MACHINE INTELLIGENCE

Ensemble Models and Bayesian Learning

K.S.Srinivas

Department of Computer Science and Engineering

Machine Intelligence

Unit III
Emsemble Models and Bayesian Learning

Srinivas K S
Department of Computer Science

Ensemble Learning

An ensemble method is a

- technique that combines the predictions from multiple machine learning algorithms together
- to make more accurate predictions than any individual model.

The learners that we use are usually **weak learners**

- They are among the most powerful techniques in machine learning, often outperforming other methods.
- This comes at the cost of increased algorithmic and model complexity

Ensemble Learning

An ensemble method is a

- technique that combines the predictions from multiple machine learning algorithms together
- to make more accurate predictions than any individual model.

The learners that we use are usually **weak learners**

- They are among the most powerful techniques in machine learning, often outperforming other methods.
- This comes at the cost of increased algorithmic and model complexity

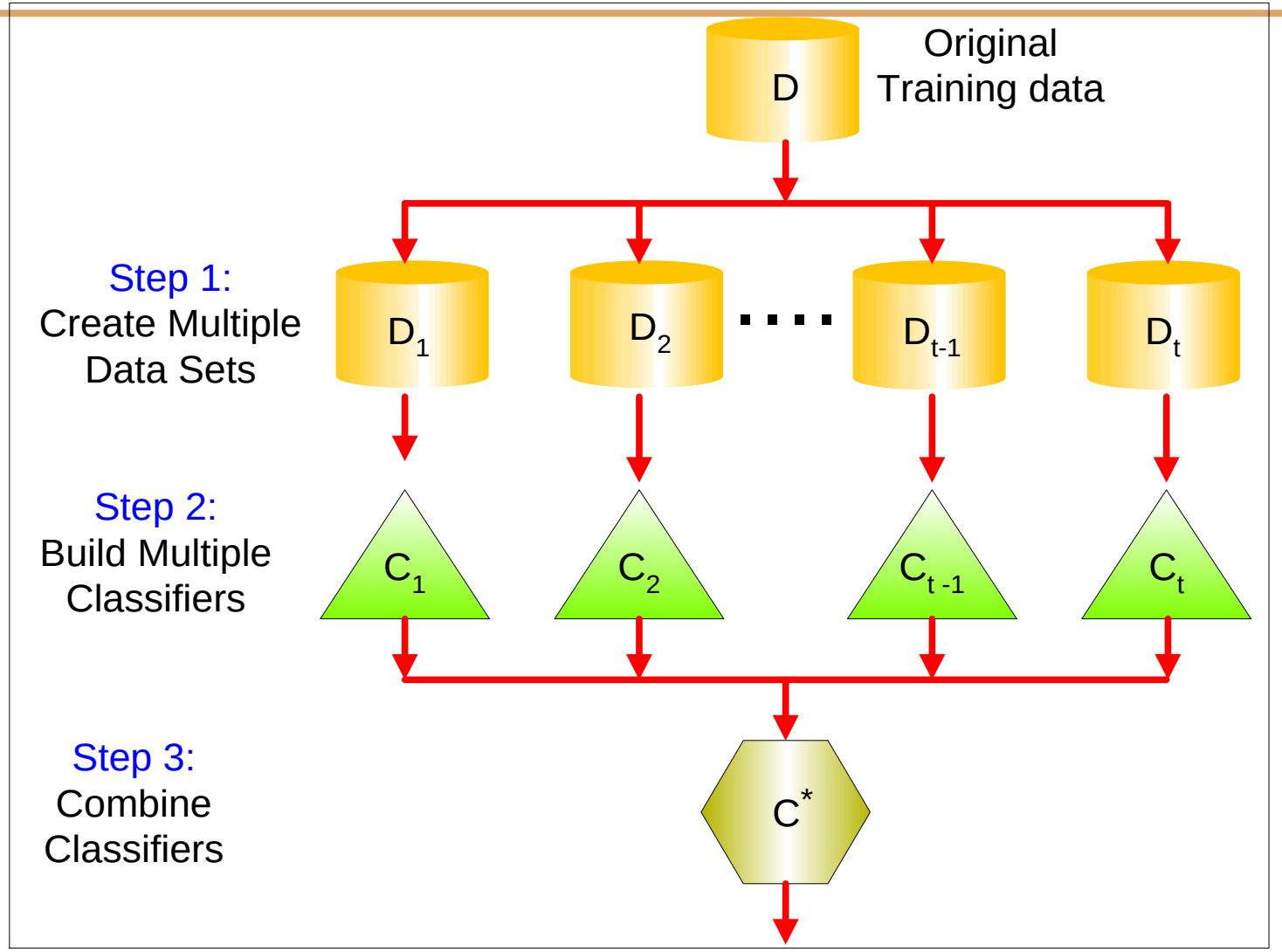
Ensemble Learning

- **The key idea 1** we have learners where the output is slightly better than chance i.e the accuracy is a little better than 50% but not significantly higher
- Multiple learners can be modelled using
  - Different Algorithms
  - Different Hyperparameters of the same algorithm
  - Different subsets of the training data
  - Different features of the training data

- **The key idea 2** They construct multiple, diverse predictive models from adapted versions of the training data (most often reweighted or resampled);

- They combine the predictions of these models in some way, often by simple averaging or voting (possibly weighted).

General Approach

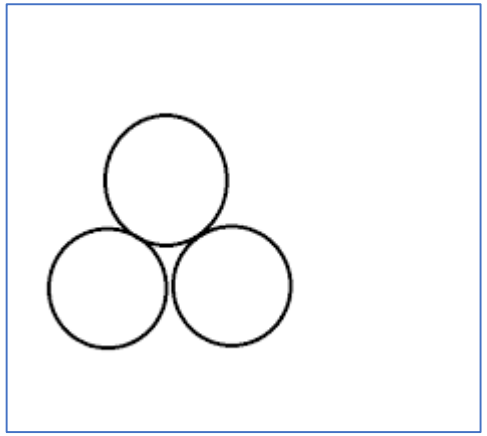


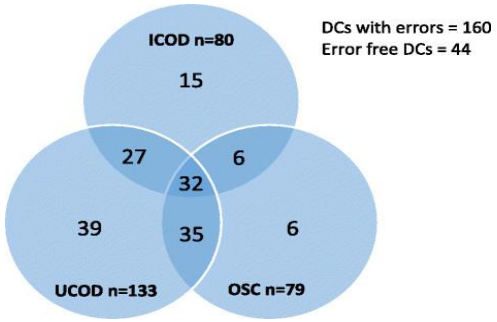
General Approach

- We have seen that decision trees earlier have a tendency to overfit i.e they have high variance
- We could offcourse prune trees but is often difficult.
- Ensemble learning ensures that the combined output of several weak learners produce a final model that has low variance
- Given a set of n independent observations $Z1, \ldots, Zn$, each with variance σ^2, the variance of the mean $\hat{Z}$ of the observations is given by σ^2/n.
- In other words, **averaging a set of observations reduces variance**.

Intuition behind ensemble learning

- Lets take the example of the 3 learners on the right.
- Let the box be the instance space and the errors produced by each of the learners marked as circles
- Lets take an arbitrary point marked in pink and make a prediction
- Now the Red makes an error on the sample but the Green and Blue guys get it right
- The average variance is lower.





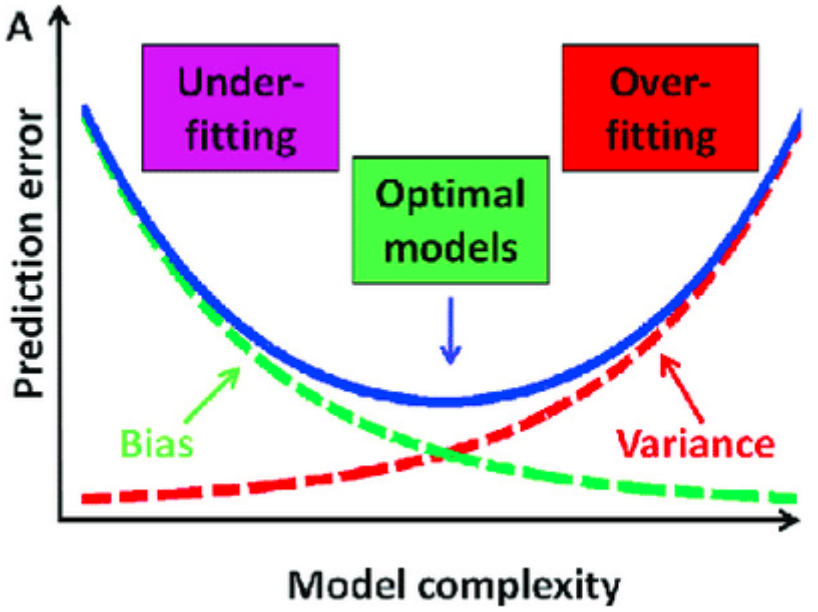
Key Concept: The errors made must be independent

1. But that's not always the case. Look at this picture .
2. The intersection areas could actually result in the voting being wrong for the pink points

Bias Variance - Recap

- A model with high bias is too simple and has low number of predictors
- Due to which it is unable to capture the underlying pattern of data.
- It pays very little attention to the training data and oversimplifies the model. This leads to high error on training and test data.
- Any model which has very large number of predictors will end up being a very complex model
  - which will deliver very accurate predictions for the training data that it has seen already but this complexity
  - makes the generalization of this model to unseen data very difficult i.e a high variance model.

Low Bias and Low Variance



Source:https://towardsdatascience.com/holy-grail-for-bias-variance-tradeoff-overfitting-underfitting-7fad64ab5d76

Bias and Variance

- We have seen in neural networks that if we have very large number of epochs we may overfit
- Low Bias – High flexibility (DT/ANN)
- High Variance – we give different subsets of training data we get different models
- More flexible representations(Low Bias) have High variance
- More powerful representations have high variance
- We want to have a low bias and low variance model

.

The ensemble itself will produce a model with low bias and low variance as illustrated earlier

In fact even if the individual learners have high bias the new combined learner will have a low bias

The hypothesis of the new learner may not even be in the HS of the individual learners

Basics models perform not so well by themselves either because they have a high bias (low degree of freedom models, for example) or because they have too much variance to be robust (high degree of freedom models, for example)

Many weak learners increase our confidence

- Ensemble prevent overfitting
- We don’t need to worry about stopping criteria
- Lets assume we have n learners in a binary classification problem
- If all of them have an accuracy of 0.7 and predict the same class for a given instance, what would your confident be on the prediction

$$C = [1 - \{1 - A\}^n]$$

- In Reality not all learners would predict the same nor would they have the same accuracy
- Lets assume that n1 of those learners predict class1 and n2 class2
- Lets also assume that with no loss of generality that n1 > n2 , meaning that if we took a voting the prediction would always be class 1

Combining ensemble learners

- Learners can be unweighted
- Learners can be weighted wt d_t accuracy , 1/variance of the learner
- For making a prediction using weights

Challenges in ensemble learners

- The critical point of ensemble learners is that they need to be independent
- **averaging a set of observations reduces variance**.
- Which can be achieved by either using different subsets or different learners .
- Shall see this in the next session – Bagging and Boosting

Types of Ensemble Methods

- Manipulate data distribution
  - Example: bagging, boosting
- Manipulate input features
  - Example: random forests
- Manipulate class labels
  - Example: error-correcting output coding

Machine Intelligence

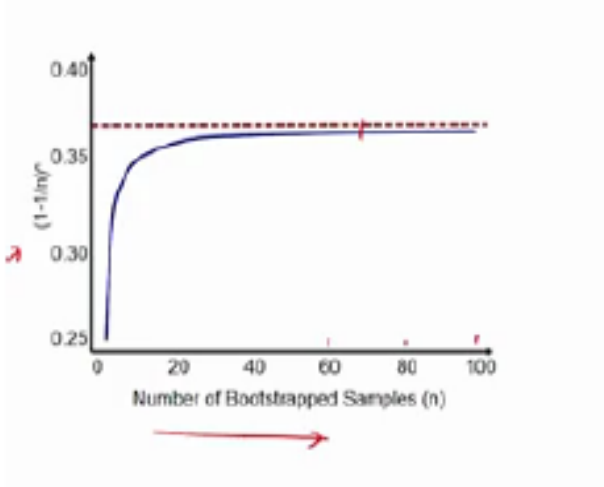
Unit III
Bagging

Srinivas K S
Department of Computer Science

Bagging

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- One way of having different learners to have independent errors is by splitting the data into subsets and passing them to different learners
- But since training instances could be small we may end up with overfitting and high variance
- Instance we would could randomly sample from the data set creating new data sets of the same size as the original or a very large fraction of the data set with replacement
- This method is called bootstrap aggregation or bagging
- It has been shown when we create a data set as described above we would have close to 67% of data from the original data set about 33% of original data is not selected for a sufficiently large sample size.

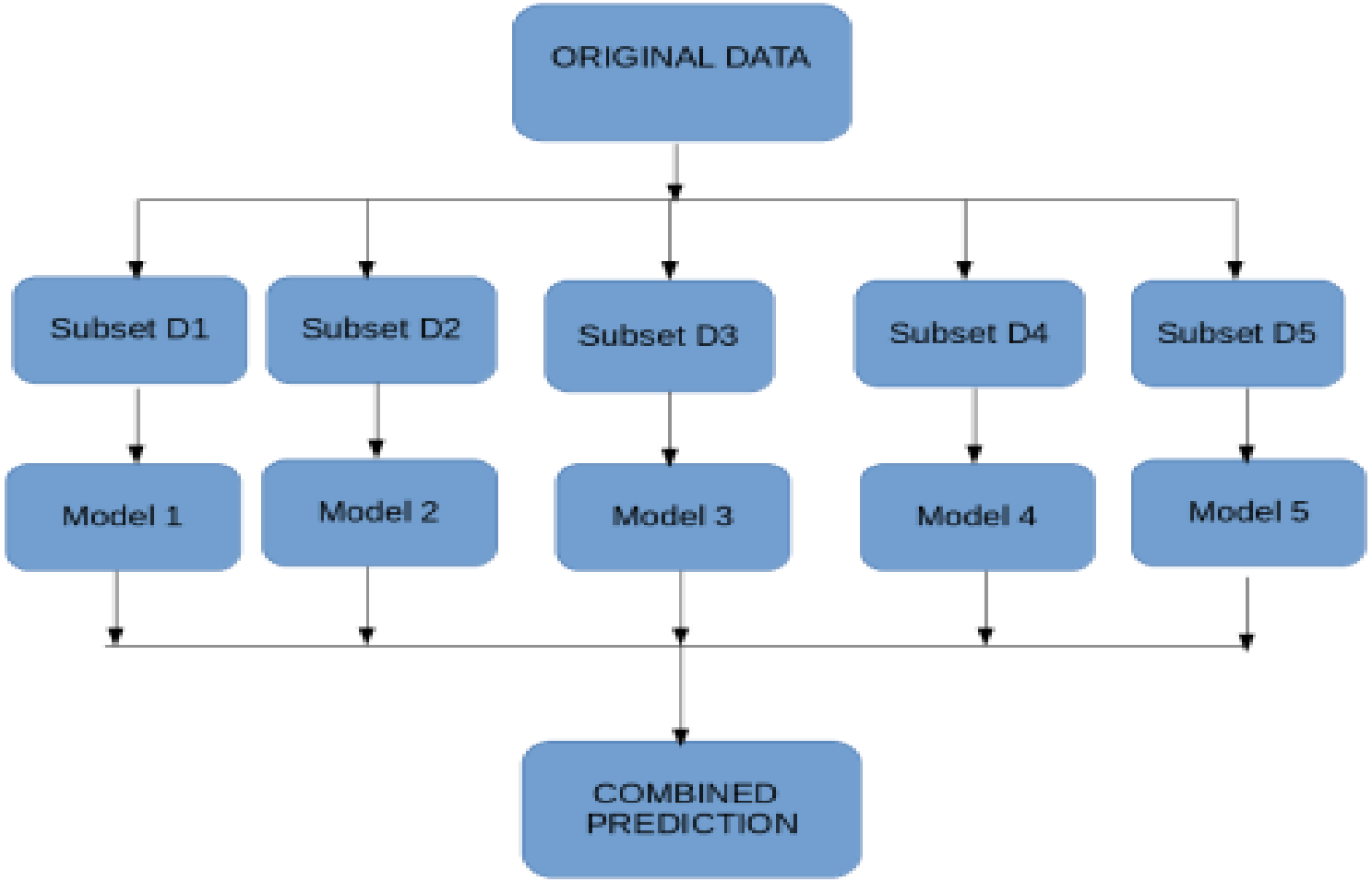


- P(Data not being selected) = $(1-(1/n))^n$

Bagging

- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base model (weak model) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models. (Voting or averaging)

Bagging


ORIGINAL DATA
Subset D1
Subset D2
Subset D3
Subset D4
Subset D5
Model 1
Model 2
Model 3
Model 4
Model 5
COMBINED PREDICTION

Bagging Error Calculation

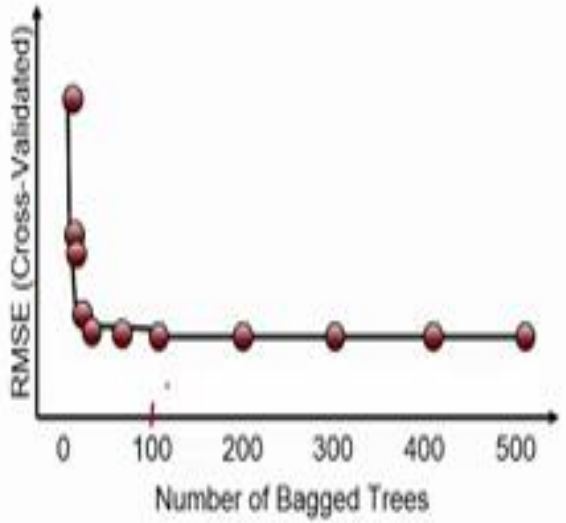
- We can do K-fold cross validation for error calculation
- Typically 1/3 of samples are left out in every subset.
- These are the out of bag examples
- All you have to do is measure the error on the unused sample in those learners that did not use them
- The average of the error from the learners gives us the error for that sample
- you can accumulate their error overall those data points that are out of bag and calculate an average

How many learners and Advantages

- Most research has shown that about 100 learners are good enough

You can formulate these output probabilities

- Bagging performance improvements increase with more trees



- Easy to interpret and implement
- Heterogeneous input data allowed, no preprocessing required

Specific to bagging:

- Less variability than decision trees
- Can grow trees in parallel

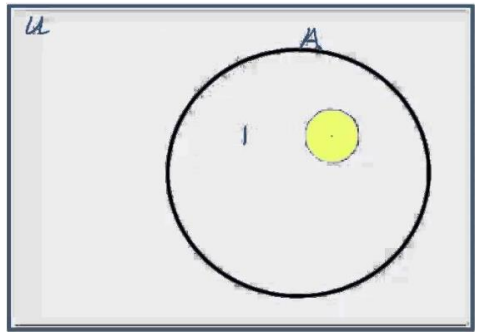
Machine Intelligence

Unit III
Boosting

Srinivas K S
Department of Computer Science

Boosting - Preamble

- Let different learners have different weights (earlier stated by accuracy or by variance)
- Let each learner progressively learn from the previous learner



1. Let U be the instance space and A be the training data
2. Let a Hypothesis h1 misclassify the instances in yellow

3. By learning progressively I mean the 2nd learner h2 is told make other errors but ensure that the h1 misclassified instances are learnt correctly

4. By giving higher weights to those sample
5. Ensure the weights of the samples that were got right reduced
6. Make sure the sum of all weights add up to 1

Boosting vs Bagging

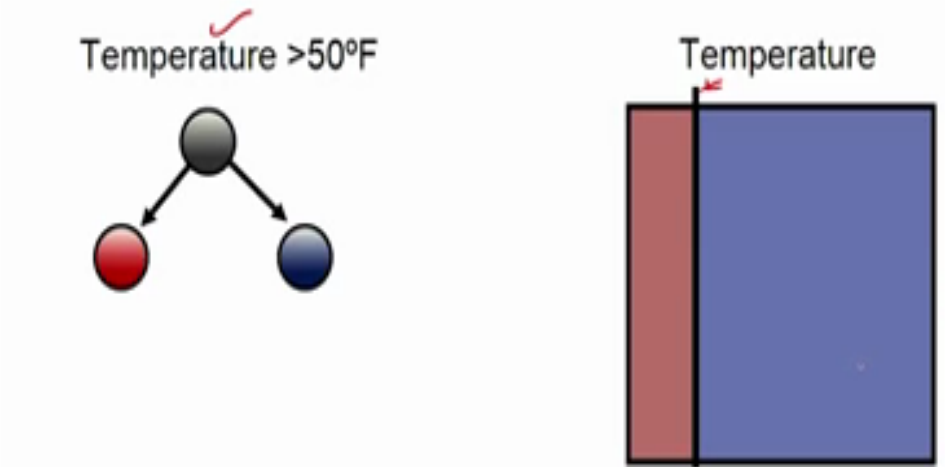
- In the case of Bagging, any element has the same probability to appear in a new data set.

- However, for Boosting the observations are weighted and therefore some of them will take part in the new sets more often.

- One of the most popular Boosting techniques is the “Adaboost”

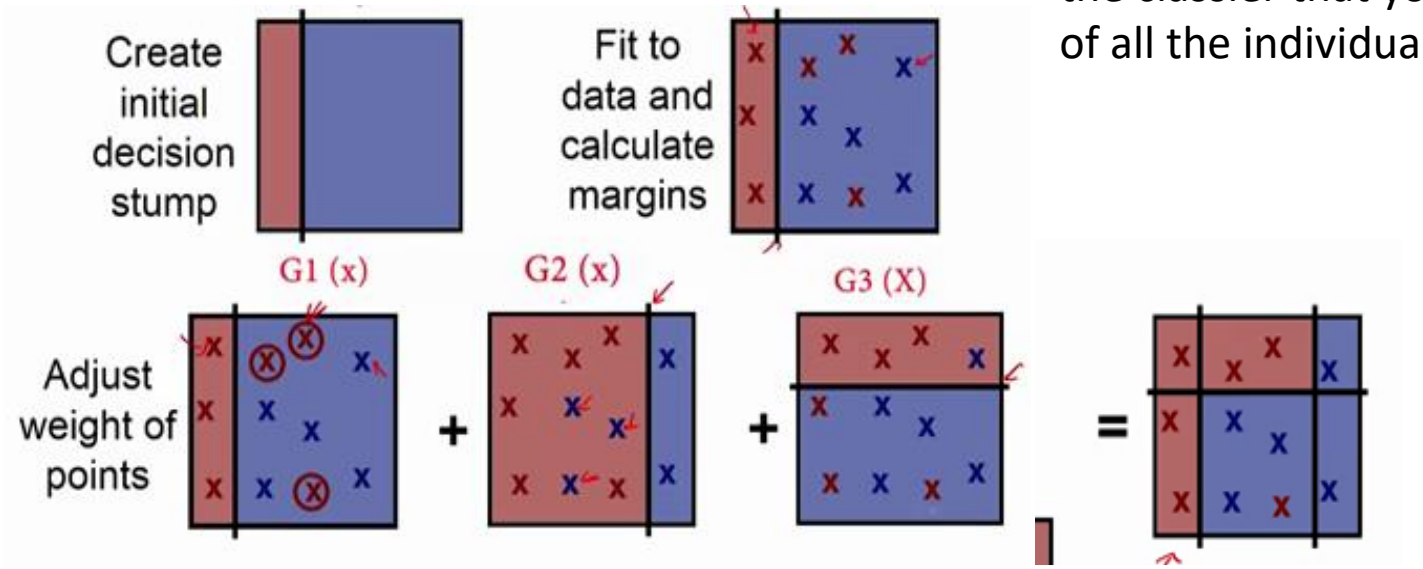
Boosting Overview

- Most example on boosting use a weak learner called a decision stump
- This has one node and based on the value does a binary split
- Boosting approach instead *learns slowly and incrementally*



Boosting Overview

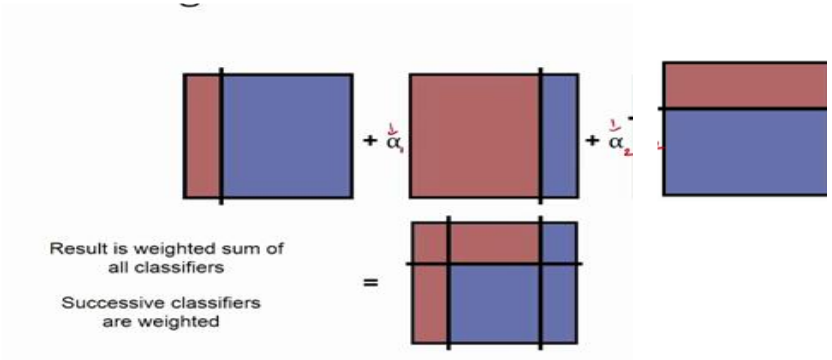
- Most example on boosting use a weak learner called a decision stump



- These red crosses on the right are basically mis-classifications
- Adjust the weights of those points.
- Further classify after assigning weights.
- Ensure the weights of the samples that were got right reduced
- the classier that you use in the end is basically the summation of all the individual classifiers

Boosting Overview

- There is chance to overfit
- have a weighting factor here α. Have $\alpha 1$, $\alpha 2$ and $\alpha 3$ for each of the learners to get your final decision boundary
- These weights have some mechanism that we will see shortly which is dependent on number of errors made
- Some learners are better than others so give them higher weights



Machine Intelligence

AdaBoost

Srinivas K S
Department of Computer Science

Boosting Algorithms

- Several different boosting algorithms exist
- Different by:
  1. How weights of training instances are updated after each boosting round
  2. How predictions made by each classifier are combined
     - Each boosting round produces one base classifier

AdaBoost

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$

$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

$$H(\vec{x}) \in \{-1, +1\} \quad h_i(\vec{x}) = \{-1, +1\}$$

- AdaBoost is a popular boosting algorithm
- Regarding predictions of final ensemble classifier:
  - Importance of a base classier depends on its error rate

$$\varepsilon_i = \left[\sum_{j=1}^{N} \omega_j I\left(C_i(x_j) \neq y_j\right) \right]$$

$$I(p) = 1 \text{ if predicate } p \text{ is true, and } 0 \text{ otherwise}$$

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

C_i: Base Classifier

ε_i: error rate

α_i: importance of classifier

AdaBoost

- α_i also used to update weight of training examples after each boosting round

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-a_j} & \text{if } C_j(x_i) = y_i \\ \exp^{a_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

j: current round

$j+1$: next round

Z_j: normalization factor

$$\sum_i w_i^{(j+1)} = 1$$

- Increases weights of incorrectly classified instances
- Decreases weights of correctly classified instances

AdaBoost Example

Dataset: 10 instances
Predictor Variable: x
Target Variable: y

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Y	1	1	1	-1	-1	-1	-1	1	1	1

- Initially all instances have equal weights

AdaBoost Example

Initial Weights:

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ω_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$\Sigma=1$

Round 1:

X	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
Y	1	-1	-1	-1	-1	-1	-1	-1	1	1

- Going to learn decision stump base classifier
- What is the model?

Model:

If x <= 0.75 then y = -1

If x > 0.75 then y = 1

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Y	1	1	1	-1	-1	-1	-1	1	1	1
Pred. Y	-1	-1	-1	-1	-1	-1	-1	1	1	1

$$\varepsilon_1 = \left[\sum \omega_j \cdot I\right] = 0.3$$

$$\text{Classifier Importance } \alpha_1 = \frac{1}{2}\ln\left(\frac{1-.3}{.3}\right) \approx 0.4236$$

AdaBoost Example

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ω_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Update:	$0.1 \times e^{0.4236}$	$0.1 \times e^{0.4236}$	$0.1 \times e^{0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$	$0.1 \times e^{-0.4236}$
	0.15275	0.15275	0.15275	0.06547	0.06547	0.06547	0.06547	0.06547	0.06547	0.06547

$\Sigma = .15275 \times 3 + 0.06547 \times 7 = 0.91654$

Normalizing: (diving each by sum)

ω_2	0.167	0.167	0.167	0.071	0.071	0.071	0.071	0.071	0.071	0.071

$\Sigma = 1$

AdaBoost Example

Initial Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	ω_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Round 1:	X	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
	Y	1	-1	-1	-1	-1	-1	-1	-1	1	1
Updated Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	ω_i	0.167	0.167	0.167	0.071	0.071	0.071	0.071	0.071	0.071	0.071
	X	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
	Y	1	1	1	1	1	1	1	1	1	1

Σ=1

Σ=1

- Going to learn decision stump base classifier
- What is the model?

Model:

If x <= 0.05 then y = -1

If x > 0.05 then y = 1

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Y	1	1	1	-1	-1	-1	-1	1	1	1
Pred. Y	1	1	1	1	1	1	1	1	1	1

$$\varepsilon_2 = \left[\sum \omega_j \cdot I\right] = (0.071 * 4) = 0.284$$

$$\text{Classifier Importance } \alpha_2 = \frac{1}{2}\ln\left(\frac{1-.284}{.284}\right) = 0.46235$$

AdaBoost Example

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ω_1	0.167	0.167	0.167	0.071	0.071	0.071	0.071	0.071	0.071	0.071
Update:	0.167 x $e^{-0.4623}$	0.167 x $e^{-0.4623}$	0.167 x $e^{-0.4623}$	0.071 **x** $e^{0.4623}$	0.071 **x** $e^{0.4623}$	0.071 **x** $e^{0.4623}$	0.071 **x** $e^{0.4623}$	0.071 **x** $e^{-0.4623}$	0.071 **x** $e^{-0.4623}$	0.071 **x** $e^{-0.4623}$
	0.1051766	0.1051766	0.1051766	**0.1127267**	**0.1127267**	**0.1127267**	**0.1127267**	0.0447158	0.0447158	0.0447158

Σ= 0.1051766 x 3 + **0.1127267** x 4 + 0.0447158 x 3 = 0.900584

Normalizing: (diving each by sum)

ω_2	0.116787	0.116787	0.116787	0.1251706	0.1251706	0.1251706	0.1251706	0.049652	0.049652	0.049652

Σ= 1

AdaBoost Example

Round	Split Point	Left Class	Right Class
1	0.75	-1	1
2	0.05	-1	1
3	0.3	1	-1

Initial Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
	ω_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	$\Sigma=1$
Round 1:	X	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1	
	Y	1	-1	-1	-1	-1	-1	-1	-1	1	1	
Updated Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
	ω_i	0.167	0.167	0.167	0.071	0.071	0.071	0.071	0.071	0.071	0.071	$\Sigma=1$
Round 2:	X	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	
	Y	1	1	1	1	1	1	1	1	1	1	
Updated Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	
	ω_i	0.116787	0.116787	0.116787	0.1251706	0.1251706	0.1251706	0.1251706	0.049652	0.049652	0.049652	$\Sigma=1$
Round 3:	X	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.8	
	Y	1	1	-1	-1	-1	-1	-1	-1	-1	1	

- Going to learn decision stump base classifier
- What is the model?

Model:
If x <= 0.3 then y = 1
If x > 0.3 then y = -1

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Y	1	1	1	-1	-1	-1	-1	1	1	1
Pred. Y	1	1	1	-1	-1	-1	-1	-1	-1	-1

$$\varepsilon_1 = \left[\sum \omega_j \cdot I\right] = (0.049652 * 3) = 0.148956$$

$$\text{Classifier Importance } \alpha_3 = \frac{1}{2}\ln\left(\frac{1-0.148956}{0.148956}\right) = 0.8714064$$

AdaBoost Prediction

- Prediction made by each base classifier C_i is weighted by α_i

AdaBoost Example

Round	Split Point	Left Class	Right Class	α
1	0.75	-1	1	0.4236
2	0.05	1	1	0.46235
3	0.3	1	-1	0.8714064

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	0.9101564	0.9101564	0.9101564	-0.8326564	-0.8326564	-0.8326564	-0.8326564	0.0145436	0.0145436	0.0145436
Sign	1	1	1	-1	-1	-1	-1	1	1	1

$$-1 \times 0.4236 + 1 \times 0.46235 + 1 \times 0.8714064 = 0.9101564$$

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Y	1	1	1	-1	-1	-1	-1	1	1	1

Boosting Algorithm

From Tan, Alg. 5.7

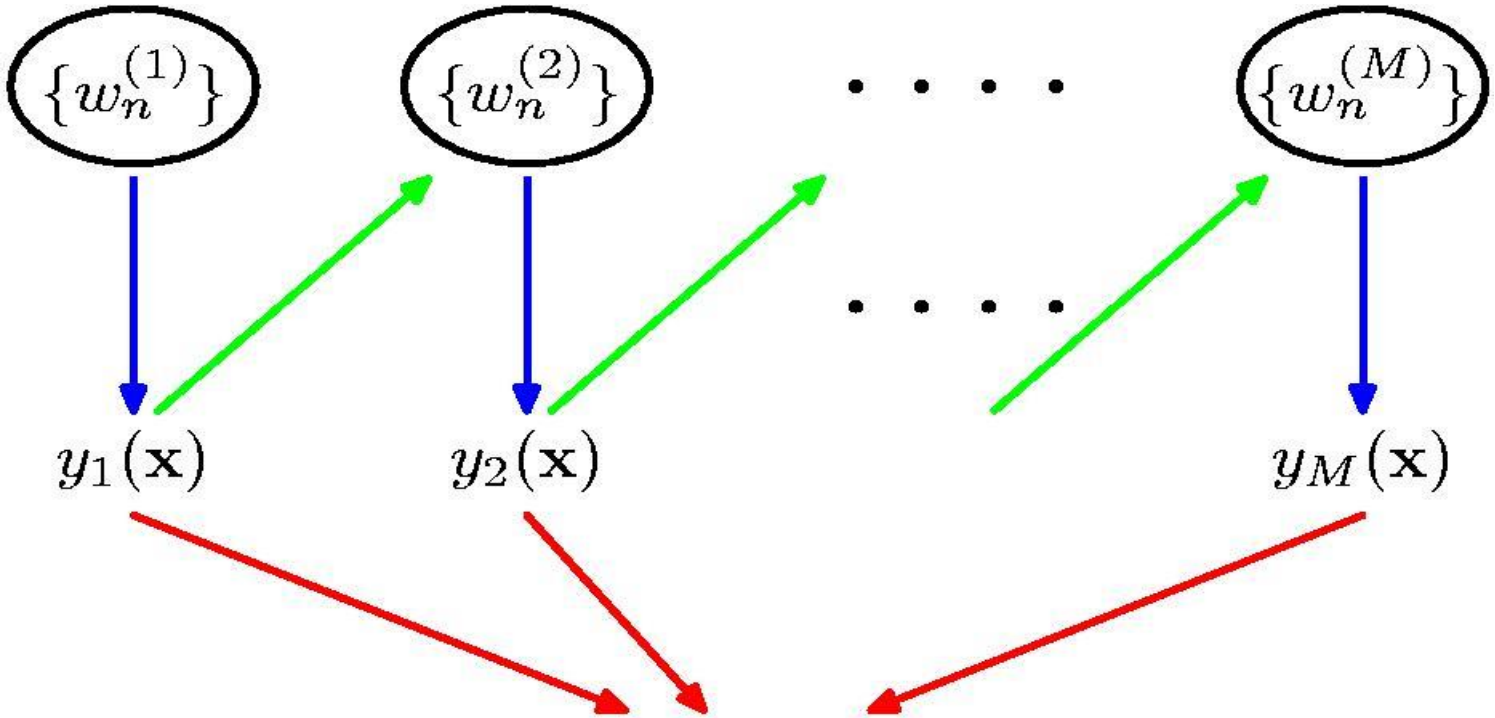
PES UNIVERSITY

Algorithm 5.7 AdaBoost algorithm.

1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Initialize the weights for all N examples.}
2: Let k be the number of boosting rounds.
3: **for** $i = 1$ to k **do**
4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
5: Train a base classifier C_i on D_i.
6: Apply C_i to all examples in the original training set, D.
7: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta\left(C_i(x_j) \neq y_j\right)\right]$ {Calculate the weighted error.}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {Reset the weights for all N examples.}
10: Go back to Step 4.
11: **end if**
12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13: Update the weight of each example according to Equation 5.69.
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y))$.

$$\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta\left(C_i(x_j) \neq y_j\right)\right]$$

Schematic illustration of Boosting



$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

Adaboost: Example2

x1	x2	Decision
2	3	true
2	2	true
4	6	true
4	3	false
4	1	false
5	7	true
5	3	false
6	5	true
8	6	false
8	2	false

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- 1st weight – instance weight
- When we start the algorithm have the same weight
- So assign weights as 1/N = 1/10 (ten sample)
- Since we are using a binary classifier convert true +1 and false to -1

x1	x2	actual	weight	weighted_actual
2	3	1	0.1	0.1
2	2	1	0.1	0.1
4	6	1	0.1	0.1
4	3	-1	0.1	-0.1
4	1	-1	0.1	-0.1
5	7	1	0.1	0.1
5	3	-1	0.1	-0.1
6	5	1	0.1	0.1
8	6	-1	0.1	-0.1
8	2	-1	0.1	-0.1

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- Create many decision stumps such as if x1 < 2.1 it to be +1 and X1>= 2.1 as -1
- You can create many such stumps and choose the one with the lowest error rate
- Assume that the decision stump above is the best one (Not true) but for this example lets just assume
- Lets make a prediction and calculate the error rate

x1	x2	actual	weight	weighted_actual	prediction	loss	weight * loss
2	3	1	0.1	0.1	1	0	0
2	2	1	0.1	0.1	1	0	0
4	6	1	0.1	0.1	-1	1	0.1
4	3	-1	0.1	-0.1	-1	0	0
4	1	-1	0.1	-0.1	-1	0	0
5	7	1	0.1	0.1	-1	1	0.1
5	3	-1	0.1	-0.1	-1	0	0
6	5	1	0.1	0.1	-1	1	0.1
8	6	-1	0.1	-0.1	-1	0	0
8	2	-1	0.1	-0.1	-1	0	0

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- Evaluate the error for the m^{th} classifier. Here $w_n^{(m)}$ is the weight of the n^{th} data instance in the m^{th} iteration. The Identity function:

  I(a,b) = 1 if a != b and = 0 otherwise.

$$\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

- Then evaluate the value of the classifier using:
- $\alpha_m = \frac{1}{2} * \ln\{(1 - \varepsilon_m)/\varepsilon_m\}$

Adaboost

- Sum of weight times loss column stores the total error
- It is 0.3 in this case.
- Compute the Stump weight

$$\alpha_m = \frac{1}{2} * \ln\{(1 - \varepsilon_m)/\varepsilon_m\}$$

alpha = ln[(1-epsilon)/epsilon] / 2 = ln[(1 – 0.3)/0.3] / 2
alpha = 0.42

- Remember our final H(x) is a weighted sum of individual hypothesis
- So alpha 1 is 0.42 for our first decision stump

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$

$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

where:

$$H(\vec{x}) \in \{-1, +1\} \quad h_i(\vec{x}) = \{-1, +1\}$$

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{-\alpha_s}$$ For correctly classified samples (we ***reduce*** their weight)

$$w_i^{s+1} = \frac{w_i^s}{N^s} \cdot e^{+\alpha_s}$$ For incorrectly classified samples (we ***increase*** their weight)

- We'll use alpha to update weights in the next round.

alpha1 = 0.42

- $w_{i+1} = w_i$ * math.exp(-alpha * actual * prediction) where i refers to instance number.

- Also, sum of weights must be equal to 1. That's why, we have to normalize weight values. Dividing each weight value to sum of weights column enables normalization.

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.1	1	0.065	0.071
2	2	1	0.1	1	0.065	0.071
4	6	1	0.1	-1	0.153	0.167
4	3	-1	0.1	-1	0.065	0.071
4	1	-1	0.1	-1	0.065	0.071
5	7	1	0.1	-1	0.153	0.167
5	3	-1	0.1	-1	0.065	0.071
6	5	1	0.1	-1	0.153	0.167
8	6	-1	0.1	-1	0.065	0.071
8	2	-1	0.1	-1	0.065	0.071

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- In the next round I choose x1 < 3.5 as -1 and x1>= 3.5 as +1
- Off course I use the new weights this time.

x1	x2	actual	weight	prediction	loss	weight * loss
2	3	1	0.071	-1	1	0.071
2	2	1	0.071	-1	1	0.071
4	6	1	0.167	1	0	0.000
4	3	-1	0.071	-1	0	0.000
4	1	-1	0.071	-1	0	0.000
5	7	1	0.167	1	0	0.000
5	3	-1	0.071	-1	0	0.000
6	5	1	0.167	1	0	0.000
8	6	-1	0.071	1	1	0.071
8	2	-1	0.071	-1	0	0.000

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- You can calculate epsilon, alpha and new weights using the same procedure
- epsilon = 0.21, alpha = 0.65
- And find weights for the next round

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.071	-1	0.137	0.167
2	2	1	0.071	-1	0.137	0.167
4	6	1	0.167	1	0.087	0.106
4	3	-1	0.071	-1	0.037	0.045
4	1	-1	0.071	-1	0.037	0.045
5	7	1	0.167	1	0.087	0.106
5	3	-1	0.071	-1	0.037	0.045
6	5	1	0.167	1	0.087	0.106
8	6	-1	0.071	1	0.137	0.167
8	2	-1	0.071	-1	0.037	0.045

$$\alpha_s = \ln\left(\frac{1-E^s}{E^s}\right)^{\frac{1}{2}} = \frac{1}{2}\ln\left(\frac{1-E^s}{E^s}\right)$$

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- At each round I update my final hypothesis

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$

$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

- I have given a table here and the calculations for 4 rounds

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	**round 2 prediction**	**round 3 prediction**	**round 4 prediction**
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost

- At each round I update my final hypothesis

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$

$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

$$H(\vec{x}) \in \{-1, +1\} \quad h_i(\vec{x}) = \{-1, +1\}$$

- I have given a table here and the calculations for 4 rounds

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	**round 2 prediction**	**round 3 prediction**	**round 4 prediction**
1	−1	1	1
1	−1	1	1
−1	1	−1	1
−1	−1	−1	1
−1	−1	−1	1
−1	1	−1	1
−1	−1	−1	1
−1	1	−1	1
−1	1	−1	−1
−1	−1	−1	−1

Source: https://sefiks.com/2018/11/02/a-step-by-ste

Adaboost

round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.65	0.38	1.1
round 1 prediction	**round 2 prediction**	**round 3 prediction**	**round 4 prediction**
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

For example, prediction of the 1st instance will be

0.42 x 1 + 0.65 x (-1) + 0.38 x 1 + 1.1 x 1 = 1.25

And we will apply sign function

Sign(1.25) = +1 aka true which is correctly classified.

$$H(\vec{x}) = sign(\alpha_1 h_1(\vec{x}) + \alpha_2 h_2(\vec{x}) + \ldots + \alpha_s h_s(\vec{x}))$$

$$H(\vec{x}) = sign(\sum_i^s \alpha_i h_i(\vec{x}))$$

$$H(\vec{x}) \in \{-1, +1\} \quad h_i(\vec{x}) = \{-1, +1\}$$

Source: https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

Adaboost toy example

Please try a calculation of this as a home assignment

X	0	1	2	3	4	5	6	7	8	9
Y	+	+	+	-	-	-	+	+	+	-

THANK YOU

Srinivas K S

Department of Computer Science & Engineering

srinivasks@pes.edu

Bagging

- Ensemble method that “manipulates the training set”
- *Action:* repeatedly sample *with replacement* according to uniform probability distribution
  - Every instance has equal chance of being picked
  - Some instances may be picked multiple times; others may not be chosen
- *Sample Size:* same as training set
- D_i*:* each bootstrap sample
- *Footnote:* also called bootstrap aggregating

Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different

Bagging Algorithm

Model Generation:

- Let *n* be the number of instances in the training data.
- For each of *t* iterations:
  - Sample *n* instances with replacement from training data.
  - Apply the learning algorithm to the sample.
  - Store the resulting model.

Classification:

- For each of the *t* models:
  - Predict class of instance using model.
- Return class that has been predicted most often.

Bagging Example

Now going to apply bagging and create many decision stump base classifiers.

Dataset: 10 instances
Predictor Variable: x
Target Variable: y

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Y	1	1	1	-1	-1	-1	-1	1	1	1

- Decision Stump: one-level binary decision tree
- Splitting condition will be $x <= k$, where k is the split point
  - Best splits: $x <= 0.35$ or $x <= 0.75$

Bagging Example

- First choose how many “bagging rounds” to perform
  - Chosen by analyst
- We’ll do 10 bagging rounds in this example:

Bagging Example

In each round, create D_i by sampling with replacement

Round 1:

X	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
Y	1	1	1	1	-1	-1	-1	-1	1	1

Learn decision stump.
What stump will be learned?

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

X	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
Y	1	1	1	1	-1	-1	-1	-1	1	1
X	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
Y	1	1	1	-1	-1	1	1	1	1	1
X	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
Y	1	1	1	-1	-1	-1	-1	-1	1	1
X	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
Y	1	1	1	-1	-1	-1	-1	-1	1	1
X	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
Y	1	1	1	-1	-1	-1	-1	1	1	1
X	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
Y	1	-1	-1	-1	-1	-1	-1	1	1	1
X	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
Y	1	-1	-1	-1	-1	1	1	1	1	1
X	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
Y	1	1	-1	-1	-1	-1	-1	1	1	1
X	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
Y	1	1	-1	-1	-1	-1	-1	1	1	1
X	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
Y	1	1	1	1	1	1	1	1	1	1

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.65 then y = -1
If x > 0.65 then y = 1

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.3 then y = 1
If x > 0.3 then y = -1

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.05 then y = -1
If x > 0.05 then y = 1

10 bagging rounds. 10 D_i's. 10 learned models.

Classify test instance by using each base classifier and taking majority vote.

Bagging Example

10 test instances. Let's see how each classifier votes:

PES UNIVERSITY ONLINE

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	X=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	-1	-1	-1	-1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-8	0	0	0
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True	1	1	1	-1	-1	-1	-1	1	1	1