Ensemble Methods

Réda DEHAK
http://isml.dehak.org

1

Contents

- Bias-Variance Tradeoff
- Ensemble Methods that minimize variance
  - Bagging
  - Random Forests
- Ensemble Methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

2

Supervised Learning

- Goal: Learn a predictor $h(x)$
  - using training dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
  - Low error (high Accuracy)
- Different classifiers:
  - None of the classifiers is perfect
  - Complementary:
    Examples which are not correctly classified by one classifier may be correctly classified by the other classifiers
- Potential Improvements? Utilize the complementary property

3

Basic idea

- Concordet's jury theorem (1785):

  imagine that a group of people has to select between two choices (from which only one is correct). They vote independently, and the probability that they vote correctly is p. The votes are combined by the majority rule. Let P denote the probability that the majority vote is correct.
  - Concordet's theorem says that if $p > 0.5$ then $P \longrightarrow 1$ if the number of votes goes to infinity
- This means that the crowd is more clever than the individuals under relatively weak assumptions
  - Each individual must be correct with $p > 0.5$ (better than random guessing)
  - Their should make independent decisions
- Now, how can we apply this idea in machine learning?

4

Ensembles of Classifiers

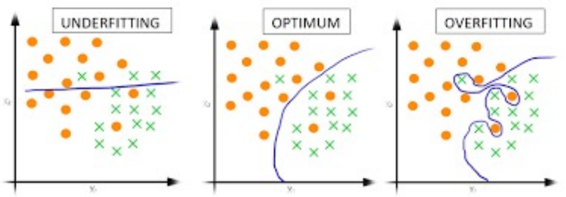
Combine the classifiers to improve the performance

- Ensembles of Classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting

Réda DEHAK 5

5

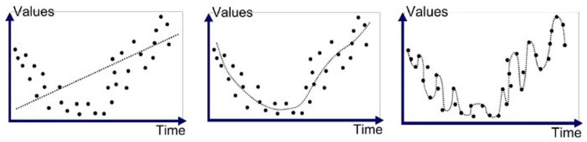
Under – Overfitting *vs.* Bias/Variance



Réda DEHAK 6

6

Under – Overfitting *vs.* Bias/Variance



Underfitted Good Fit/Robust Overfitted

7

Generalization Error

- True data distribution: $P(x, y)$
  unknown
- Train a predictor: $h(x) = y$
  - Using training dataset S=$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ sampled from $P(x, y)$
  - Minimize the cost function
    E.g. MSE
$$\frac{1}{N}\sum_{\substack{i=1 \\ (x_i,y_i)\in S}}^{N} \|h(x_i) - y_i\|^2$$
- Generalization Error:
$$\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\|h(x) - y\|^2]$$

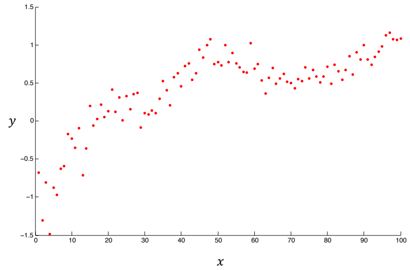
8

Bias/Variance Tradeoff

- Treat $h(x|S)$ as a random function which Depends on training dataset S.
- Expected generation Error:
$$\mathcal{L}(h) = E_S\left[E_{(x,y)\sim P(x,y)}[\|h(x) - y\|^2]\right]$$
Expected generalization error over the randomness of S

9

Example



10

$\mathbf{h}(\boldsymbol{x}|\boldsymbol{S})$

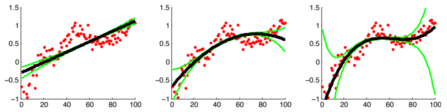
11

$\mathbf{h}(\boldsymbol{x}|\boldsymbol{S})$

12

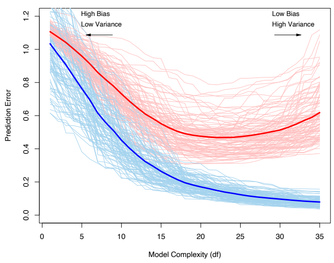
$\mathbf{h}(x|S)$

13



14

Bias/Variance Tradeoff



Réda DEHAK

15

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - **Good:** Low variance, don't usually overfit
  - **Bad:** High bias, can't solve hard learning problems
- **Sophisticated learners**
  - Kernel SVMs, Deep Neural Nets, Deep Decision Trees
  - **Good:** Low bias, have the potential to learn with Big Data
  - **Bad:** High variance, difficult to generalize
- **Can we make combine these properties**
  - In general, No!!
  - But often yes...

Réda DEHAK 16

16

Contents

- Bias-Variance Tradeoff
- Ensemble Methods that minimize variance
  - Bagging
  - Random Forests
- Ensemble Methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Réda DEHAK 17

17

Reduce Variance

- Averaging reduces variance:

An average of M i.i.d. random variables, each with variance σ^2, has variance:

$$\mathrm{VAR}\left(\frac{1}{M}\sum_{i=1}^{M} x_i\right) = \frac{\sigma^2}{M}$$

If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\mathrm{VAR}\left(\frac{1}{M}\sum_{i=1}^{M} x_i\right) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$$

Réda DEHAK 18

18

Strong vs. weak learners

- Strong learner: we seek to produce one classifier for which the classification error can be made arbitrarily small
  - So far we were looking for such methods
- Weak learner: a classifier which is just better than random guessing
  - Now this will be our only expectation
- Ensemble learning: instead of creating one strong classifier, we create a huge set of weak classifiers, then we combine their outputs into one final decision
  - According to Concordet's theorem, under proper conditions we can expect that the ensemble model can attain an error rate that is arbitrarily close to zero
  - While creating a lot of weak classifiers is hopefully a much easier task than to create one strong classifier

19

Ensembles of Classifiers

Combine the classifiers to improve the performance

- Ensembles of Classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting

Different classifiers ⟺ independent

Réda DEHAK 20

20

How to produce diverse classifiers?

- We can combine *different learning algorithms* ("hybridization")
  - E.g. we can train a GMM, an SVM, a k-NN,… over the same data, and then combine their output
- We can combine the same learning algorithm trained several times over the same data
  - This works only if there is some random factor in the training method
  - E.g.: neural networks trained with *different random initialization*
- We can combine the same learning algorithm trained over *different subsets of the training data*
  - We can also try using *different subsets of the features*
  - Or *different subsets of the target classes* (multi-class task, lot of classes)
- For certain algorithms we can use the same algorithm over the same data, but with a *different weighting over the data instances*

21

Ensemble Methods

- Instead of learning a single predictor, learn **many predictors**
- **Output class:** (Weighted) combination of each predictor
- With sophisticated learners
  - Uncorrelated errors → expected error goes down
  - On average, do better than single classifier!
  - **Bagging**
- With weak learners
  - each one good at different parts of the input space
  - On average, do better than single classifier!
  - **Boosting**

Réda DEHAK 22

22

Bagging: Bootstrap Aggregation *Leo Breiman (1994)*

- **Goal:** reduce variance
- **Ideal setting:** many training sets S' (sampled independently)
  - Train model using each S'
  - Average predictions

Variance reduces linearly
Bias unchanged

Réda DEHAK 23

23

Bagging: Bootstrap Aggregation *Leo Breiman (1994)*

- **Goal:** reduce variance
- **Ideal setting:** many training sets S' (sampled independently)
  - Train model using each S'
  - Average predictions

Variance reduces sub-linearly
(Because S' are correlated)
Bias often increases slightly

Réda DEHAK 24

24

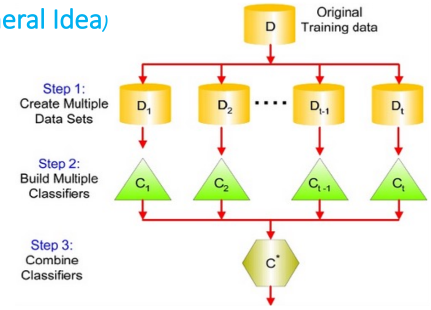
Bagging: Bootstrap Aggregation *Leo Breiman (1994)*

- Take repeated bootstrap samples from training set D
- Bootstrap sampling: Given set D containing N training examples, create D' by drawing N examples at random with replacement from D.
- **Bagging:**
  - Create k bootstrap samples $D_1, D_2, \dots, D_k$.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote/average.

Réda DEHAK 25

25

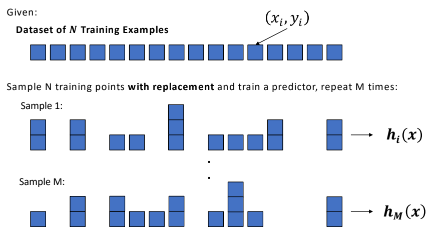
General Idea)



Réda DEHAK 26

26

Bagging: Bootstrap Aggregation *Leo Breiman (1994)*



At test time, output the (weighted) average output of these predictors.

27

27

When To Use Bagging

- In practice, completely uncorrelated predictors don't really happen, but there also wont likely be perfect correlation either, so bagging may still help!

- Use bagging when…
  - … you have overfit sophisticated learners (averaging lowers variance)
  - … you have a somewhat reasonably sized dataset
  - … you want an extra bit of performance from your models

Random Forest *Ho and Kam (1995) Leo Breiman (2001)*

- **Decision Tree**
  - High Variance
  - Low Bias
  - Adapted for bagging
- **Random Forest**
  - Ensemble method specifically designed for decision tree classifiers
  - Introduce two sources of randomness: "Bagging" and "Random input vectors"
  - **Bagging method:** each tree is grown using a bootstrap sample of training data
  - **Random vector method:** At each node, best split is chosen from a random sample of d attributes instead of all attributes

Random Forest *Ho and Kam (1995) Leo Breiman (2001)*

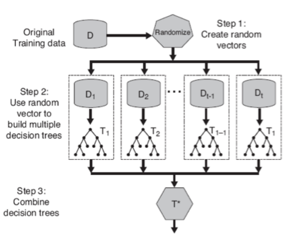


Figure 5.40. Random forests.

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

31

31

Contents

- Bias-Variance Tradeoff
- Ensemble Methods that minimize variance
  - Bagging
  - Random Forests
- Ensemble Methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Réda DEHAK

32

32

Boosting

Core Idea: Combine multiple weak learners to reduce error/bias by reweighting hard examples!

Réda DEHAK

33

33

Boosting

- Bagging created a diversity of base learners by creating different variants of the training dataset randomly
  - However, we do not have direct control over the usefulness of the newly added classifiers
- We would expect a better performance if the learners also complemented each other
  - They would have “expertise” on different subsets of the data
  - So they would work better on different subsets
- The basic idea of boosting is to generate a series of base learners which complement each other
  - For this, we will force each learner to focus on the mistakes of the previous learner

34

Boosting

- We represent the importance of each sample by assigning weights to the samples
  - Correct classification → smaller weights
  - Misclassified samples → larger weights
- The weights can influence the algorithm in two ways
  - Boosting by sampling: the weights influence the resampling process
    - This is a more general solution
  - Boosting by weighting: the weights influence the learner
    - Works only with certain learners
- Boosting also makes the aggregation process more clever: We will aggregate the base learners using weighted voting
  - Better weak classifier gets a larger weight
  - We iteratively add new base learners, and iteratively increase the accuracy of the combined model

35

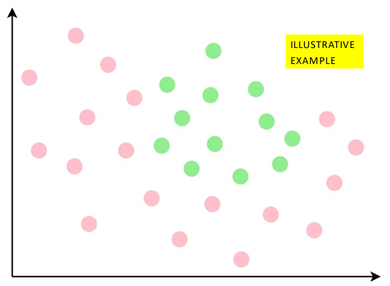
AdaBoost (Adaptive Boosting) Freund and Schapire (1997)

Training:
- For all $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$, initialize $p_1^t = 1/N$ — Start with equal weights
- For all base-learners $j = 1, \ldots, L$
  - Randomly draw $\mathcal{X}_j$ from $\mathcal{X}$ with probabilities p_j^t — Random resampling
  - Train d_j using $\mathcal{X}_j$
  - For each (x^t, r^t), calculate $y_j^t \leftarrow d_j(x^t)$ — Estimated labels
  - Calculate error rate: $\epsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$ — Error is the weighted sum of not hit samples
  - If $\epsilon_j > 1/2$, then $L \leftarrow j - 1$; stop — Not a weak learner, must stop
  - $\beta_j \leftarrow \epsilon_j / (1 - \epsilon_j)$
  - For each (x^t, r^t), decrease probabilities if correct:
    - If $y_j^t = r^t$ $p_{j+1}^t \leftarrow \beta_j p_j^t$ Else $p_{j+1}^t \leftarrow p_j^t$
  - Normalize probabilities:
    - $Z_j \leftarrow \sum_t p_{j+1}^t; \quad p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

Testing:
- Given x, calculate $d_j(x), j = 1, \ldots, L$ — Weighted aggregation of the classifiers
- Calculate class outputs, $i = 1, \ldots, K$:
  - $y_i = \sum_{j=1}^L \left(\log \frac{1}{\beta_j}\right) d_{ji}(x)$

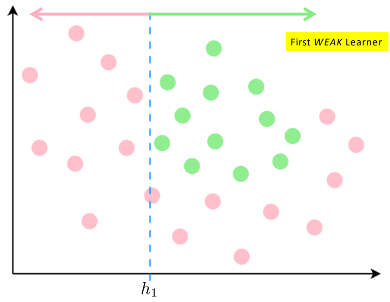
36

AdaBoost



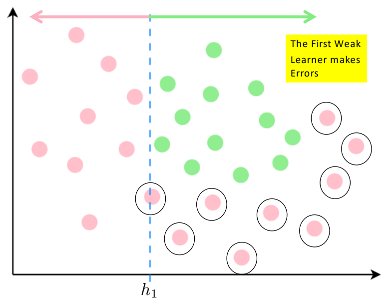
37

AdaBoost

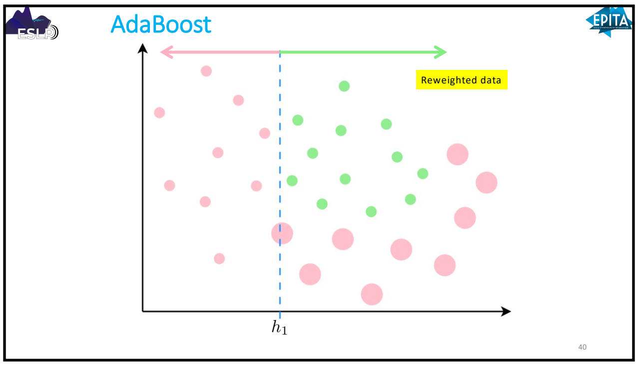


38

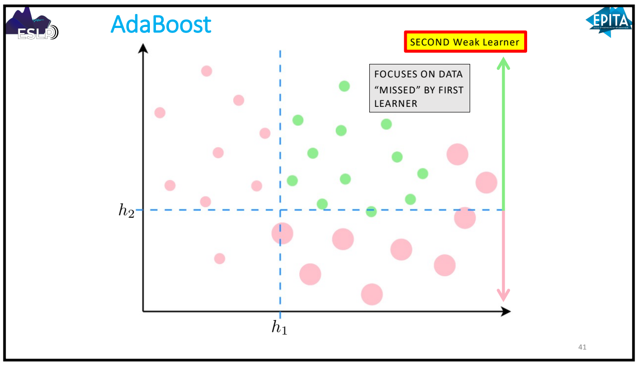
AdaBoost



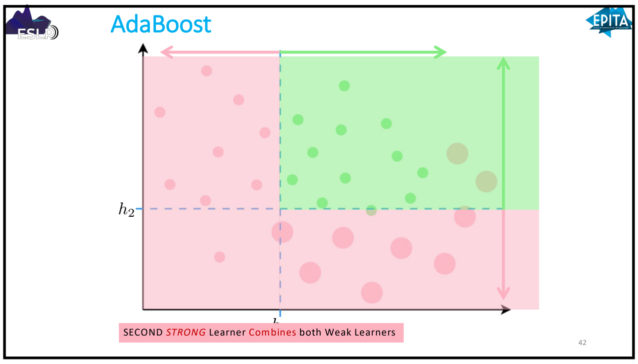
39


AdaBoost
Reweighted data
h_1
40

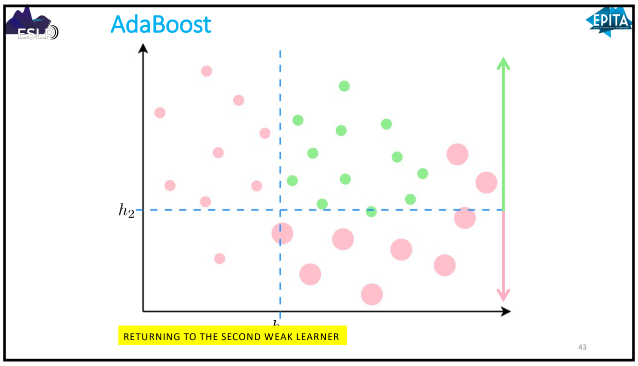
40


AdaBoost
SECOND Weak Learner
FOCUSES ON DATA "MISSED" BY FIRST LEARNER
h_2
h_1
41

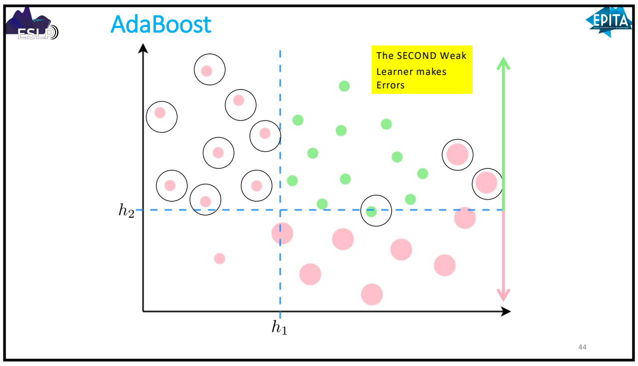
41


AdaBoost
h_2
SECOND STRONG Learner Combines both Weak Learners
42

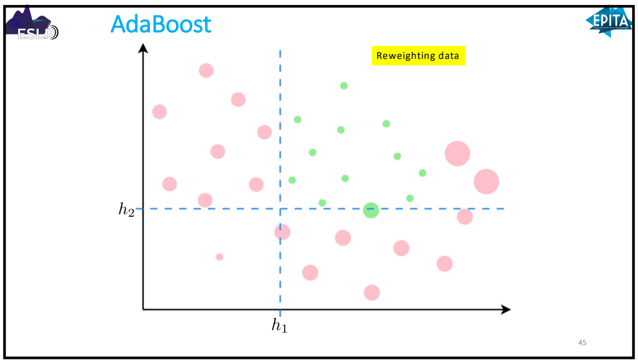
42


AdaBoost
EPITA
h_2
RETURNING TO THE SECOND WEAK LEARNER
43

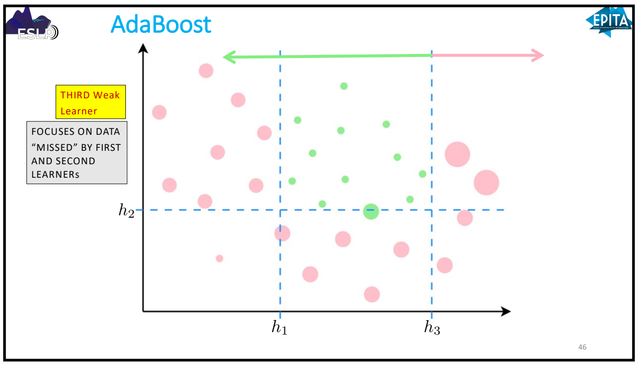
43


AdaBoost
EPITA
The SECOND Weak
Learner makes
Errors
h_2
h_1
44

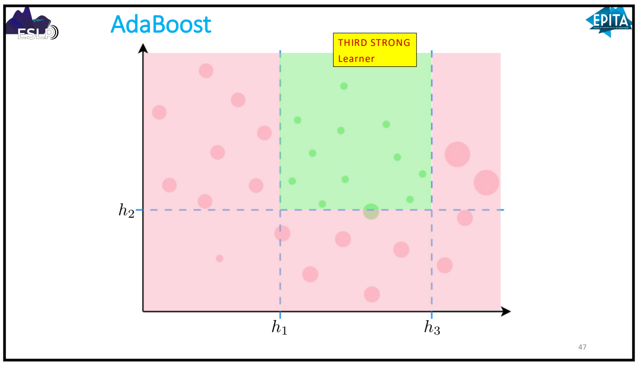
44


AdaBoost
EPITA
Reweighting data
h_2
h_1
45

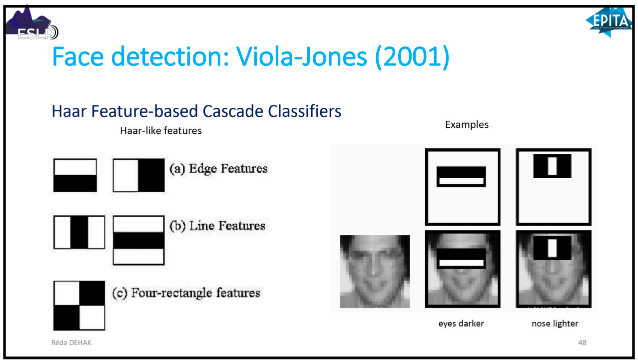
45


AdaBoost
THIRD Weak Learner
FOCUSES ON DATA "MISSED" BY FIRST AND SECOND LEARNERs
h_2
h_1
h_3
46

46

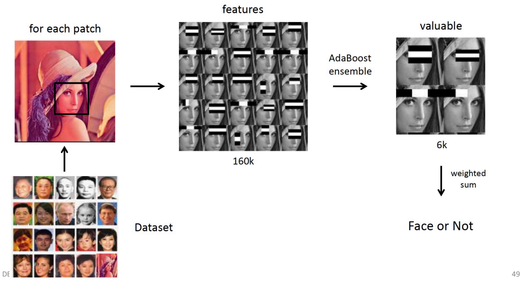

AdaBoost
THIRD STRONG Learner
h_2
h_1
h_3
47

47


Face detection: Viola-Jones (2001)
Haar Feature-based Cascade Classifiers
Haar-like features
(a) Edge Features
(b) Line Features
(c) Four-rectangle features
Examples
eyes darker
nose lighter
Réda DEHAK
48

48

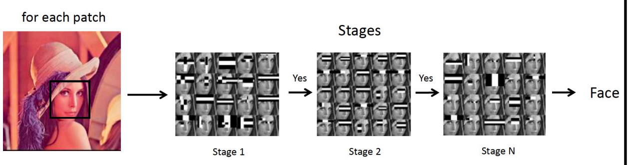
Viola-Jones algorithm: training



49

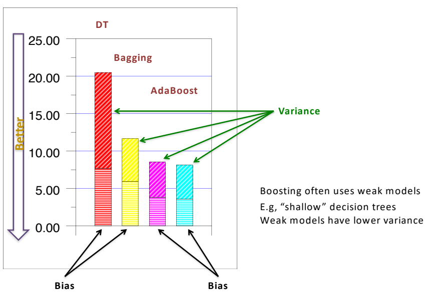
Viola-Jones algorithm: Inference

- Features are grouped into stages
- If a patch fails any stage ⟹ Discard



Réda DEHAK 50

50



Boosting often uses weak models
E.g, "shallow" decision trees
Weak models have lower variance

"An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants"
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

51

Ensemble Selection

S → Training S', Validation V'

H = {2000 models trained using S'}

Maintain ensemble model as combination of H:
$h(x) = h_1(x) + h_2(x) + \dots + h_n(x) + h_{n+1}(x)$

Denote as h_{n+1}

Add model from H that maximizes performance on V'

Repeat

Models are trained on S'
Ensemble built to optimize V'

"Ensemble Selection from Libraries of Models"
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML 2004

52

Conclusions

Method	Minimize Bias?	Minimize Variance?	Other Comments
Bagging	Complex model class. (Deep DTs)	Bootstrap aggregation (resampling training data)	Does not work for simple models.
Random Forests	Complex model class. (Deep DTs)	Bootstrap aggregation + bootstrapping features	Only for decision trees.
Gradient Boosting (AdaBoost)	Optimize training performance.	Simple model class. (Shallow DTs)	Determines which model to add at run-time.
Ensemble Selection	Optimize validation performance	Optimize validation performance	Pre-specified dictionary of models learned on training set.

…and many other ensemble methods as well.

53