Data Mining

Ensemble Techniques

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

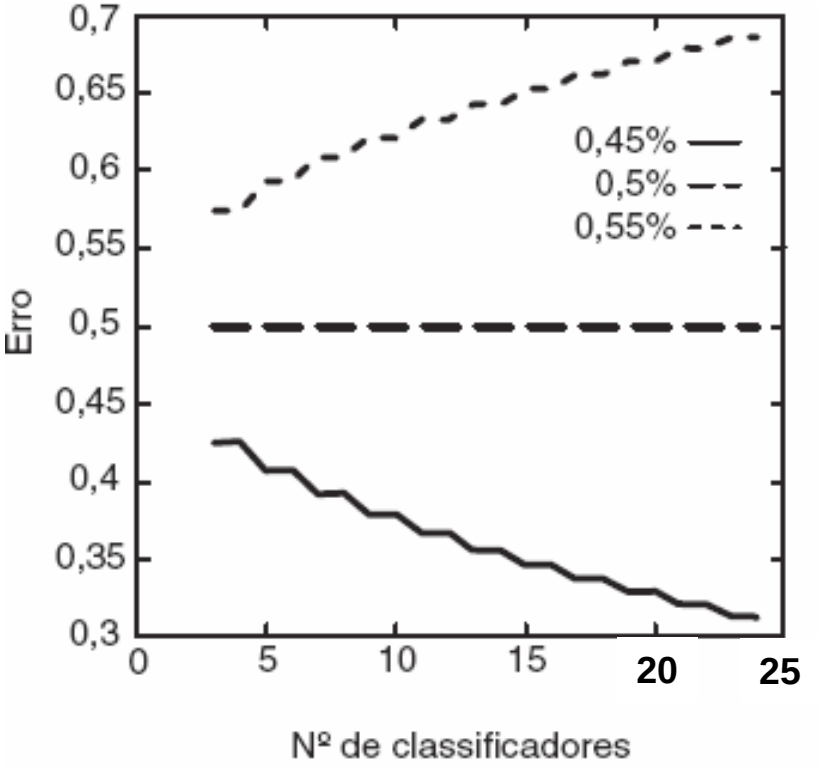
Ensemble Methods

- Techniques used for improving classification accuracy by aggregating the predictions of multiple classifiers
- Construct a set of base classifiers learned from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers (e.g., by taking majority vote)

Example: Why Do Ensemble Methods Work?

- Modelos múltiplos são mais úteis quando os modelos constituintes cometem erros independentes
- Quando...
  - (i) todos os modelos têm a mesma taxa de erro
  - (ii) a taxa de erro é menor que 0,5
  - (iii) todos cometem erros completamente independentes
- ...o erro esperado do conjunto decresce linearmente com o número de modelos

Example: Why Do Ensemble Methods Work?



Evolução da taxa de erro variando o número de classificadores em um modelo múltiplo

Esse é um estudo de simulação, em um problema de duas classes equiprováveis, ou seja, a probabilidade de observar cada classe é 50%

Todos os classificadores têm a mesma probabilidade de cometer um erro, mas os erros são independentes uns dos outros

Quando a probabilidade é 45%, a taxa de erro do conjunto decresce linearmente

Quando a probabilidade é 55%, a taxa de erro cresce linearmente

Quando a probabilidade é igual a 50%, a taxa de erro do conjunto permanece constante

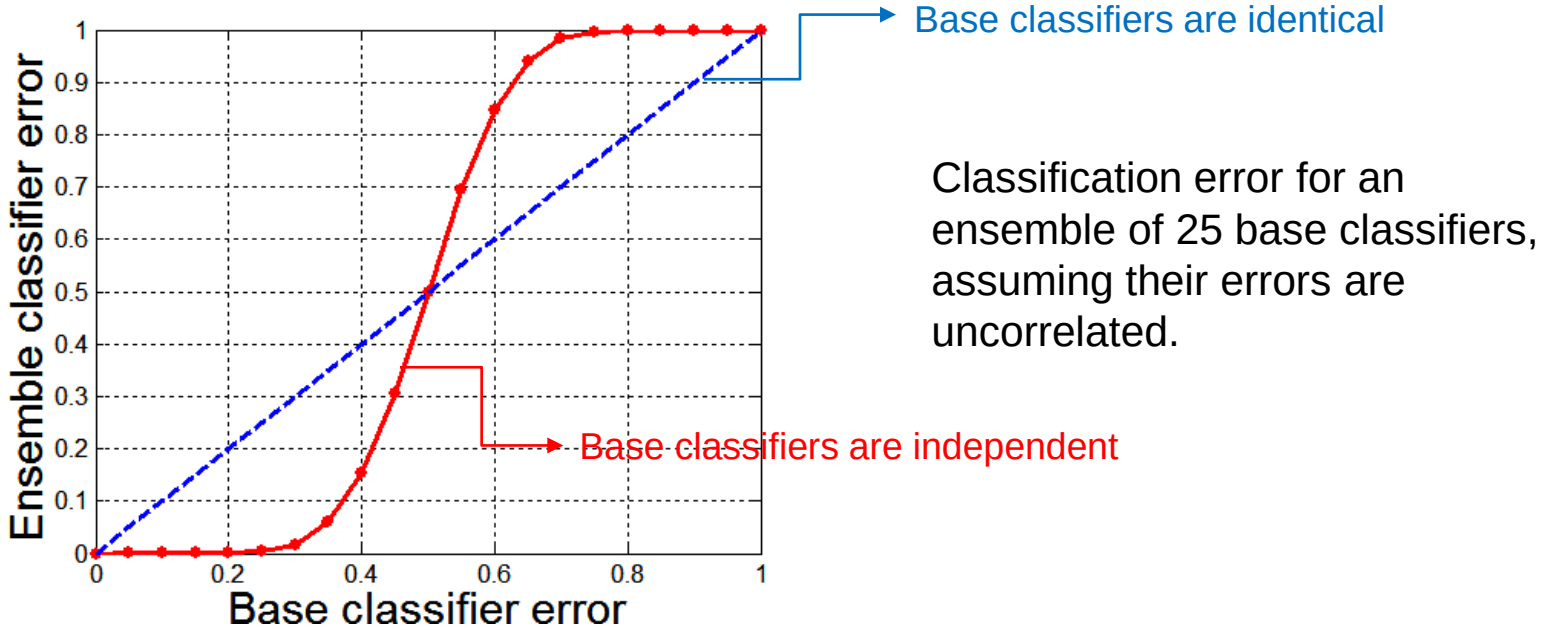
Example: Why Do Ensemble Methods Work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\epsilon = 0.35$
  - Majority vote of classifiers used for classification
  - If all classifiers are identical:
    - Error rate of ensemble = ϵ (0.35)
  - If all classifiers are independent (errors are uncorrelated): then the ensemble makes a wrong prediction only if more than half of the base classifiers predict incorrectly
    - Error rate of ensemble = probability of having more than half of base classifiers being wrong

$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1-\epsilon)^{25-i} = 0.06$$

Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
  1. All base classifiers are independent of each other
  2. All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)



Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.

Bias-Variance Decomposition

- Bias-variance decomposition is a formal method for analyzing the generalization error of a predictive model

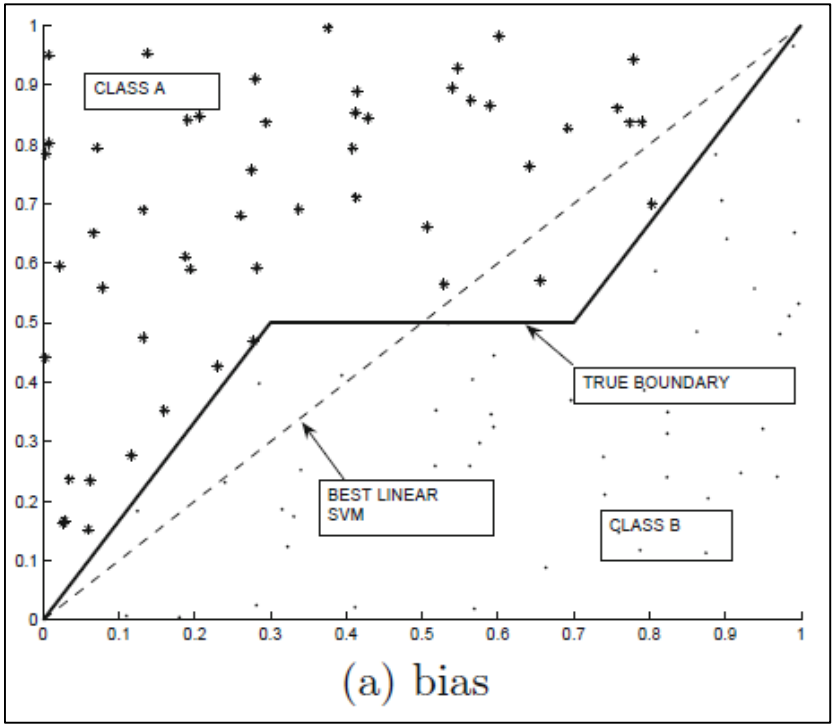
Bias-Variance Decomposition

- Prediction errors can be decomposed into two main subcomponents:
  - error due to "bias“
  - error due to "variance“
- There is a tradeoff between a model's ability to minimize bias and variance

https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Decomposition

- **Bias**: every classifier makes its own modeling assumptions about the nature of the decision boundary between classes

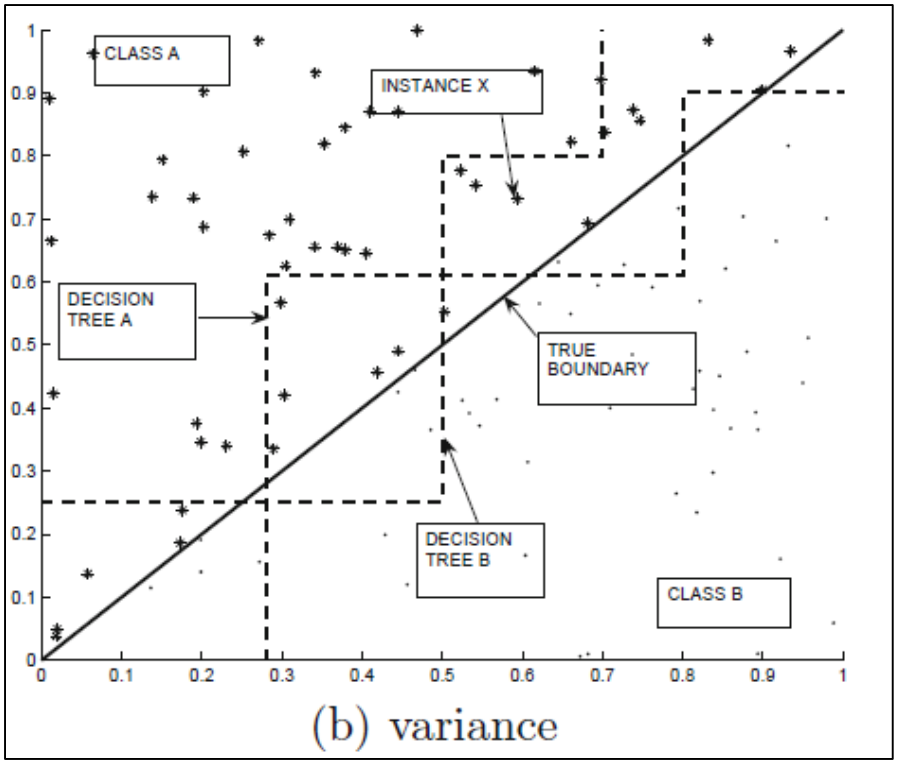


- No (linear) SVM classifier can classify all the possible test instances correctly even if the best possible SVM model is constructed with a very large training data set

- When a classifier has **high bias**, it will **make *consistently incorrect* predictions** over particular choices of test instances near the incorrectly modeled decision-boundary, even when different samples of the training data are used for the learning process

Bias-Variance Decomposition

- **Variance**: random variations in the choices of the training data will lead to different models



- Different choices of training data might lead to different split choices, as a result of which the decision boundaries of trees A and B are very different
- Therefore, (test) instances such as X are *inconsistently classified* by decision trees which were created by different choices of training data sets - this is a manifestation of model *variance*
- Model **variance** is closely **related** to **overfitting**. When a classifier has an overfitting tendency, it will make ***inconsistent* predictions** for the **same test instance** over **different training** data **sets**

Bias-Variance Decomposition

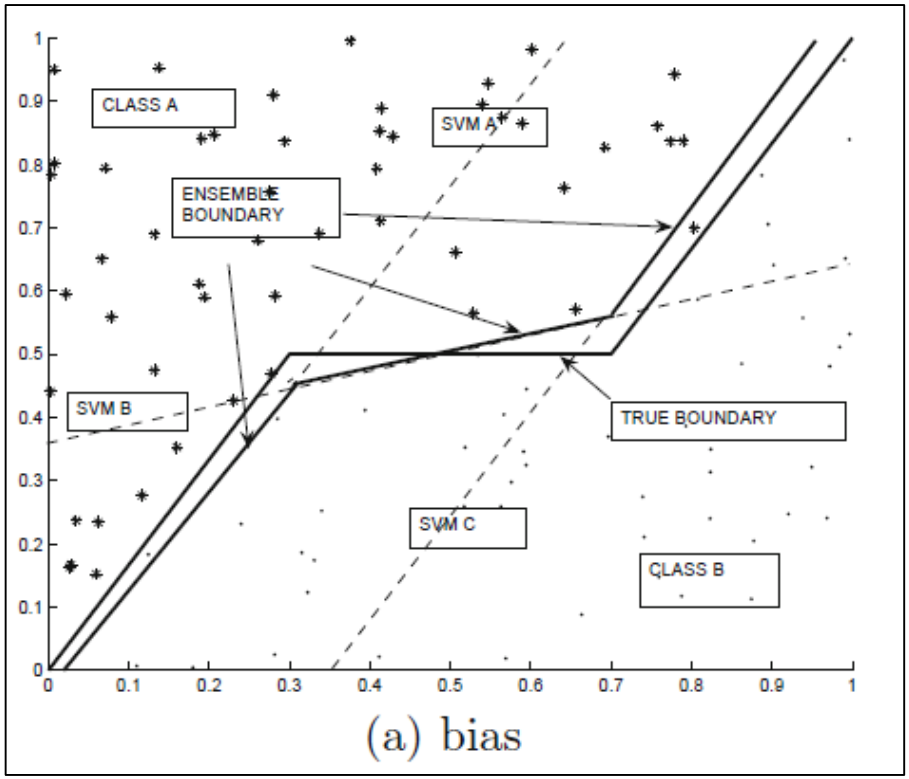
- The design choices of a classifier often reflect a trade-off between the bias and the variance
  - Pruning a decision tree results in a more stable classifier and therefore reduces the variance
  - On the other hand, because the pruned decision tree makes stronger assumptions about the simplicity of the decision boundary than the unpruned tree, the former leads to greater bias

Bias-Variance Decomposition

- Simplified assumptions about the decision boundary lead to greater bias but lower variance
- On the other hand, complex assumptions reduce bias but are harder to robustly estimate with limited data
- The bias and variance are affected by virtually every design choice of the model, such as the choice of the base algorithm or the choice of model parameters

Bias-Variance Decomposition

- Ensemble analysis can often be used to reduce both the bias and variance of the classification process

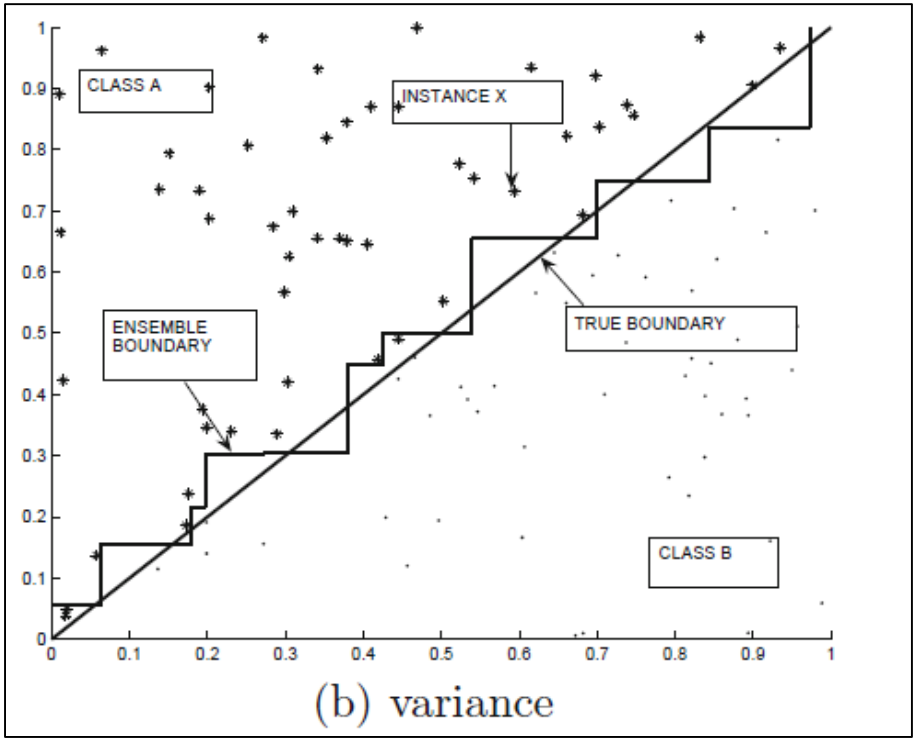


The decision boundary of this ensemble is not linear and has lower bias with respect to the true decision boundary (boosting)

The reason for this is that different classifiers have different levels and directions of bias in different parts of the training data, and the majority vote across the different classifiers is able to obtain results that are generally less biased in any specific region than each of the component classifiers

Bias-Variance Decomposition

- Ensemble analysis can often be used to reduce both the bias and variance of the classification process



By using the aggregation over sufficiently independent classifiers, it becomes increasingly likely that instances close to the decision boundary, such as X, will be correctly classified

Bias-Variance Decomposition

Different classification models have different sources of bias and variance

Technique	Source/level of bias	Source/level of variance
Simple models	Oversimplification increases bias in decision boundary	Low variance. Simple models do not overfit
Complex models	Generally lower than simple models. Complex boundary can be modeled	High variance. Complex assumptions will be overly sensitive to data variation
Shallow decision trees	High bias. Shallow tree will ignore many relevant split predicates	Low variance. The top split levels do not depend on minor data variations
Deep decision trees	Lower bias than shallow decision tree. Deep levels model complex boundary	High variance because of overfitting at lower levels
Rules	Bias increases with fewer antecedents per rule	Variance increases with more antecedents per rule
Naive Bayes	High bias from simplified model (e.g., Bernoulli) and naive assumption	Variance in estimation of model parameters. More parameters increase variance
Linear models	High bias. Correct boundary may not be linear	Low variance. Linear separator can be modeled robustly
Kernel SVM	Bias lower than linear SVM. Choice of kernel function	Variance higher than linear SVM
k-NN model	Simplified distance function such as Euclidean causes bias. Increases with k	Complex distance function such as local discriminant causes variance. Decreases with k
Regularization	Increases bias	Reduces variance

Bias-Variance Decomposition

- **Error due to Bias**: difference between the expected (or average) prediction of our model and the correct value which we are trying to predict
  - Remember that although we have only one model we repeat the whole model building process more than once
  - Due to randomness in the underlying data sets, the resulting models will have a range of predictions
  - Bias measures how far off in general these models' predictions are from the correct value

https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Decomposition

- **Error due to Variance**: variability of a model prediction for a given data point
  - Again, we can repeat the entire model building process multiple times
  - The variance is how much the predictions for a given point vary between different realizations of the model

https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Decomposition

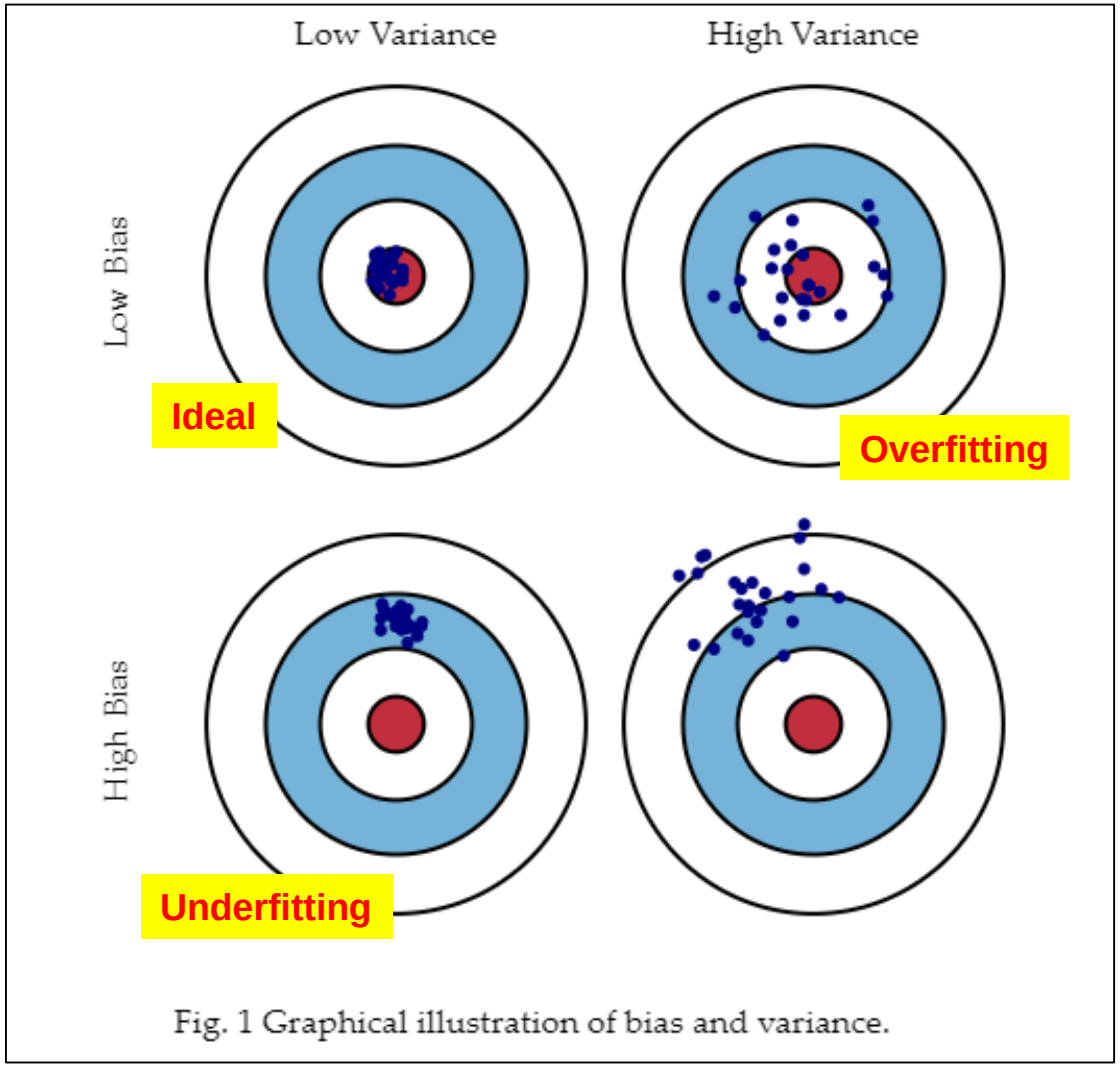
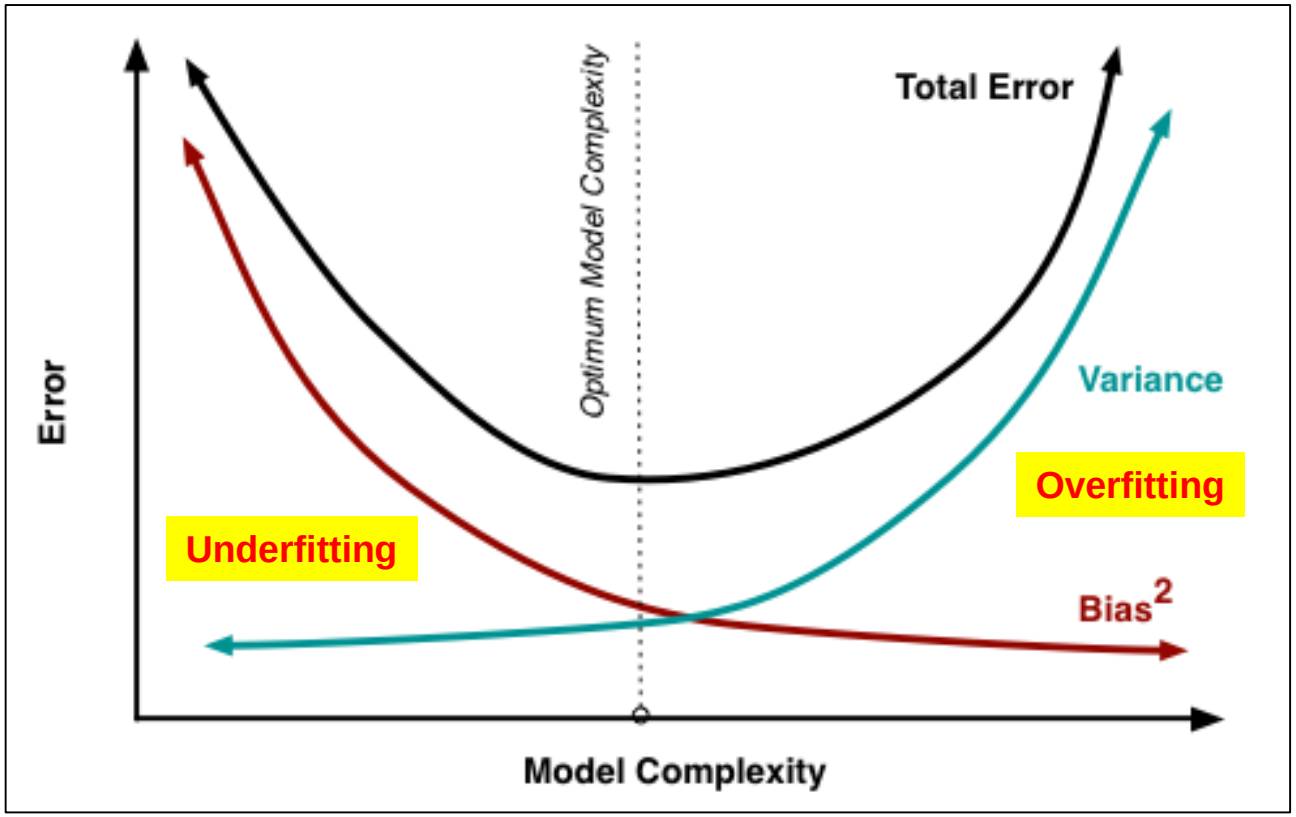


Fig. 1 Graphical illustration of bias and variance.

https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias-Variance Trade-off



https://scott.fortmann-roe.com/docs/BiasVariance.html

As more and more parameters are added to a model, the complexity of the model rises and variance becomes our primary concern while bias steadily falls

Understanding bias and variance is critical for understanding the behavior of prediction models, but in general what you really care about is overall error

Combinando Ensembles

- Quando combinamos as predições dos classificadores, podemos diferenciar
  - métodos de votação versus métodos de seriação
  - métodos dinâmicos versus métodos estáticos

Combinando Ensembles

- Métodos de Votação versus Métodos de Seriação

- Métodos de Votação (mais comumente usado)
  - Os classificadores de base produzem um rótulo de classe
    - Votação uniforme = todos os classificadores de base contribuem igualmente para a classificação final
    - Votação com peso = cada classificador de base tem um peso associado

Combinando Ensembles

- Métodos de Votação versus Métodos de Seriação

- Métodos de Seriação
  - A saída dos classificadores de base é probabilística, isto é, associam, para cada exemplo teste, uma probabilidade para cada possível classe

m = nro. de modelos

- Regra da soma: $S_k = \sum_{i=1}^{m} P_{ik}$
- Regra da média: $S_k = \sum_{i=1}^{m} P_{ik} / m$
- Regra da média geométrica: $S_j = \sqrt[m]{\prod_{i=1}^{m} P_{ik}}$
- Regra do produto: $S_k = \prod_{i=1}^{m} P_{ik}$
- Regra do máximo: $S_k = \max_i P_{ik}$
- Regra do mínimo: $S_k = \min_i P_{ik}$

Combinando Ensembles

- Métodos Dinâmicos versus Métodos Estáticos

- A distribuição da taxa de erros sobre o espaço de atributos, geralmente, não é homogênea
- Dependendo do classificador, a taxa de erro será mais concentrada em certas regiões do espaço de objetos do que em outras
- Métodos Estáticos = consideram as previsões de todos os elementos do conjunto
- Métodos Dinâmicos = consideram o exemplo de teste e realizam uma seleção do modelo para classificá-lo

Combinando Ensembles

- Método Dinâmico: MAI (Model Applicability Induction)
- Caracteriza as situações em que cada modelo é capaz de fazer previsões corretas
  - Feito a partir do aprendizado de um metaclassificador para cada modelo disponível da base
  - O objetivo desse metaclassificador é predizer onde o modelo de base classificará corretamente o exemplo de teste

Combinando Ensembles

MAI (Model Applicability Induction)

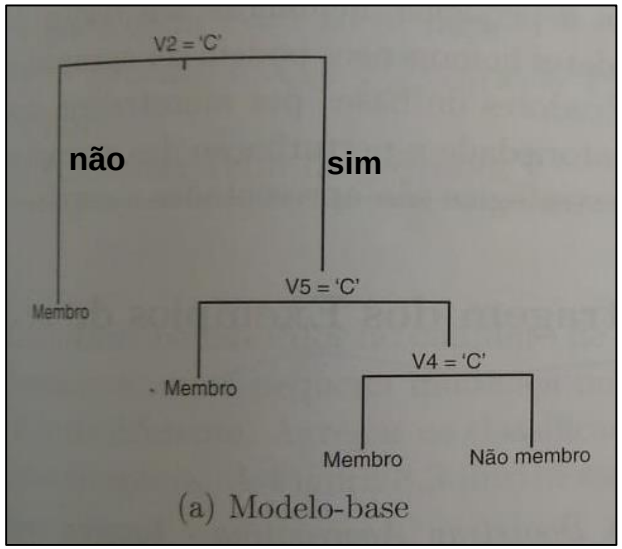
Define as regiões do espaço onde os classificadores de base são mais (ou menos) propensos a erro

V1	V2	V3	V4	V5	Classe
t	a	c	t	a	membro
t	g	c	t	a	membro
g	t	a	c	t	não membro
a	a	t	t	g	membro
t	c	g	a	t	não membro
a	g	g	g	g	membro

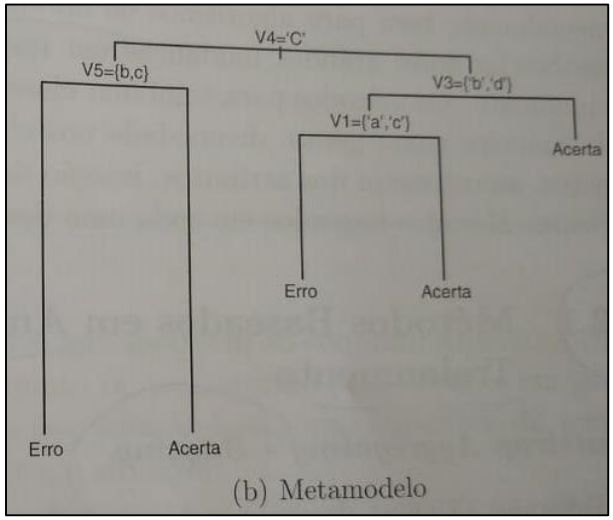
Conjunto de dados original

V1	V2	V3	V4	V5	Erro
t	a	c	t	a	+
t	g	c	t	a	−
g	t	a	c	t	+
a	a	t	t	g	+
t	c	g	a	t	−
a	g	g	g	g	+

Conjunto de dados $Nível_1$



(a) Modelo-base



(b) Metamodelo

Os exemplos positivos são os exemplos corretamente classificados pelo algoritmo de aprendizado de base, e os exemplos negativos são os incorretamente classificados

Os dados de Nível1 representam sempre um problema de duas classes

Para objetos novos, os metaclassificadores são primeiramente consultados para selecionar o modelo de predição mais apropriado, e a predição do modelo selecionado é então devolvida

Combinando Ensembles

- **Combinando Classificadores Homogêneos**
  - Métodos Baseados em Amostragem dos Exemplos de Treinamento (Bagging, Boosting)
  - Métodos Baseados na Injeção de Aleatoriedade (Random Forests)
  - etc.
- Combinando Classificadores Heterogêneos
  - Generalização em Pilha (Stacking)
  - Generalização em Cascata
  - etc.

Combinando Classificadores Homogêneos

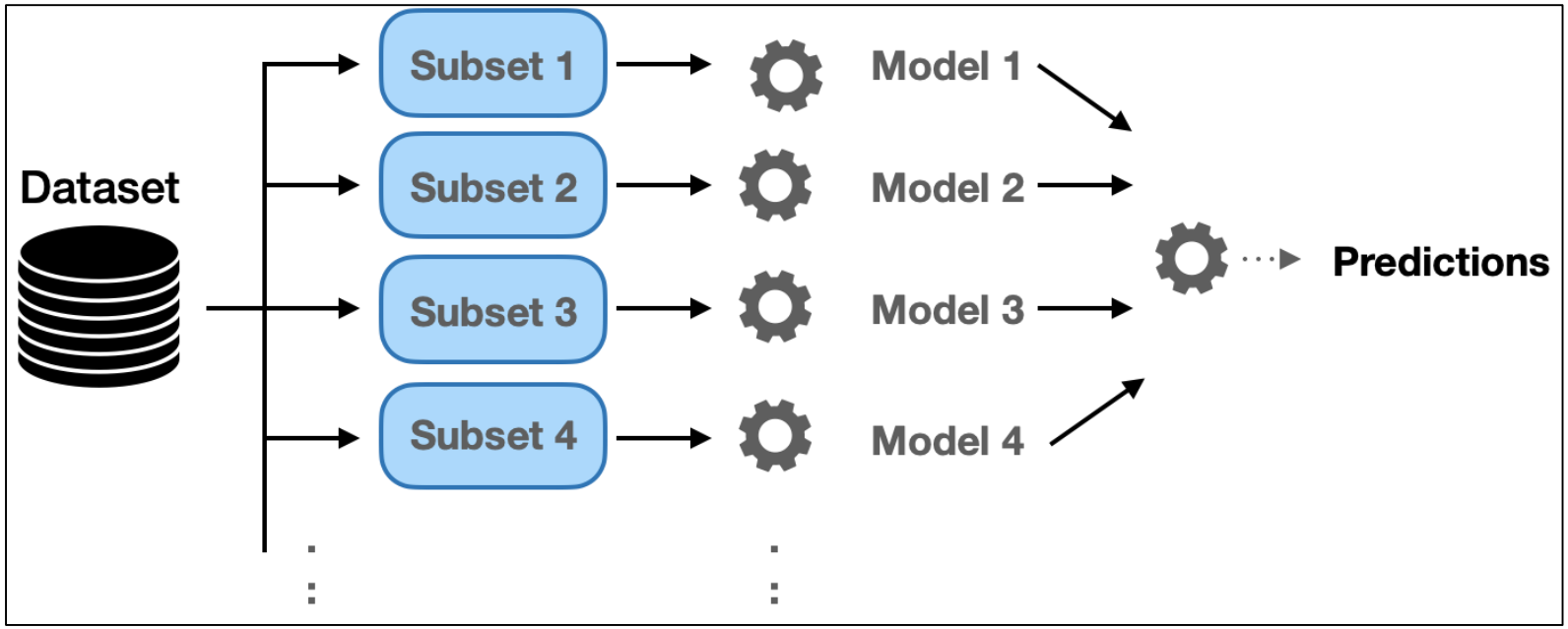
- Métodos que combinam modelos gerados por um único algoritmo
- Diversidade é um dos requisitos quando são usados modelos múltiplos

Combinando Classificadores Homogêneos

- Várias estratégias foram propostas para geração de classificadores diferentes usando o mesmo algoritmo de aprendizado
  - A maioria manipula o conjunto de treinamento para gerar múltiplas hipóteses
  - O algoritmo de aprendizado é executado várias vezes, utilizando cada vez uma distribuição diferente de exemplos de treinamento

Bagging (Bootstrap AGGregatING)

Bagging technique to make final predictions by combining predictions from multiple models



https://towardsdatascience.com/ensemble-models-5a62d4f4cb0c

Para algoritmos com ↑variância ↓bias => ↓variância

Bagging (Bootstrap AGGregatING)

- Bagging is a technique that repeatedly samples (with replacement) from a data set
- Each bootstrap sample has the same size as the original data
- Because the sampling is done with replacement, some instances may appear several times in the same training set, while others may be omitted from the training set

Bagging (Bootstrap AGGregatING)

- Bootstrap sampling: sampling with replacement

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

- Build classifier on each bootstrap sample
- Probability of a training instance being selected in a bootstrap sample is:
  - $1 - (1 - 1/n)^n$ (n: number of training instances)
  - ~0.632 (63,2%) when n is large (36,8% duplicatas)

Bagging Algorithm

Algorithm 4.5 Bagging algorithm.

1: Let k be the number of bootstrap samples.
2: **for** $i = 1$ to k **do**
3: Create a bootstrap sample of size N, D_i.
4: Train a base classifier C_i on the bootstrap sample D_i.
5: **end for**
6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta\big(C_i(x) = y\big)$.

$\{\delta(\cdot) = 1$ if its argument is true and 0 otherwise.$\}$

A test instance is assigned to the class that receives the highest number of votes

Bagging Example

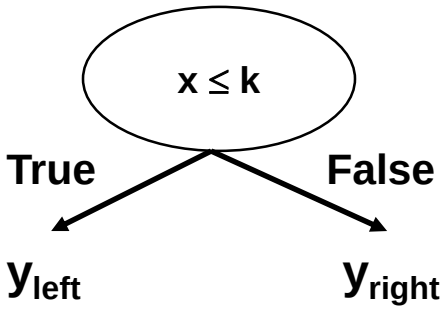
https://en.wikipedia.org/wiki/Decision_stump

- Consider 1-dimensional data set:

Original Data:

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

- Classifier is a decision stump (decision tree of size 1)
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



Bagging Example

Bagging 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

x <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

x <= 0.7 ➔ y = 1
x > 0.7 ➔ y = 1

Bagging Round 3:

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 <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Round 4:

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 <= 0.3 ➔ y = 1
x > 0.3 ➔ y = -1

Bagging Round 5:

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 <= 0.35 ➔ y = 1
x > 0.35 ➔ y = -1

Bagging Example

Bagging Round 6:

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 <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 7:

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 <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 8:

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 <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 9:

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 <= 0.75 ➔ y = -1
x > 0.75 ➔ y = 1

Bagging Round 10:

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

x <= 0.05 ➔ y = 1
x > 0.05 ➔ y = 1

Bagging Example

- Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Use majority vote (sign of sum of predictions) to determine class of ensemble classifier

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	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

- Ensemble classifier perfectly classifies all 10 examples in the original data

Bagging Example

- Bagging can also increase the complexity (representation capacity) of simple classifiers such as decision stumps
  - Even though each base classifier is a decision stump, combining the classifiers can lead to a decision boundary that mimics a decision tree of depth 2

Bagging

Para algoritmos com ↑variância ↓bias => ↓variância

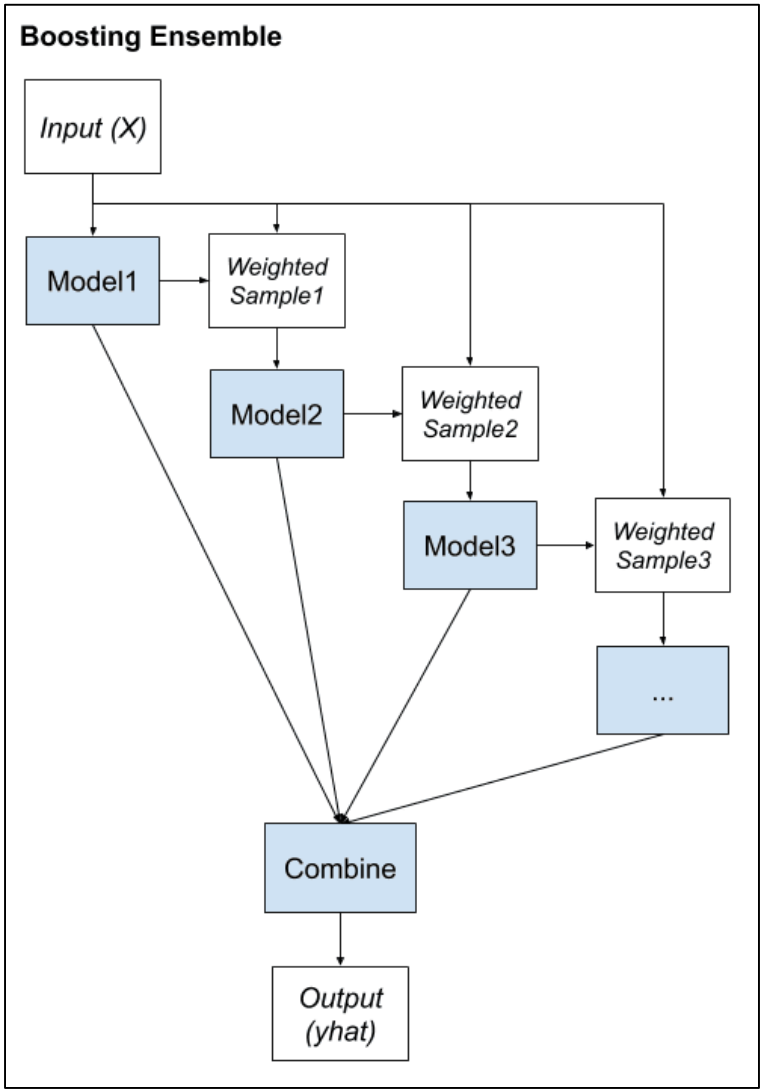
- **Bagging improves generalization error by reducing the variance of the base classifiers**
  - Essa técnica funciona bem para **algoritmos de aprendizado instáveis** (**unstable base classifiers**)
    - Classifiers that are sensitive to minor perturbations in training set, due to **high model complexity**
    - Examples: Unpruned decision trees, ANNs, …

Bagging

- The performance of bagging depends on the stability of the base classifier
  - If a base classifier is unstable, bagging helps to reduce the errors associated with random fluctuations in the training data
  - If a base classifier is stable, i.e., robust to minor perturbations in the training set, then the error of the ensemble is primarily caused by bias in the base classifier
    - In this situation, bagging may not be able to improve the performance of the base classifiers significantly

Boosting

Para algoritmos com ↑bias ↓variância => ↓bias



https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

Boosting

- Poderá um conjunto de modelos de aprendizado fracos gerar um modelo forte?
  - Um classificador fraco é definido como um classificador cuja capacidade de generalização é pouco melhor que a escolha aleatória
  - Por outro lado, um classificador forte pode aproximar qualquer distribuição com um erro arbitrariamente pequeno

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  - Initially, all N records are assigned equal weights (for being selected for training)
  - Unlike bagging, weights may change at the end of each boosting round

Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

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

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

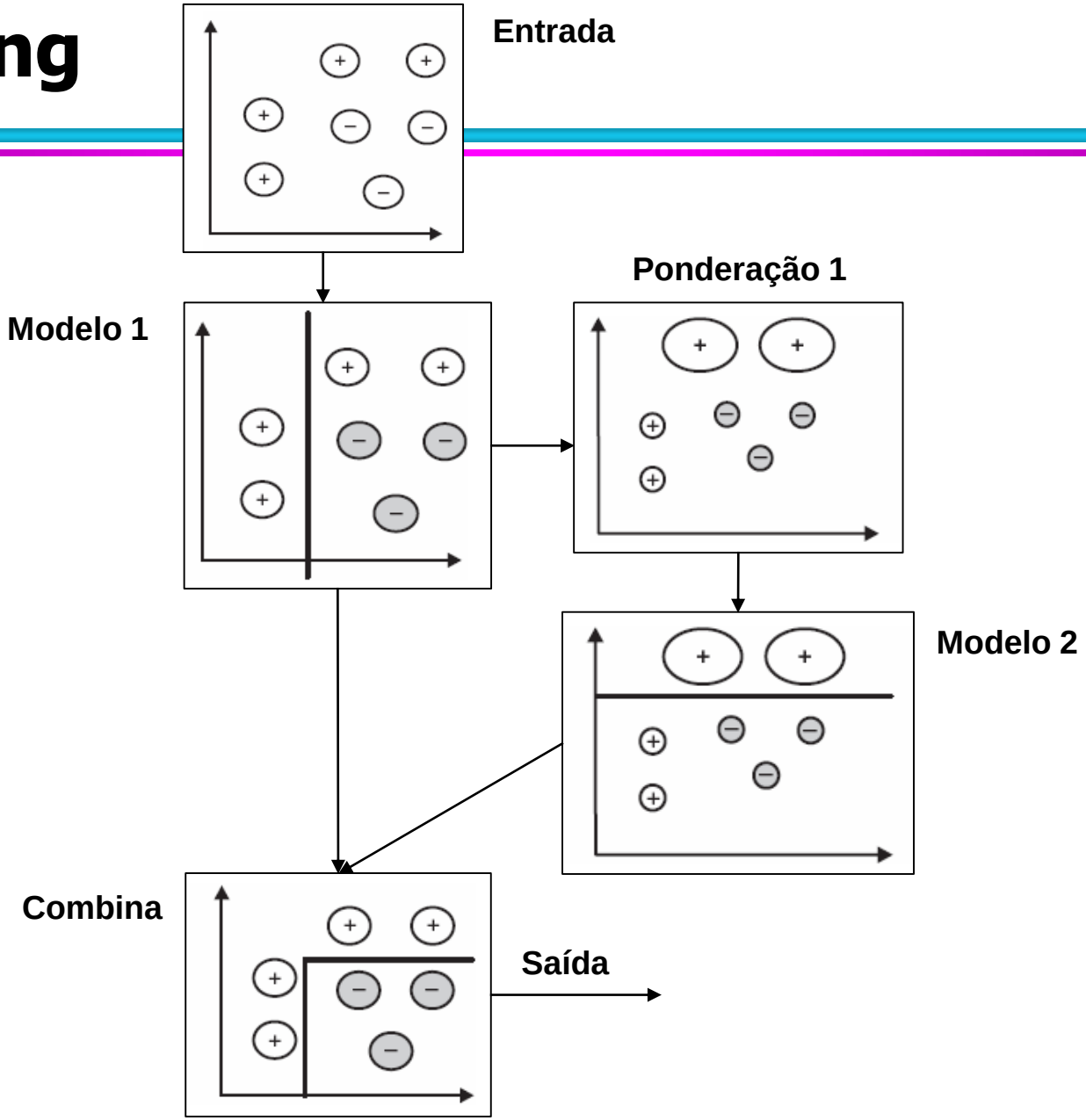
Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Examples that were not chosen in the previous round, e.g., examples 1 and 5, also have a better chance of being selected in the next round since their predictions in the previous round were likely to be wrong

Boosting

- The final ensemble is obtained by aggregating the base classifiers obtained from each boosting round
- Several implementations of the boosting algorithm have been developed, differing in terms of
  - (1) how the weights of the training examples are updated at the end of each boosting round
  - (2) how the predictions made by each classifier are combined

Boosting



Boosting

Para algoritmos com ↑bias ↓variância => ↓bias

- Em geral, trabalham com modelos fracos (não complexos), como shallow decision trees
- Funciona pq.:
  - By focusing on examples that are difficult to classify by base classifiers, it is able to **reduce the bias** of the final predictions
    - Os exemplos observados tendem a ter níveis diversos de dificuldade de classificação. Exemplos perto da superfície de decisão, por exemplo, são mais difíceis de classificar do que os exemplos mais afastados
  - However, because of its tendency to focus on training examples that are wrongly classified, the boosting technique can be susceptible to overfitting

AdaBoost Algorithm (Boosting)

Algorithm 4.6 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: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta\big(C_i(x_j) \neq y_j\big)\right]$ {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 4.103.
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y)\big)$.

Importance of the classifier (depends on its error rate)

AdaBoost Algorithm

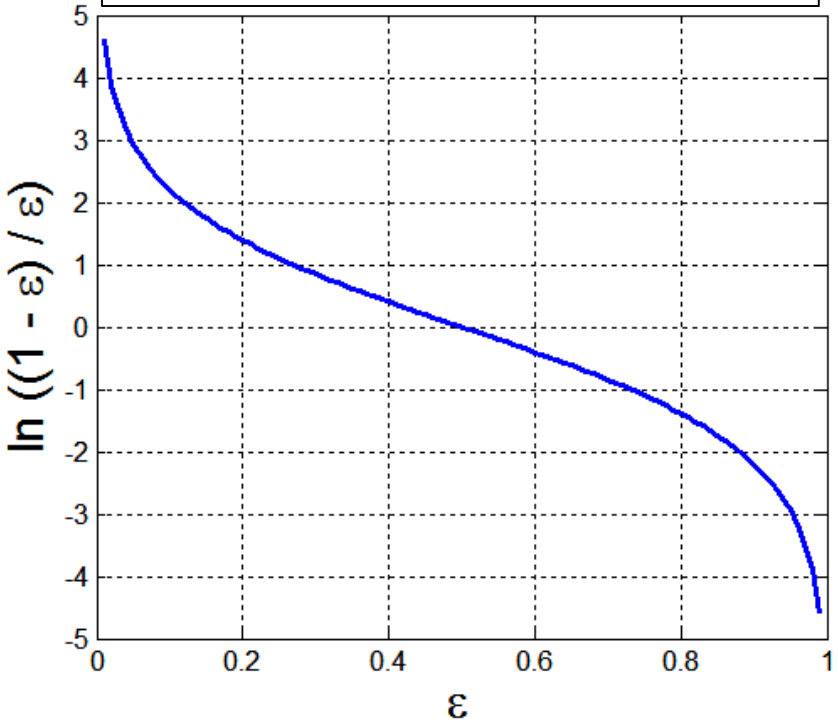
- Base classifiers: C_1, C_2, …, C_T
- Error rate of a base classifier:

$$\epsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j^{(i)}\, \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

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

α has a large positive value if the error rate is close to 0 and a large negative value if the error rate is close to 1



AdaBoost Algorithm

- Weight update:

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

Where Z_i is the normalization factor **(Soma dos pesos = 1)**

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg\max_y \sum_{i=1}^{T} \alpha_i \delta(C_i(x) = y)$$

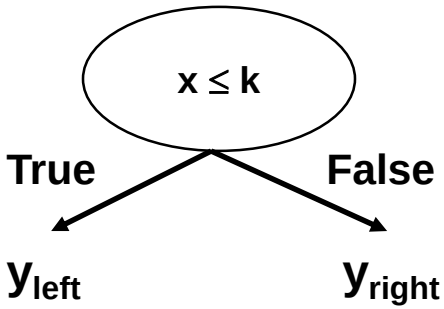
AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

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

- Classifier is a decision stump
  - Decision rule: $x \leq k$ versus $x > k$
  - Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting 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

Boosting 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

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

AdaBoost Example

- Weights

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	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

- Classification

Ensemble classifier perfectly classifies all 10 examples in the original data

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
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted Class

Gradient Boosting (GB, GBM)

- Different from Adaboost, the principle idea behind here is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble

Gradient boosting machines, a tutorial. 2013. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/

- The idea is to build models sequentially so that these subsequent models try to reduce the errors of the previous model (this is done by building a new model on the errors or residuals of the previous model)

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

$$F(m) = F(m-1) + \eta * -\frac{\partial L}{\partial F(m-1)}$$

https://towardsdatascience.com/a-visual-guide-to-gradient-boosted-trees-8d9ed578b33

Gradient Boosting

- One can arbitrarily specify both the loss function and the base-learner models on demand
  - However, in general, trees are used (GBDT)

Algorithm 1 Friedman's Gradient Boost algorithm

Inputs:

- input data $(x, y)_{i=1}^{N}$
- number of iterations M
- choice of the loss-function $\Psi(y, f)$
- choice of the base-learner model $h(x, \theta)$

Algorithm:

1: initialize $\widehat{f}_0$ with a constant
2: **for** $t = 1$ to M **do**
3: compute the negative gradient $g_t(x)$
4: fit a new base-learner function $h(x, \theta_t)$
5: find the best gradient descent step-size ρ_t:

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{N} \Psi\left[y_i, \widehat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)\right]$$

least square minimization

6: update the function estimate:

$$\widehat{f}_t \leftarrow \widehat{f}_{t-1} + \rho_t h(x, \theta_t)$$

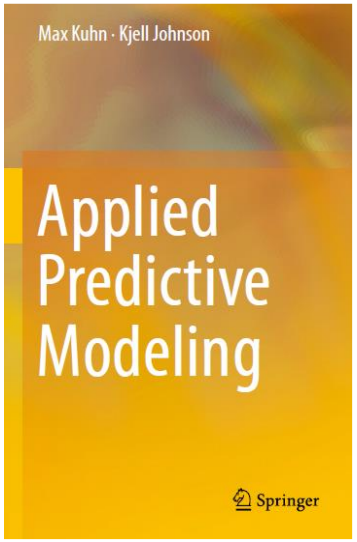
7: **end for**

Gradient boosting machines, a tutorial. 2013.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/

Gradient Boosting

- Trees make an excellent base learner for boosting for several reasons:
  - They have the flexibility to be weak learners by simply restricting their depth
  - Separate trees can be easily added together
  - Trees can be generated very quickly
    - Thus, results from individual trees can be directly aggregated, thus making them inherently suitable for an additive modeling process



Gradient Boosting (GBDT)

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Row No.	Cylinder Number	Car Height	Engine Location	Price
1	Four	48.8	Front	12000
2	Six	48.8	Back	16500
3	Five	52.4	Back	15500
4	Four	54.3	Front	14000

$$\Psi(y, f)_{L_2} = \frac{1}{2}(y - f)^2$$

squared-error L_2 loss

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1
1	Four	48.8	Front	12000	14500
2	Six	48.8	Back	16500	14500
3	Five	52.4	Back	15500	14500
4	Four	54.3	Front	14000	14500

F_0

Gradient Boosting (GBDT)

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1
1	Four	48.8	Front	12000	14500	-2500
2	Six	48.8	Back	16500	14500	2000
3	Five	52.4	Back	15500	14500	1000
4	Four	54.3	Front	14000	14500	-500

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

In the case of the L_2 loss-function, its derivative is the residual $-(y - f)$, which implies that the GBM algorithm simply performs residual refitting

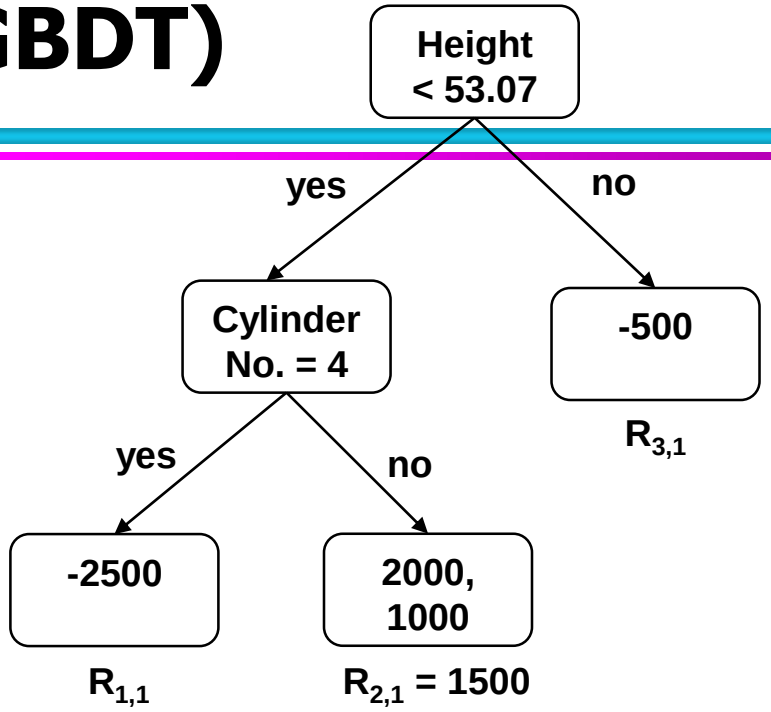
$$\Psi(y, f)_{L_2} = \frac{1}{2}(y - f)^2$$

squared-error L_2 loss

$$(Observed - Predicted)$$

Gradient Boosting (GBDT)

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/



$\eta = [0, 1]$

$$F(m) = F(m-1) + \eta * -\frac{\partial L}{\partial F(m-1)}$$

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1	Prediciton 2
1	Four	48.8	Front	12000	14500	-2500	14500 - 0.1 * 2500 = 14250
2	Six	48.8	Back	16500	14500	2000	14500 + 0.1 * 1500 = 14650
3	Five	52.4	Back	15500	14500	1000	14500 + 0.1 * 1500 = 14650
4	Four	54.3	Front	14000	14500	-500	14500 - 0.1 * 500 = 14450

Gradient Boosting (GBDT)

https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/

F_1

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1	Prediciton 2
1	Four	48.8	Front	12000	14500	-2500	14500 - 0.1 * 2500 = 14250
2	Six	48.8	Back	16500	14500	2000	14500 + 0.1 * 1500 = 14650
3	Five	52.4	Back	15500	14500	1000	14500 + 0.1 * 1500 = 14650
4	Four	54.3	Front	14000	14500	-500	14500 - 0.1 * 500 = 14450

e assim sucessivamente...

Gradient Boosting (GBDT)

- Existem diversas implementações/variações
  - XGBoost, Light GBM, CatBoost
    - Comparação: https://medium.com/octave-john-keells-group/xgboost-light-gbm-and-catboost-a-comparison-of-decision-tree-algorithms-and-applications-to-a-f1d2d376d89c
  - Scikit-Learn API
  - Unlike RFs, GBMs can have high variability in accuracy dependent on their hyperparameter settings [https://bradleyboehmke.github.io/HOML/gbm.html] [https://machinelearningmastery.com/gradient-boosting-machine-ensemble-in-python/]
- SGBM (Stochastic GBM): (i) subsample rows before creating each tree; (ii) subsample columns before creating each tree; (iii) subsample columns before considering each split

Random Forest Algorithm

- Alguns algoritmos de aprendizado usam parâmetros inicializados aleatoriamente
- Essa característica pode ser explorada no sentido de gerar diferentes modelos pela injeção de aleatoriedade nas entradas ou parâmetros do algoritmo de aprendizado

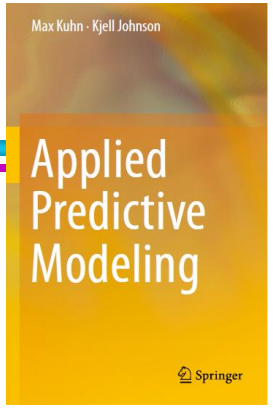
Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating training set as well as features, using the following steps:
  - (1) Use bootstrap sample to train every decision tree (similar to Bagging)
  - (2) Use the following tree induction algorithm:
    - At every internal node of decision tree, randomly sample p attributes for selecting split criterion
    - Repeat this procedure until all leaves are pure (unpruned tree)

Characteristics of Random Forest

- Base classifiers are unpruned trees and hence are *unstable classifiers* **(low bias, high variance)**
- Base classifiers are *decorrelated* (due to randomization in training set as well as features)
- Random forests reduce variance of unstable classifiers without negatively impacting the bias **(robust to overfitting)**
- Selection of hyper-parameter p
  - Small value ensures lack of correlation
  - High value promotes strong base classifiers
  - Common default choices: $\sqrt{d}$, $\log_2(d+1)$

RF X GBDT



- In random forests, all trees are created independently, each tree is created to have maximum depth, and each tree contributes equally to the final model
- The trees in boosting, however, are dependent on past trees, have minimum depth, and contribute unequally to the final model
- Computation time for boosting is often greater than for random forests, since random forests can be easily parallel processed given that the trees are created independently

Combinando Ensembles

- Combinando Classificadores Homogêneos
  - Métodos Baseados em Amostragem dos Exemplos de Treinamento (Bagging, Boosting)
  - Métodos Baseados na Injeção de Aleatoriedade (Random Forests)
  - etc.
- **Combinando Classificadores Heterogêneos**
  - Generalização em Pilha (Stacking)
  - Generalização em Cascata
  - etc.

Combinando Classificadores Heterogêneos

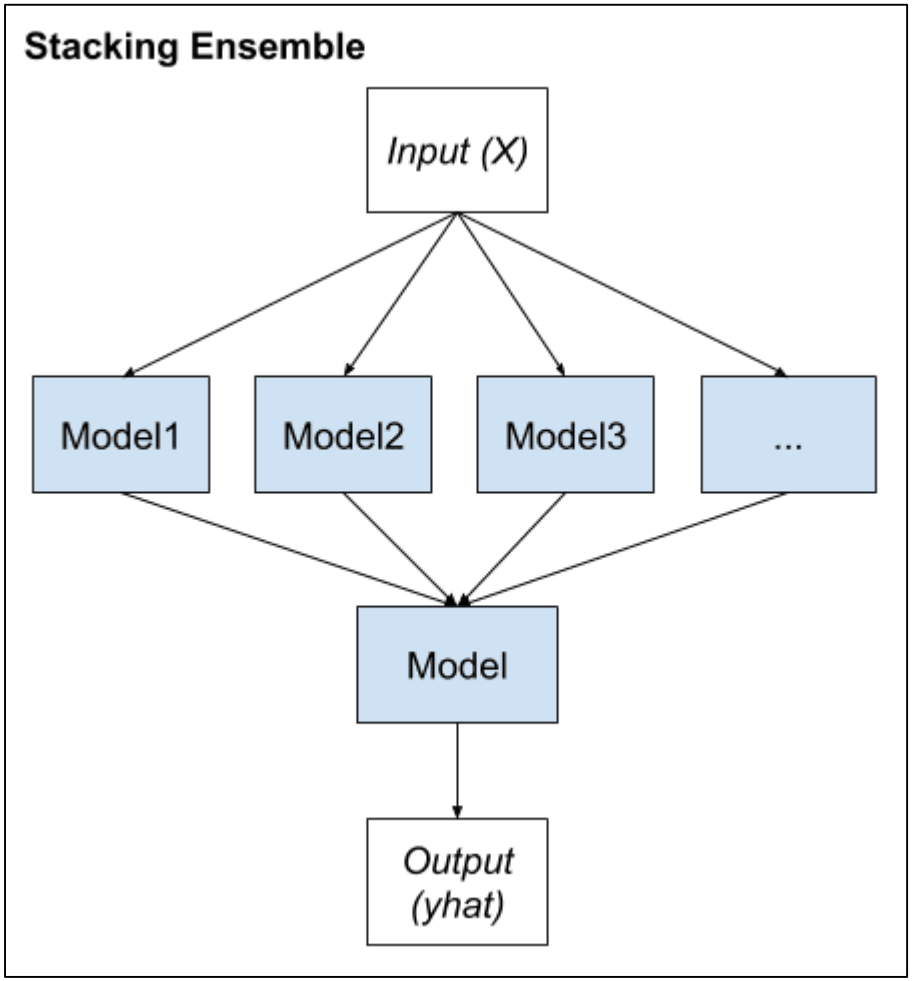
- Uma maneira de garantir a diversidade dos classificadores de base é com o uso de diferentes algoritmos para a produção dos classificadores
- Nesse caso, temos um conjunto heterogêneo de classificadores para combinar

↑ **Força preditiva dos modelos individuais**

Generalização em Pilha (Stacking)

- O stacking possui uma arquitetura de aprendizado em camadas
- Os classificadores no Nível_0 recebem como entrada os dados originais, e cada classificador produz uma predição
- Camadas sucessivas recebem como entrada as predições das camadas imediatamente precedentes, e a saída é passada para a próxima camada
- Um único classificador no nível mais alto produz a predição final

Generalização em Pilha (Stacking)



https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/

Generalização em Pilha (Stacking)

- A ideia é que os classificadores dos níveis mais altos aprendam como os classificadores anteriores cometem erros, em qual classe eles concordam ou discordam, e usar o seu conhecimento para fazer predições

Generalização em Pilha (Stacking)

- A maioria dos trabalhos concentra-se na arquitetura duas camadas

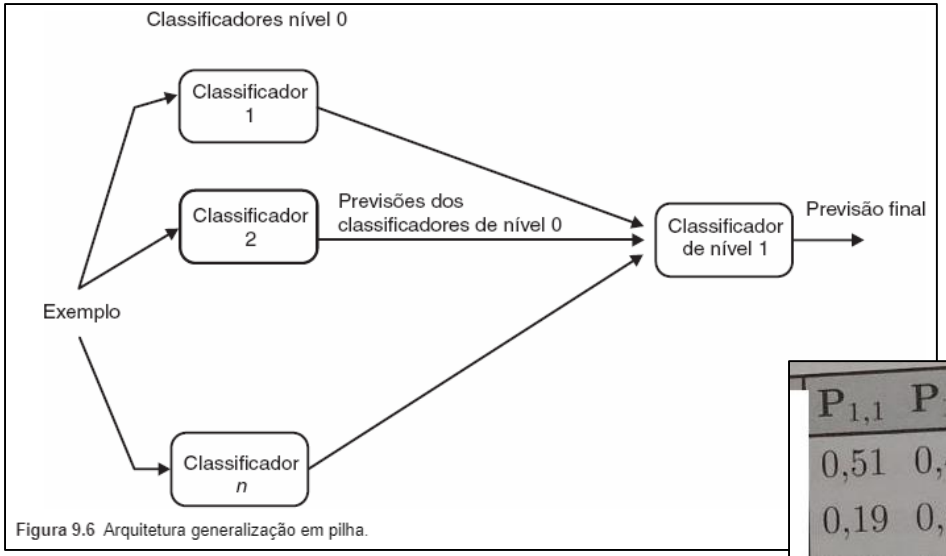


Figura 9.6 Arquitetura generalização em pilha.

V1	V2	V3	V4	V5	Classe
t	a	c	t	a	Membro
t	g	c	t	a	Membro
g	t	a	c	t	Não Membro
a	a	t	t	g	Membro
t	c	g	a	t	Não Membro
a	g	g	g	g	Membro

Conjunto de dados original

$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$P_{3,2}$	Classe
0,51	0,49	0,13	0,87	0,12	0,88	Membro
0,19	0,81	0,07	0,93	0,81	0,19	Membro
0,68	0,32	0,55	0,45	0,69	0,31	Não Membro
0,74	0,26	0,66	0,34	0,94	0,06	Membro
0,62	0,38	0,01	0,99	0,78	0,22	Não Membro
0,65	0,35	0,90	0,10	0,55	0,45	Membro

Conjunto de dados de $Nível_1$

Generalização em Pilha (Stacking)

- Passos
  - (1) Treinar cada um dos classificadores Nível_0 usando validação cruzada com o método deixar-um-de-fora da seguinte forma:
    - para cada exemplo no conjunto de treinamento, deixe um de fora e treine com os demais exemplos
    - depois do treinamento, classifique o exemplo excluído
    - crie um vetor a partir das predições de todos os classificadores Nível_0 e a classe atual daquele exemplo

Generalização em Pilha (Stacking)

- Passos
  - (2) Treinar o classificador Nível_1, usando como conjunto de treinamento a coleção de vetores gerados nos passos anteriores

  O número de exemplos nos dados Nível_1 é igual ao número de exemplos no conjunto de treinamento original

Generalização em Pilha (Stacking)

- Passos
  - (3) No passo 1, classificadores são gerados usando um método deixar um de fora

  Para explorar completamente o conjunto de treinamento, todos os classificadores Nível_0 são treinados novamente usando o conjunto de treinamento inteiro

  Os modelos gerados são usados para classificar os exemplos no conjunto de teste

Generalização em Pilha (Stacking)

- Passos
  - Predição: quando um novo exemplo é apresentado, este é classificado por todos os classificadores Nível_0

  O vetor de predições é então classificado pelo classificador Nível_1, que produz como saída a predição final para o exemplo

Generalização em Pilha (Stacking)

- Existem variações em relação ao modelo básico descrito
- Contudo, em todos os casos, é uma técnica sofisticada para **reduzir o erro em função da redução do bias (↓bias)**

Generalização em Cascata (Cascading)

- É uma composição sequencial de classificadores que em cada nível de generalização aplica-se um operador construtivo
- O operador construtivo constrói novos atributos
- Dados:
  - Um conjunto de treinamento D
  - Um conjunto de teste T
  - Dois algoritmos $\Im 1$ e $\Im 2$

Os próximos passos representam a sequência básica da generalização em cascata - existem extensões que incluem, por exemplo, a composição de *nc* classificadores

Generalização em Cascata (Cascading)

- A generalização em cascata procede como segue:
  - O algoritmo $\Im 1$ gera um classificador, f_1, usando o conjunto de treinamento D
  - O modelo gerado, f_1, classifica todos os exemplos de treinamento e teste
  - Assume-se que o resultado de aplicar o modelo f_1 a um exemplo é uma distribuição de probabilidade da classe
  - O operador construtivo concatena cada exemplo x com o vetor c (probabilidade da classe)

Generalização em Cascata (Cascading)

- A generalização em cascata procede como se segue (cont.):
  - O resultado da aplicação do operador construtivo é um novo conjunto de dados, com o mesmo número de exemplos do conjunto original, mas em que cada exemplo é acrescido de novos atributos, um novo atributo para cada classe
  - Cada novo atributo é a probabilidade de o exemplo pertencer a uma das classes dadas pelo modelo
  - O classificador $\Im 2$ aprende com os novos dados de treinamento

Generalização em Cascata (Cascading)

Tabela 9.3 Dois exemplos do conjunto de dados de $Nível_0$ no problema `Monks-2`

Cabeça	Corpo	Sorriso	Objeto	Cor	Gravata	Classe
redonda	redondo	sim	espada	vermelho	sim	inimigo
redonda	redondo	não	balão	azul	não	amigo

Naive Bayes

Tabela 9.4 Dois exemplos do conjunto de dados de $Nível_1$

Cabeça	Corpo	Sorriso	Objeto	Cor	Gravata	P(amigo)	P(inimigo)	Classe
redonda	redondo	sim	espada	vermelho	sim	0,135	0,864	inimigo
redonda	redondo	não	balão	azul	não	0,303	0,696	amigo

Generalização em Cascata (Cascading)

C4.5

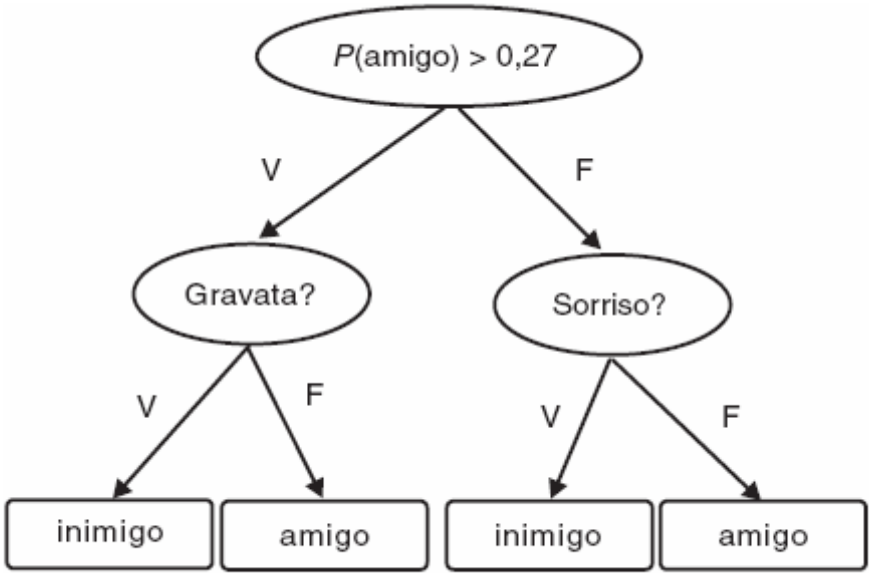


Figura 9.7: A árvore gerada pelo C4.5 ∇ *naive* Bayes no conjunto de dados Monks-2.

Erros

- **C4.5 = 32,9%**
- **Naive Bayes = 34,2%**
- **C4.5 ∇ Naive Bayes = 8,9%**

Generalização em Cascata (Cascading)

- Generalização em cascata pode ser considerada um caso especial de generalização em pilha, principalmente em função da estrutura de aprendizado em camadas. Contudo:
  - Enquanto a generalização em pilha tem natureza paralela, a generalização em cascata é sequencial
    - O efeito é que classificadores intermediários têm acesso aos atributos originais mais as predições dos classificadores de baixo nível

Generalização em Cascata (Cascading)

- Generalização em cascata pode ser considerada um caso especial de generalização em pilha, principalmente em função da estrutura de aprendizado em camadas. Contudo (cont.):
  - Enquanto o objetivo final da generalização em pilha é combinar predições, o objetivo da generalização em cascata é obter um modelo que possa usar termos na linguagem de representação dos classificadores de mais baixo nível
  - Na generalização em cascata, os classificadores de mais baixo nível adiam a decisão final para os classificadores de alto nível

Generalização em Cascata (Cascading)

- Generalização em cascata pode ser considerada um caso especial de generalização em pilha, principalmente em função da estrutura de aprendizado em camadas. Contudo (cont.):
  - O perfil para os classificadores no **início** da sequência é **baixa variância**, enquanto para os classificadores no **fim** da sequência **baixo bias**