\text{variance}}$



$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

- Bayesian

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j) P(\mathcal{M}_j)$$

Fixed Combination Rules

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L} \sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

- All possible models in the model space used weighted by their probability of being the "Correct" model
- Optimal given the correct model space and priors

Bayes Optimal Classifier

- The Bayes Optimal Classifier is an ensemble of all the hypotheses in the hypothesis space.
- On average, no other ensemble can outperform it.
- The vote for each hypothesis
  - proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true.
  - is multiplied by the prior probability of that hypothesis.

$$y = \mathrm{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

$$y = \text{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

- y is the predicted class,
- C is the set of all possible classes,
- H is the hypothesis space,
- T is the training data.

The Bayes Optimal Classifier represents a hypothesis that is not necessarily in H.

But it is the optimal hypothesis in the ensemble space.

Practicality of Bayes Optimal Classifier

- Cannot be practically implemented.
- Most hypothesis spaces are too large
- Many hypotheses output a class or a value, and not probability
- Estimating the prior probability for each hypothesizes is not always possible.

- All possible models in the model space used weighted by their probability of being the “Correct” model
- Optimal given the correct model space and priors

Challenge for developing Ensemble Models

- The main challenge is to obtain base models which are independent and make independent kinds of errors.
- Independence between two base classifiers can be assessed in this case by measuring the degree of overlap in training examples they misclassify
  ($|A \cap B|/|A \cup B|$)

Bagging

- Bagging = "bootstrap aggregation"
  - Draw N items from X with replacement
- Desired learners with high variance (unstable)
  - Decision trees and ANNs are unstable
  - K-NN is stable
- Use bootstrapping to generate *L* training sets and train one base-learner with each (Breiman, 1996)
- Use voting

- Sampling with replacement
- Build classifier on each bootstrap sample
- Each sample has probability $(1 - 1/n)^n$ of being selected

Boosting

- An iterative procedure. Adaptively change distribution of training data.
  - Initially, all N records are assigned equal weights
  - Weights change at the end of boosting round
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a hypothesis: h_t
  - A strength for this hypothesis: α_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength
- "weak" learners
  - P(correct) > 50%, but not necessarily much better

Adaboost

- Boosting can turn a weak algorithm into a strong learner.
- Input: S={$(x_1, y_1), \ldots, (x_m, y_m)$ }
- $D_t(i)$: weight of i th training example
- Weak learner A
- For $t = 1,2, \ldots, T$
  - Construct D_t on $\{x_1, x_2 \ldots\}$
  - Run A on D_t producing $h_t: X \to \{-1,1\}$

    ϵ_t =error of h_t over D_t

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak classifier $h_t: X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.

$$D_t + 1(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) exp\,(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where
$x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train weak learner using distribution D_t.
- Get weak classifier $h_t : X \rightarrow \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_t + 1(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) exp\,(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Choose α_t to minimize training error

$$\alpha_t = \frac{1}{2} In\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

where

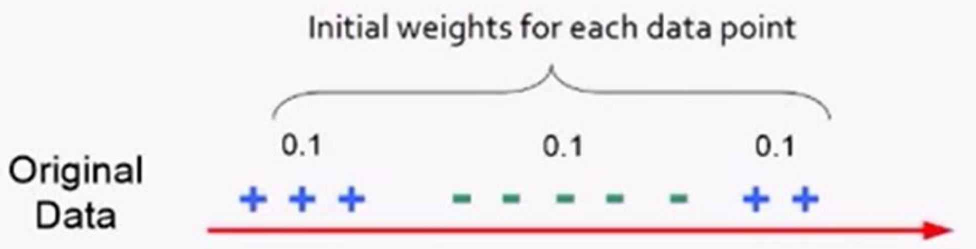
$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x_i) \neq y_i)$$

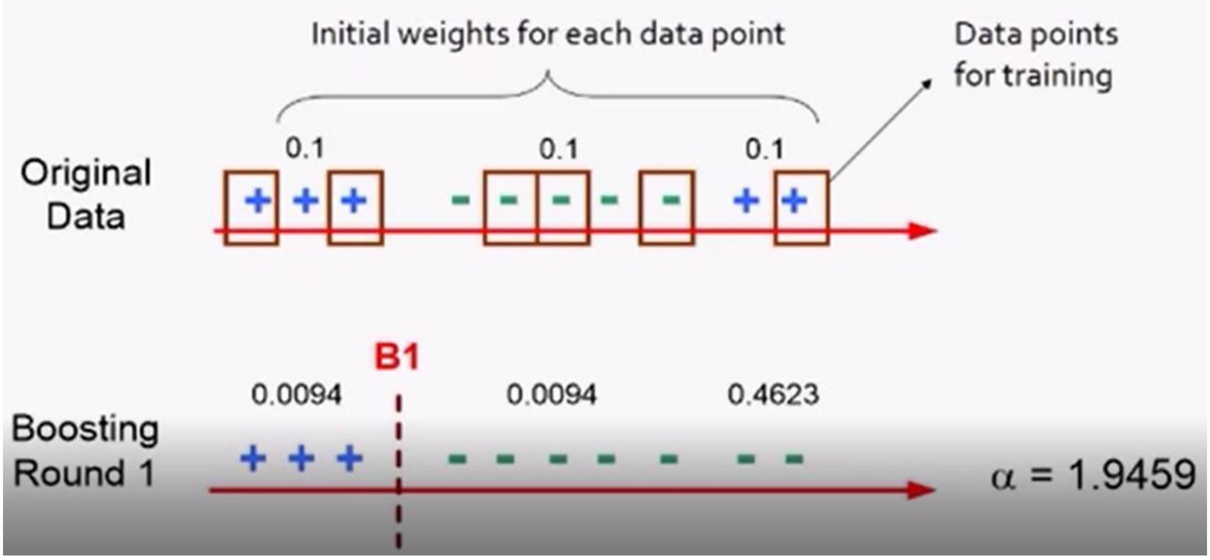
Strong weak classifiers

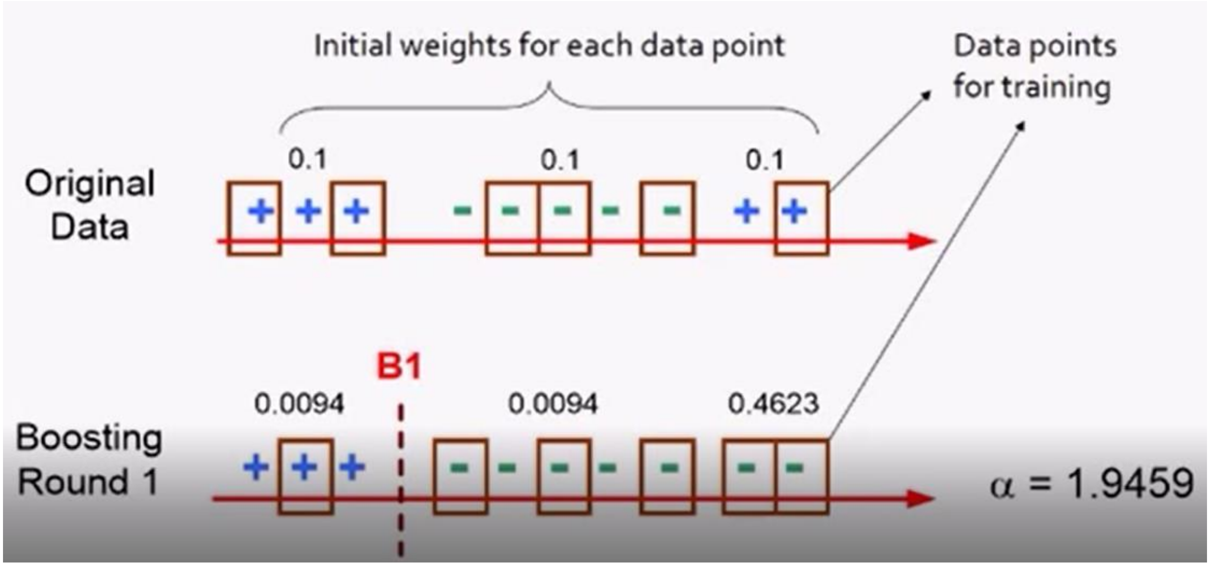
- If each classifiers is (at least slightly) better than random $\epsilon_t < 0.5$
- Ican be shown that AdaBoost will achieve zero training error (expotentially fast):

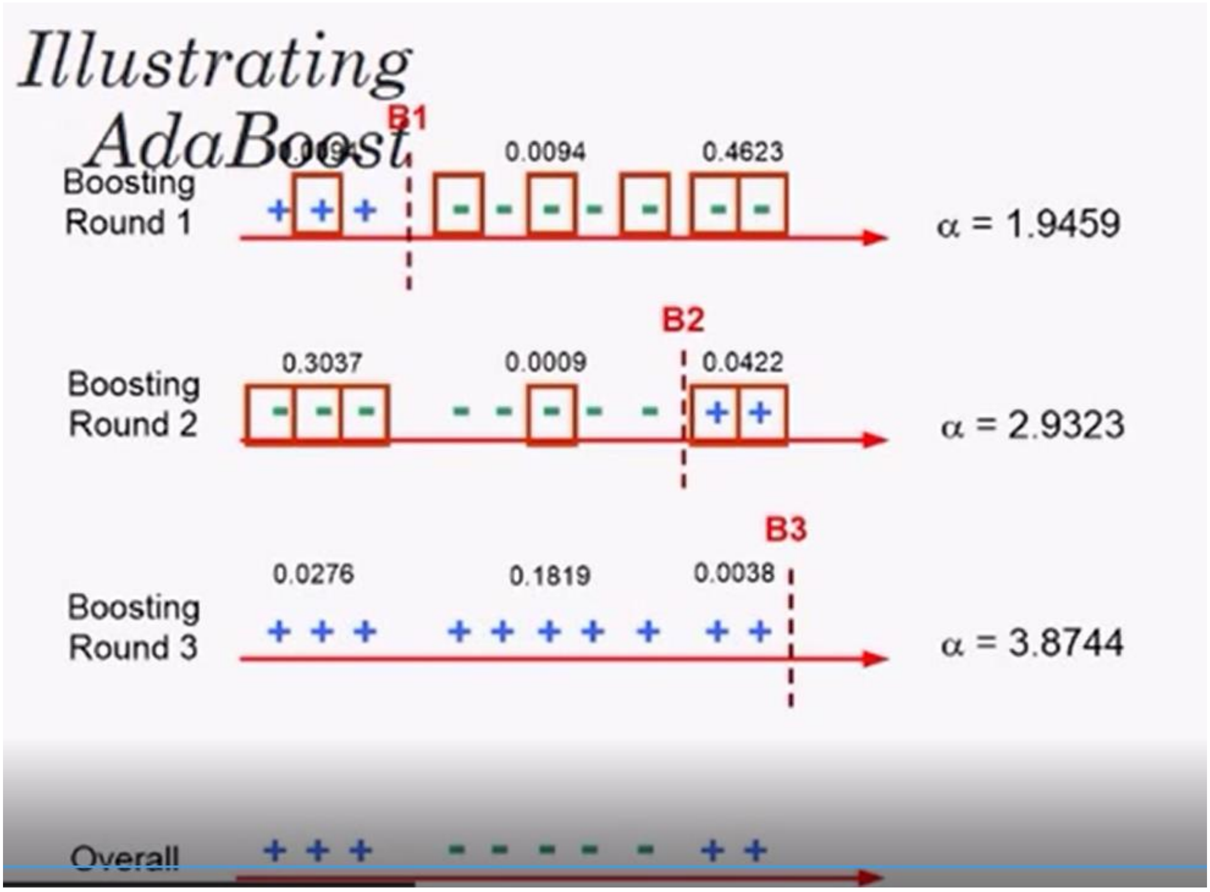
$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \prod_t Z_t \leq exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

Illustrating AdaBoost






Initial weights for each data point
Data points for training
Original Data
0.1
0.1
0.1
B1
Boosting Round 1
0.0094
0.0094
0.4623
α = 1.9459


Illustrating AdaBoost
Boosting Round 1
B1
0.0094
0.4623
α = 1.9459
Boosting Round 2
B2
0.3037
0.0009
0.0422
α = 2.9323
Boosting Round 3
B3
0.0276
0.1819
0.0038
α = 3.8744
Overall