

Ensembles

Prof. Jefferson T. Oliva

Aprendizado de Máquina e Reconhecimento de Padrões (AM28CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco



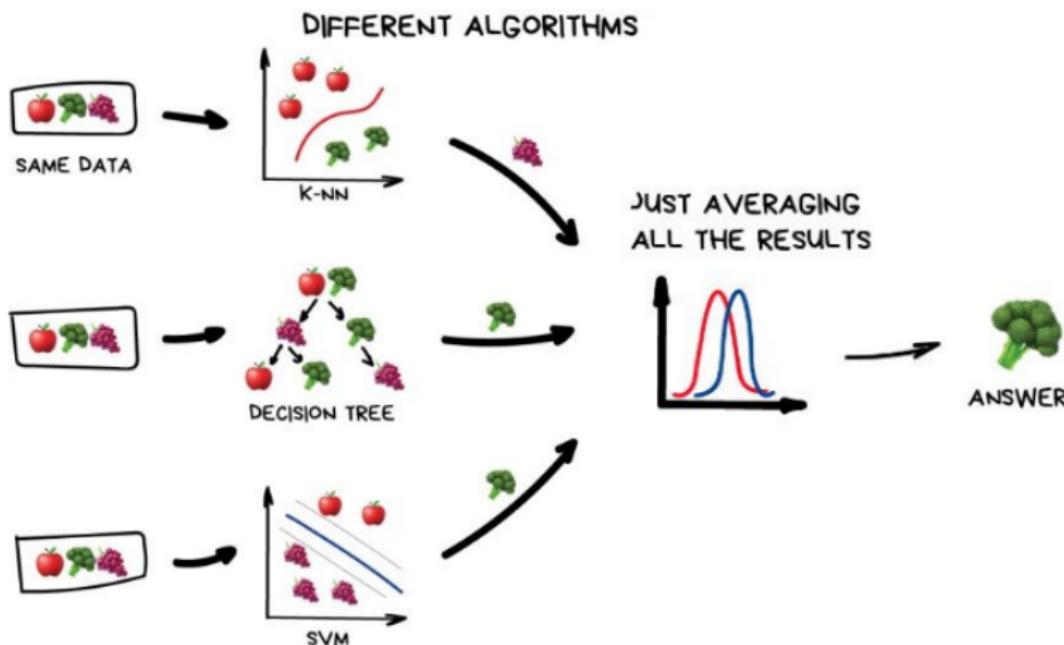
- Ensemble
- Classificação por Votação
- Bagging
- Random Forest
- Adaboost

- Procuramos uma segunda, terceira ou mais opiniões em diversas situações
 - Saúde
 - Financeiro
 - Entre outros assuntos
- Para cada opinião, determinados um peso que influencia em uma decisão final
- Em outras palavras, procuramos a melhor opinião possível

Ensemble

Ensemble

- Também conhecido como comitê de classificadores



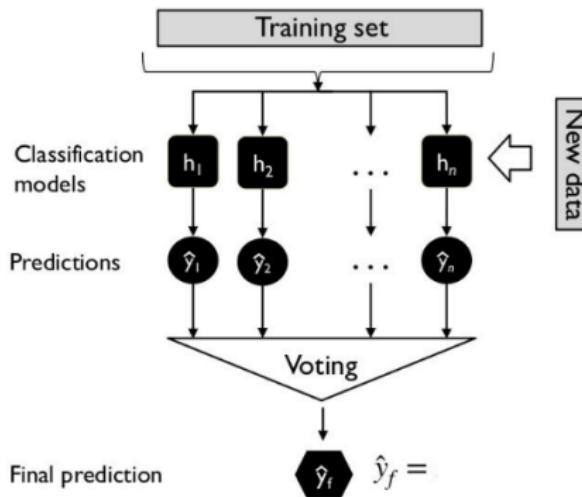
- Os métodos ensembles são geralmente divididos em duas famílias
 - Baseado em média: o princípio norteador é construir vários estimadores de forma independente e, em seguida, calcular a média de suas previsões
 - O estimador combinado é geralmente melhor do que qualquer um dos estimadores de base única, pois sua variância é reduzida
 - Exemplos de métodos: *bagging*, floresta randômica (*random forest*) ...
 - *Boosting*: os estimadores base são construídos sequencialmente e tenta-se reduzir o viés do estimador combinado
 - Combinação de vários modelos fracos para produzir um conjunto poderoso
 - Exemplos de métodos: AdaBoost, *Gradient Tree Boosting* ...

- Acurácia e diversidade são vitais para a construção de classificadores ensembles
- Todos os métodos visam construir boas hipóteses individuais com erros não correlacionados (diversidade)
- Principais abordagens:
 - Manipulação de hipóteses/classificadores: votação e empilhamento (*stacking*)
 - Manipulação de exemplos de treinamento: *bagging* e *boosting*
 - Manipulação de características de entrada: subespaço de atributos
 - Manipulação de exemplos e de atributos de treinamento: *floresta randômica*
 - Manipulação de saídas alvos (*target*): classificação multiclasse
 - Injeção de aleatoriedade: redes neurais ensembles

Classificação por Votação

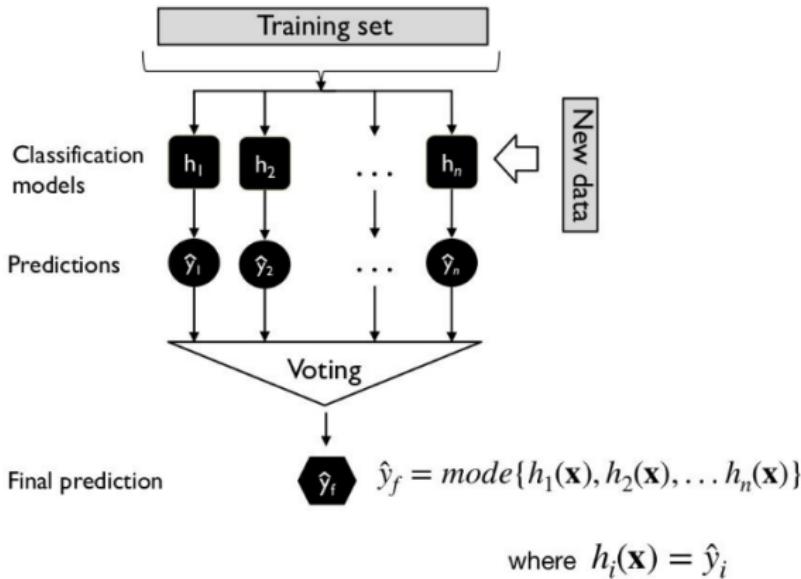
Classificação por Votação

- A ideia é combinar classificadores de aprendizado de máquina conceitualmente diferentes e usar a votação majoritária ou a média das probabilidades previstas (voto suave) para a predição dos rótulos das classes
- Essa abordagem pode ser útil para um conjunto de modelos com desempenho igualmente bom para o equilíbrio de duas fraquezas individuais



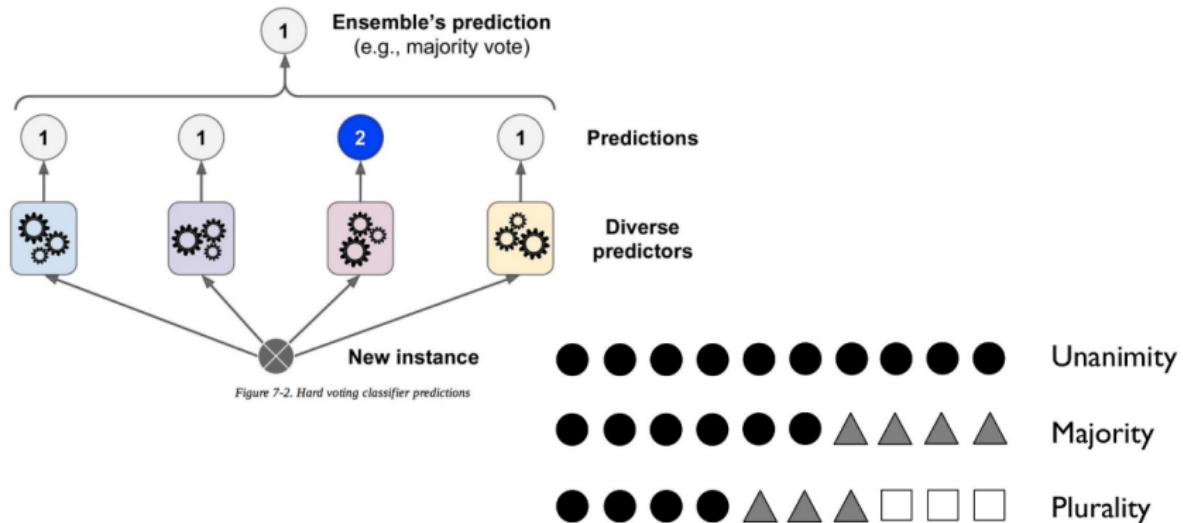
Classificação por Votação

- Rotulação pela classe majoritária (votação rígida): a classe prevista para uma nova amostra é o rótulo majoritário predito pelos classificadores individuais



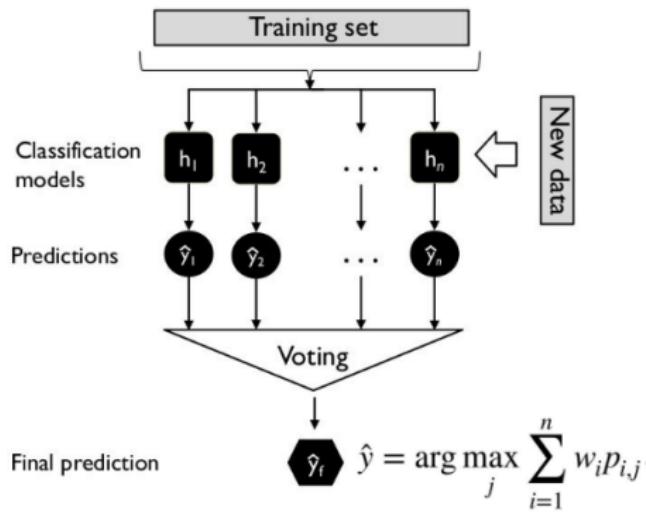
Classificação por Votação

- Rotulação pela classe majoritária (votação rígida)



Classificação por Votação

- Rotulação utilizando probabilidades médias ponderadas (votação suave): diferentemente da votação rígida, a votação suave retorna o rótulo da classe como *argmax* da soma das probabilidades previstas
 - $p_{i,j}$: probabilidade de associação da classe do i -ésimo classificador para o rótulo j
 - w_i : parâmetro de ponderação opcional, cuja configuração padrão é $w_i = \frac{1}{n}, \forall w_i \in \{w_1, \dots, w_n\}$



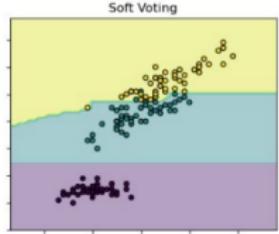
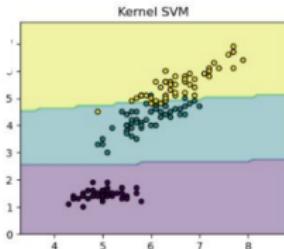
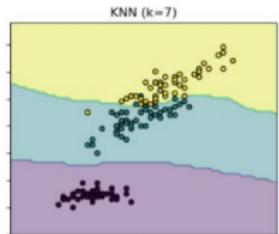
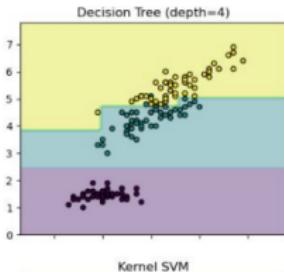
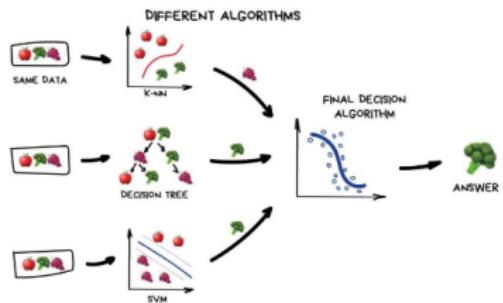
- Rotulação utilizando probabilidades médias ponderadas (votação suave)
 - Dado um problema de classificação binária, onde $i \in \{0, 1\}$, suponha que o nosso ensemble possa fazer as seguintes predições:
 - $C_1(x) \rightarrow [0.9, 0.1]$
 - $C_2(x) \rightarrow [0.8, 0.2]$
 - $C_3(x) \rightarrow [0.4, 0.6]$

$$\hat{y} = \arg \max_i \sum_{j=1}^n w_j p_{i,j}$$

- Utilizando pesos uniformes, temos os seguintes resultados
 - $p(i_0|x) = \frac{0.9+0.8+0.4}{3} = 0,7$
 - $p(i_1|x) = \frac{0.1+0.2+0.6}{3} = 0,3$
 - $\arg \max_i [p(i_0|x), p(i_1|x)] = 0$

- Rotulação utilizando probabilidades médias ponderadas (votação suave)
 - Dado um problema de classificação binária, onde $i \in \{0, 1\}$, suponha que o nosso ensemble possa fazer as seguintes predições:
 - $C_1(x) \rightarrow [0.9, 0.1]$
 - $C_2(x) \rightarrow [0.8, 0.2]$
 - $C_3(x) \rightarrow [0.4, 0.6]$
 - Caso utilizamos os pesos $w = \{0.1, 0.1, 0.8\}$, temos uma predição diferente:
 - $p(i_0|x) = 0.1 * 0.9 + 0.1 * 0.8 + 0.8 * 0.4 = 0,49$
 - $p(i_1|x) = 0.1 * 0.1 + 0.1 * 0.2 + 0.8 * 0.6 = 0,51$
 - $\arg \max_i[p(i_0|x), p(i_1|x)] = 1$

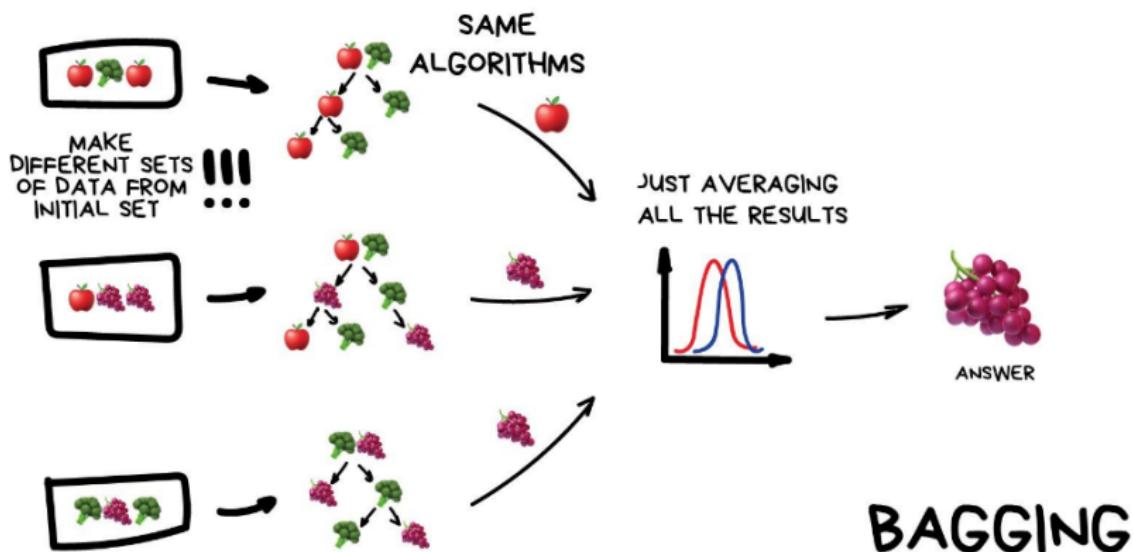
Classificação por Votação



Bagging

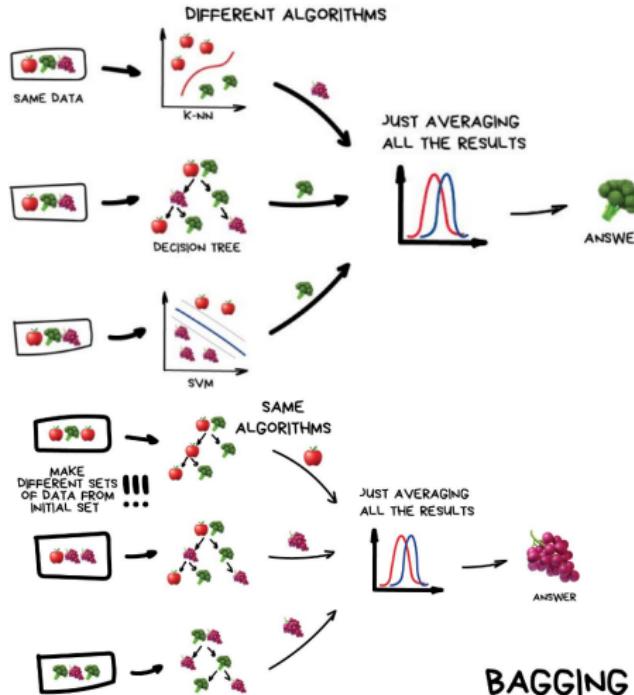
- *Bootstrap Aggregating*
 - Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.
- Família ensemble: baseado média (vários estimadores independentemente)
- Erros não correlacionados (diversidade)
 - Manipulação de exemplos de treinamento
 - Manipulação de características de entrada

Bagging

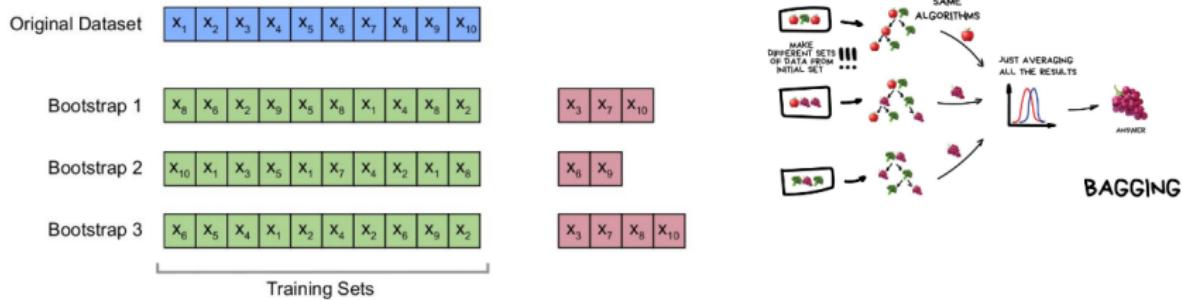


Bagging

- Votação vs. bagging



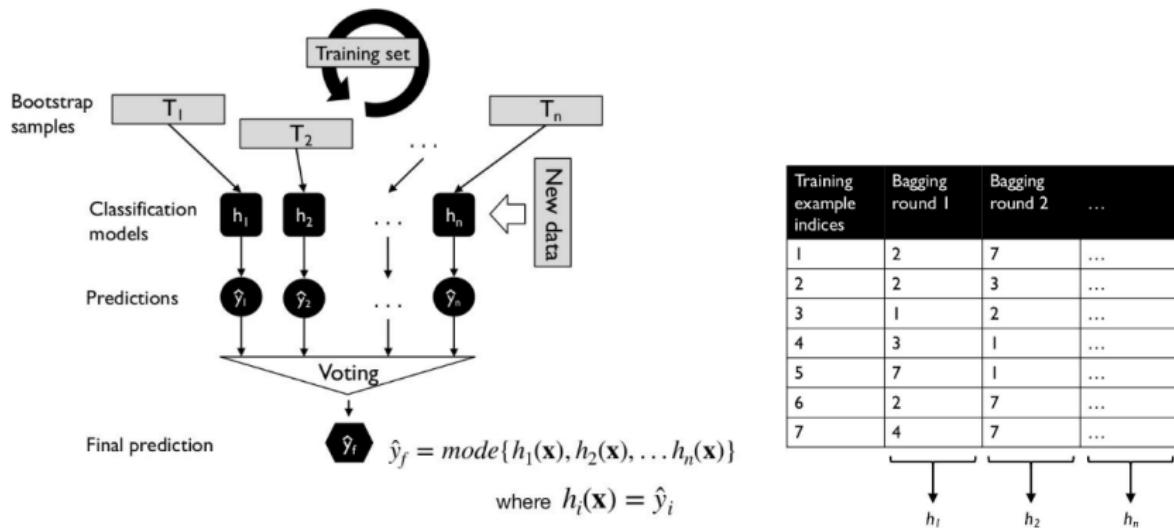
Bagging



Algorithm 1 Bagging

- 1: Let n be the number of bootstrap samples
 - 2:
 - 3: **for** $i=1$ to n **do**
 - 4: Draw bootstrap sample of size m , \mathcal{D}_i
 - 5: Train base classifier h_i on \mathcal{D}_i
 - 6: $\hat{y} = \text{mode}\{h_1(\mathbf{x}), \dots, h_n(\mathbf{x})\}$
-

Bagging



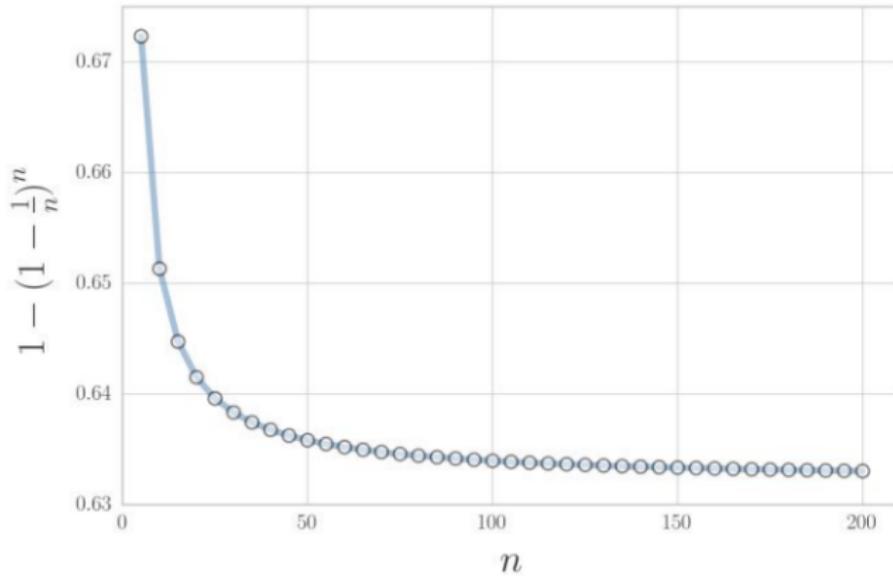
Bagging

- Probabilidade de amostragem

$$P(\text{amostra não escolhida}) = \left(1 - \frac{1}{n}\right)^2$$

$$\frac{1}{e} \approx 0,368, n \rightarrow \infty$$

$$P(\text{amostra escolhida}) = 1 - \left(1 - \frac{1}{n}\right)^2 \approx 0,632$$

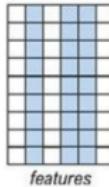


- Os métodos *bagging* formam uma classe de algoritmos que constroem várias instâncias de um ensemble utilizando subconjuntos aleatórios gerados a partir conjunto de treinamento original e, em seguida, agregam suas previsões individuais para formar uma previsão final
- Existem diversos métodos *bagging*, cujas principal diferença encontra-se pela forma em que os subconjuntos de treinamento são formados:
 - Colagem (*pasting*): subconjuntos aleatórios de amostras são gerados para o treinamento do modelo
 - *Bagging*: amostras são extraídas com reposição
 - Subespaços aleatórios: os subconjuntos aleatórios de características
 - *Patches* aleatórios: os estimadores base são construídos em subconjuntos de amostras e de características

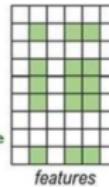
Bagging



**bagging
(sample instances)**
`max_samples=0.75,
bootstrap=True,
max_features=1.0,
bootstrap_features=False`



**random subspaces
(sample features)**
`max_samples=1.0,
bootstrap=False,
max_features=0.5,
bootstrap_features=True`

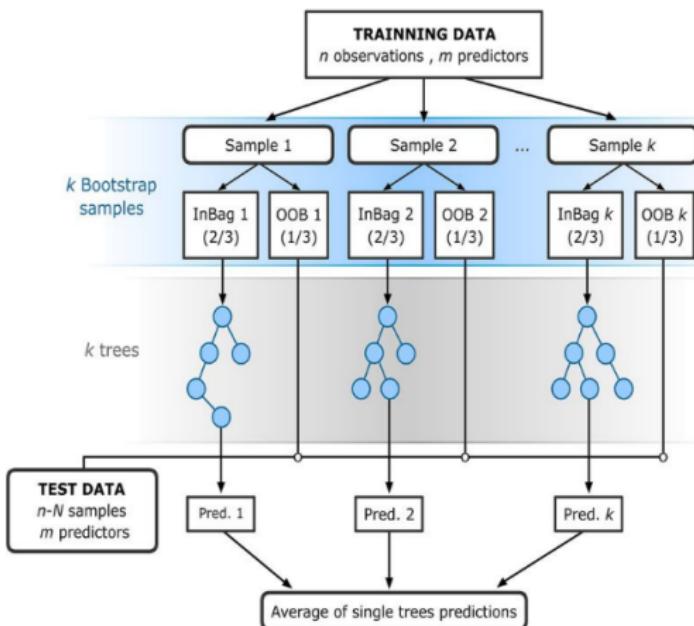


**random patches
(sample both)**
`max_samples=0.75,
bootstrap=True,
max_features=0.5,
bootstrap_features=True`

Random Forest

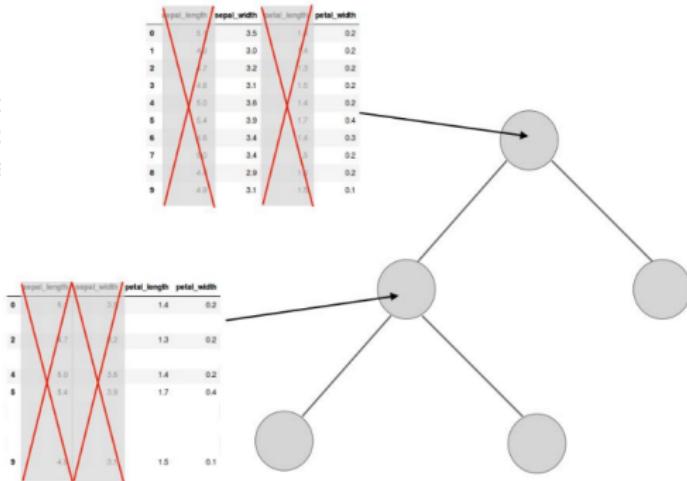
Random Forest

- Floresta randômica | floresta aleatória
- Cada árvore no conjunto é construída a partir de uma amostra retirada com substituição (ou seja, uma amostra *bootstrap*) do conjunto de treinamento



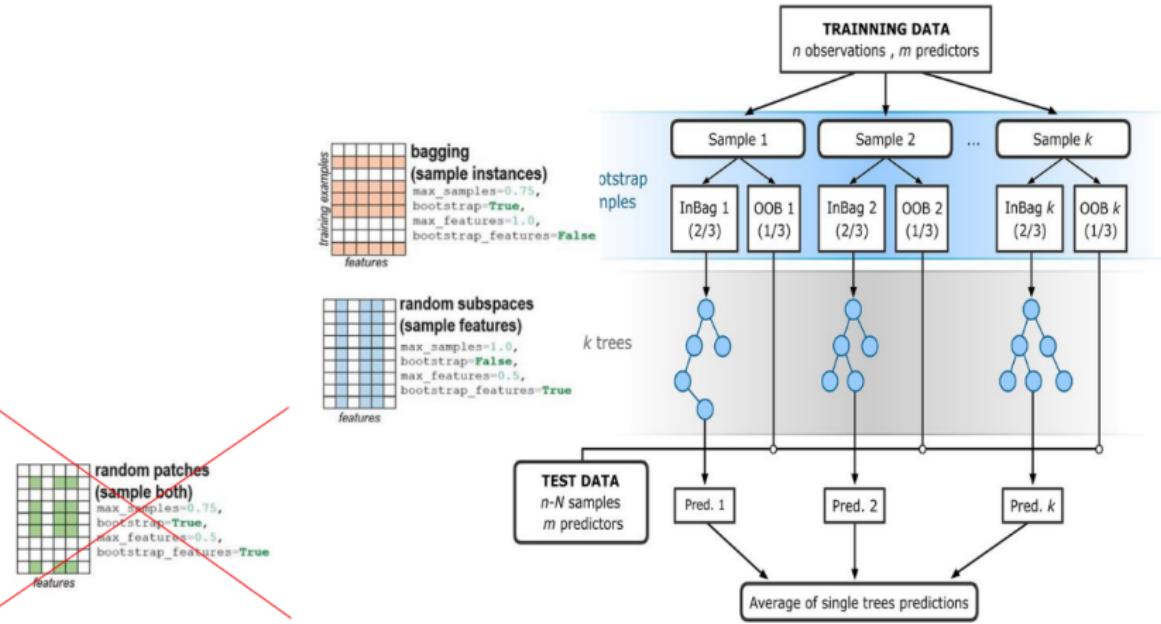
Random Forest

- Ao dividir cada nó durante a construção de uma árvore, a melhor divisão é encontrada a partir de todos as características de entrada ou de um subconjunto aleatório de atributos



Random Forest

- Bagging com árvores + subconjuntos de características aleatórias

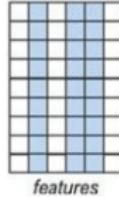


- Ho, Tin Kam. "The random subspace method for constructing decision forests." *IEEE transactions on pattern analysis and machine intelligence* 20.8 (1998): 832-844.
 - "Nosso método se baseia em um procedimento autônomo e pseudoaleatório para selecionar um pequeno número de dimensões de um determinado espaço de características"
 - No método subespaço aleatório, cada árvore recebe um subconjunto aleatório de características
- Breiman, Leo. "Random Forests" *Machine learning* 45.1 (2001): 5-32.
 - "... uma floresta aleatória com características aleatórias é formada pela seleção aleatória, em cada nó, de um pequeno grupo de variáveis ??de entrada para divisão"

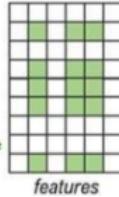
Random Forest



**bagging
(sample instances)**
`max_samples=0.75,
bootstrap=True,
max_features=1.0,
bootstrap_features=False`



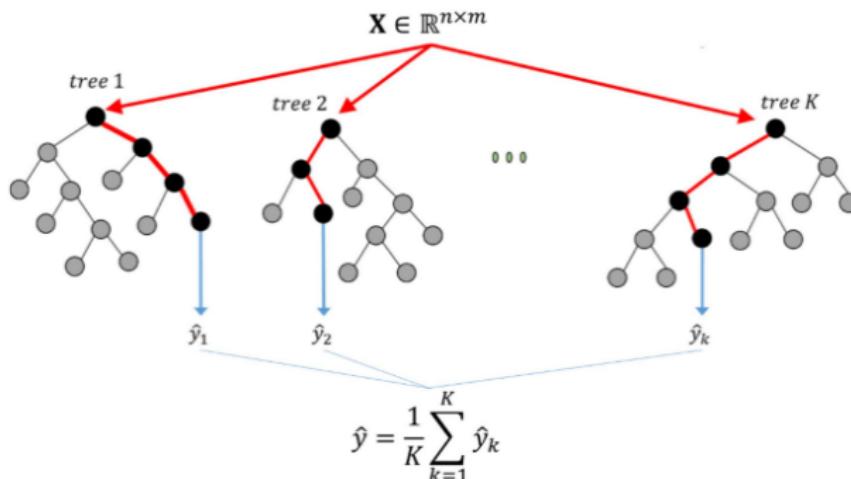
**random subspaces
(sample features)**
`max_samples=1.0,
bootstrap=False,
max_features=0.5,
bootstrap_features=True`



**random patches
(sample both)**
`max_samples=0.75,
bootstrap=True,
max_features=0.5,
bootstrap_features=True`

Random Forest

- Breiman, Leo. ?Random Forests? Machine learning 45.1 (2001): 5-32.
- Em contraste com a publicação original, a implementação do *scikit-learn* combina classificadores calculando a média de sua previsão probabilística, em vez de deixar cada classificador votar em uma única classe



- Limite superior para o erro de generalização

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

- $\bar{\rho}$: correlação média entre as árvores
- s : "força" do ensemble

Random Forest

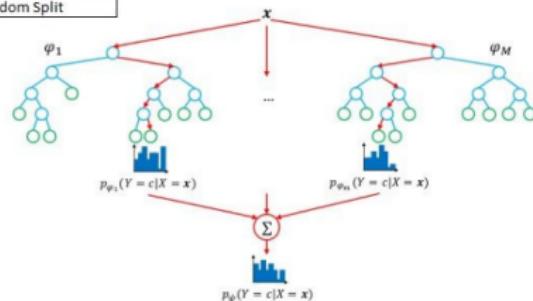
Árvores extremamente aleatórias (ExtraTrees)

- Componentes de uma árvore randômica
 - Cada árvore no conjunto é construída a partir de uma amostra retirada com reposição (*i.e.*, *bagging*) do conjunto de treinamento
 - Ao dividir cada nó durante a construção de uma árvore, a melhor divisão é encontrada em um subconjunto aleatório de características (*i.e.*, subespaços aleatórios)
- ExtraTree adiciona mais um componente aleatório
 - Os limiares são obtidos aleatoriamente para cada característica candidata e o melhor desses limiares gerados aleatoriamente seja escolhido como regra de divisão
 - Isso geralmente permite reduzir um pouco mais a variância do modelo, à custa de um aumento ligeiramente maior no viés

Random Forest

Árvores extremamente aleatórias (Extra Trees)

	Decision Tree	Random Forest	Extra Trees
Number of trees	1	Many	Many
No of features considered for split at each decision node	All Features	Random subset of Features	Random subset of Features
Bootstrapping(Drawing Sampling without replacement)	Not applied	Yes	No
How split is made	Best Split	Best Split	Random Split



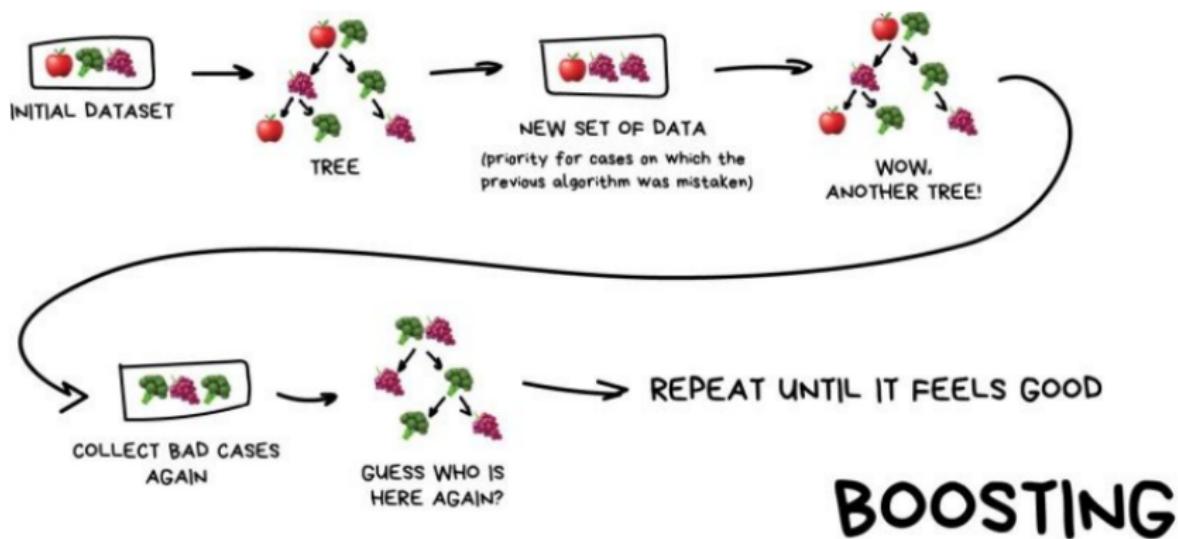
Randomization

- Bootstrap samples
 - Random selection of $K \leq p$ split variables
 - Random selection of the threshold
- } Random Forests } Extra-Trees

Adaboost

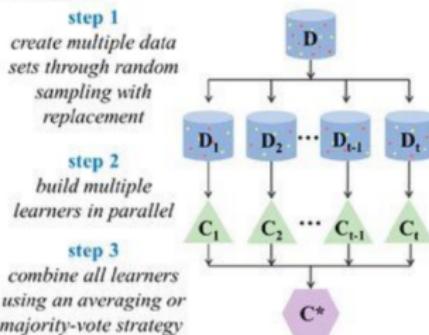
- Adaptive Boosting
 - Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1), 119-139.
- Método *boosting* (vários estimadores sequencialmente)
- Erros não correlacionados (diversidade)
 - Manipulação dos exemplos de treinamento
 - Aprendizes fracos ("weak learners")

Adaboost

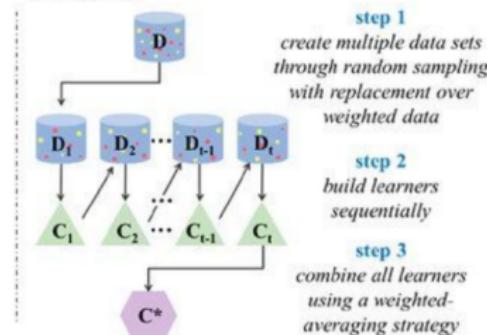


- O princípio básico do AdaBoost é ajustar uma sequência de aprendizes fracos (ou seja, modelos que são apenas ligeiramente melhores do que suposições aleatórias, como pequenas árvores de decisão) em versões repetidamente modificadas dos dados
- Métodos de média (votação, *bagging*, árvores aleatórias) X métodos sequenciais (AdaBoost)

(A) bagging

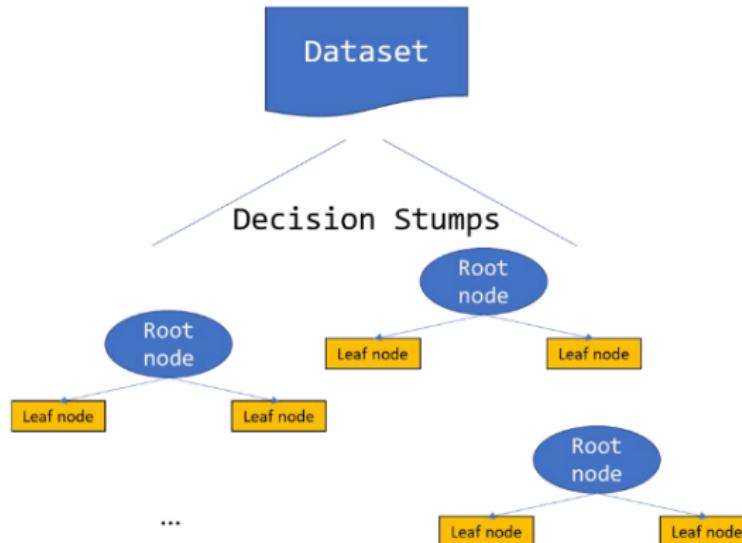


(B) boosting



- AdaBoost inicializa um vetor com pesos uniformes
- *Loop*
 - Aplicação do aprendiz fraco (ligeiramente melhor do que a adivinhação aleatória) nos exemplos de treinamento ponderados
 - Em vez do conjunto de treinamento original, pode-se extrair amostras por *bootstrap* com probabilidade ponderada
 - Aumento do peso para exemplos classificados incorretamente
 - Votação "suave" em classificadores treinados

- Aprendiz fraco: árvore de decisão para classificação binária com rótulos -1 e 1
 - $h(x) = s(1(x_k \leq t))$
- Onde:
 - $s(x) \in \{-1, 1\}$
 - $k \in \{1, \dots, K\}$ (k é o número de características)



Algorithm 1 AdaBoost

- 1: Initialize k : the number of AdaBoost rounds
 - 2: Initialize \mathcal{D} : the training dataset, $\mathcal{D} = \{\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, \dots, \langle \mathbf{x}^{[n]}, y^{[n]} \rangle\}$
 - 3: Initialize $w_1(i) = 1/n, \quad i = 1, \dots, n, \quad \mathbf{w}_1 \in \mathbb{R}^n$
 - 4:
 - 5: **for** $r=1$ to k **do**
 - 6: For all $i : \mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$ [normalize weights]
 - 7: $h_r := \text{FitWeakLearner}(\mathcal{D}, \mathbf{w}_r)$
 - 8: $\epsilon_r := \sum_i w_r(i) \mathbf{1}(h_r(i) \neq y_i)$ [compute error]
 - 9: if $\epsilon_r > 1/2$ then stop
 - 10: $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$ [small if error is large and vice versa]
 - 11: $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$
 - 12: Predict: $h_{ens}(\mathbf{x}) = \arg \max_j \sum_r^k \alpha_r \mathbf{1}[h_r(\mathbf{x}) = j]$
 - 13:
-

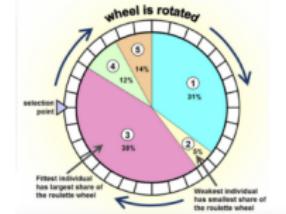
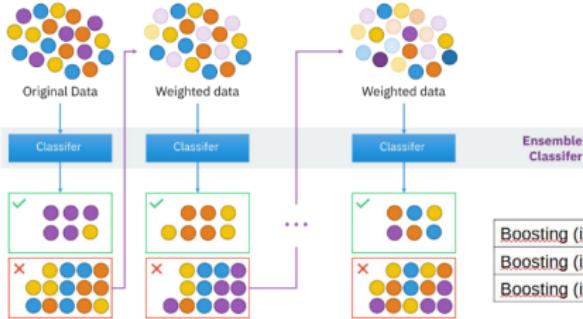
- Exemplo encontra-se no apêndice

• Boosting

- As modificações de dados em cada iteração, chamada de *boosting*, consistem na aplicação de pesos a cada uma das amostras de treinamento
- Inicialmente, esses pesos são definidos, de modo que a primeira etapa simplesmente treina um aprendiz fraco com os dados originais
- Para cada iteração sucessiva, os pesos da amostra são modificados individualmente e o algoritmo de aprendizado é reaplicado aos dados reponderados
- O peso atribuído a cada amostra de treinamento pode ser usado das seguintes maneiras:
 - ① Pode ser usado como uma distribuição amostral para realizar um *bootstrap* a partir do conjunto amostral dos dados originais
 - ② Pode ser usado pelo classificador base para aprender um modelo que atribui alto peso a exemplos de classificação difícil

Adaboost

Boosting



Boosting (iteration 1)	7	3	2	8	7	9	4	10	6	3
Boosting (iteration 2)	5	4	9	4	2	5	1	7	4	2
Boosting (iteration 3)	4	4	8	10	4	5	4	6	3	4

- Boosting

- No classificador base para aprender um modelo que atribui alto peso a exemplos de classificação difícil

weighted knn $d_w(x, y) = \sum_{i=1}^n w_i (x_i - y_i)^2$

replace with weights

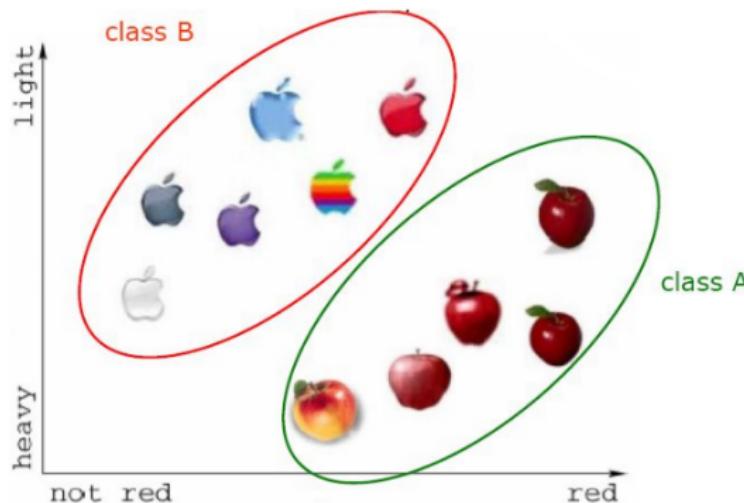
weighted decision tree

$$\text{Gini}_{divisão} = \sum_{t=1}^k \frac{N(v_t)}{N} \text{Gini}(v_t)$$

-  ALMEIDA, L. M.
Sistemas Baseados em Comitês de Classificadores.
Slides. Pós-Graduação em Ciência da Computação. UFPE, 2010.
-  BISHOP, C. M.
Pattern Recognition and Machine Learning.
Springer, 2006.
-  CASANOVA, D.
Ensemble methods. Aprendizado de Máquina.
Slides. Engenharia de Computação. Dainf/UTFPR, 2020.
-  SANT'ANA, D.
Ensemble methods. Aprendizado de Máquina.
Slides. Ensemble Learning. Departamento de Informática.
CEFET-RJ, 2020.

Adaboost

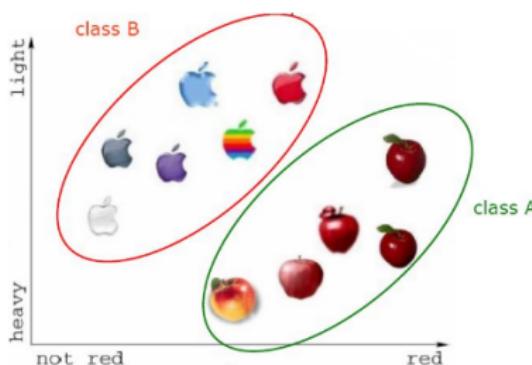
- Maças de plástico X maçãs reais



Initialization

- 1: Initialize k : the number of AdaBoost rounds
- 2: Initialize \mathcal{D} : the training dataset, $\mathcal{D} = \{\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, \dots, \langle \mathbf{x}^{[n]}, y^{[n]} \rangle\}$
- 3: Initialize $w_1(i) = 1/n, \quad i = 1, \dots, n, \quad \mathbf{w}_1 \in \mathbb{R}^n$

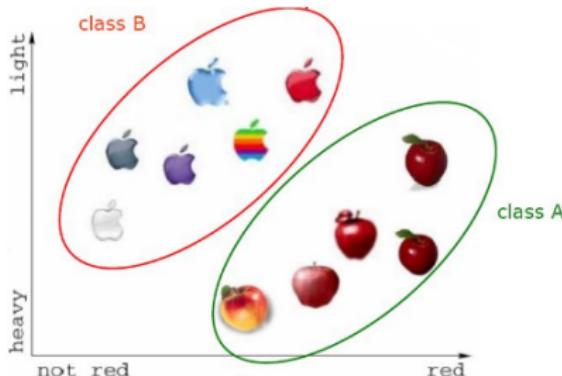
- $w_1(i) = 1/11 \quad \forall i \in D$
 - $w_1(i) = 0,091$



Normalize weights

5: **for** $r=1$ to k **do**
6: For all i : $\mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$ [normalize weights]

- $w_1(i) = 0,091/1$
 - $w_1(i) = 0,091$

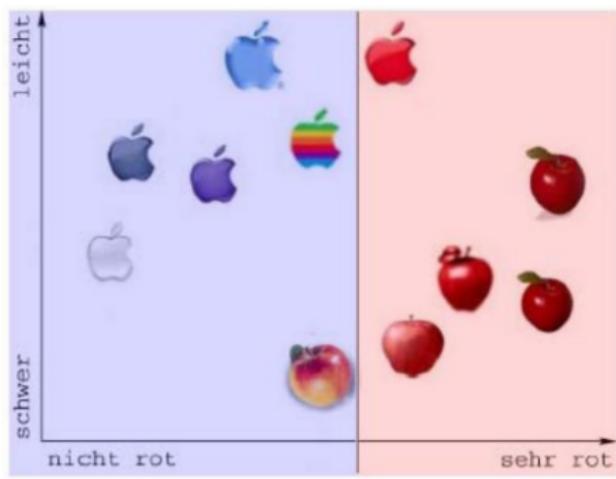


Fit a weak Learner and compute error

7: $h_r := \text{FitWeakLearner}(\mathcal{D}, \mathbf{w}_r)$

8: $\epsilon_r := \sum_i w_r(i) \mathbf{1}(h_r(i) \neq y_i)$ [compute error]

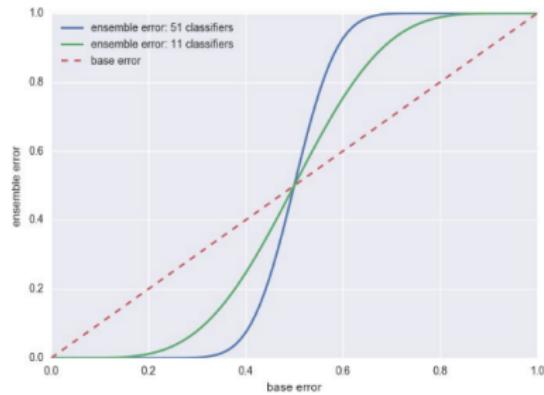
- Error (ϵ): 2/11
 - $\epsilon = 0.18$



Verify the sanity

9: if $\epsilon_r > 1/2$ then stop

If the error is greater than a random guessing, combine these classifier leads to an increase in the classification error



Estimador weight α

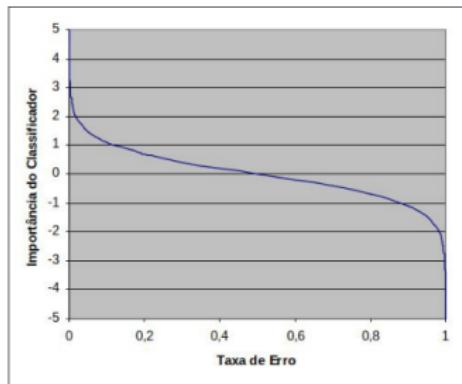
10: $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$ [small if error is large and vice versa]

The lower the error rate, the greater the importance attributed to the classifier!

- $\epsilon = 0.18$
- Estimador weight (α): $\frac{1}{2} \log(1-0,18/0,18)$
 - $\alpha_1 = 0,75$



$$\alpha_r = \frac{1}{2} \log \left(\frac{1 - \epsilon_r}{\epsilon_r} \right)$$



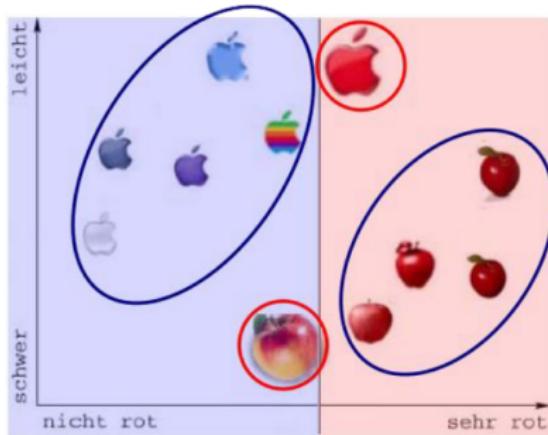
Weights update

- $w_1(i) = 0,091 \quad \forall i \in D$
- $\alpha_1 = 0,75$

$$11: \quad w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$$

The parameter α_r is used to update the weights of the training examples

- Correctly classified records ($h_r(\mathbf{x}^i) = y^i$):
 - weight decreases;
 - $w_{r+1}(i) = 0,091 * \exp(-0,75)$
 - $w_{r+1}(i) = 0,091 * 0,047 = 0,043$
- Wrongly classified records ($h_r(\mathbf{x}^i) \neq y^i$)
 - weight increases;
 - $w_{r+1}(i) = 0,091 * \exp(0,75)$
 - $w_{r+1}(i) = 0,091 * 2,117 = 0,193$



Weights update

$$11: \quad w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$$

- $w_1(i) = 0,091 \quad \forall i \in D$
- $w_2(i) = 0,043$ if correct $w_2(i) = 0,193$ if not



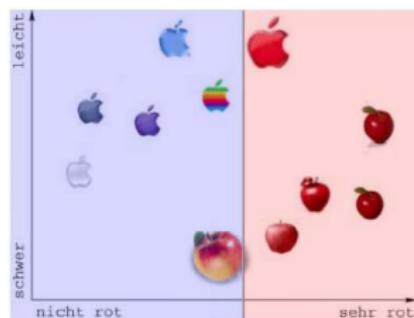
Repeat / Normalize weights

5: **for** $r=1$ to k **do**

6: For all i : $\mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$ [normalize weights]

$$(0,043 * 9) + (0,193 * 2) = 0,773$$

- Weights of correctly classified samples
 - $w_2(i) = 0,043/0,773 = 0,056$
- Weights of incorrectly classified samples
 - $w_2(i) = 0,193/0,773 = 0,248$

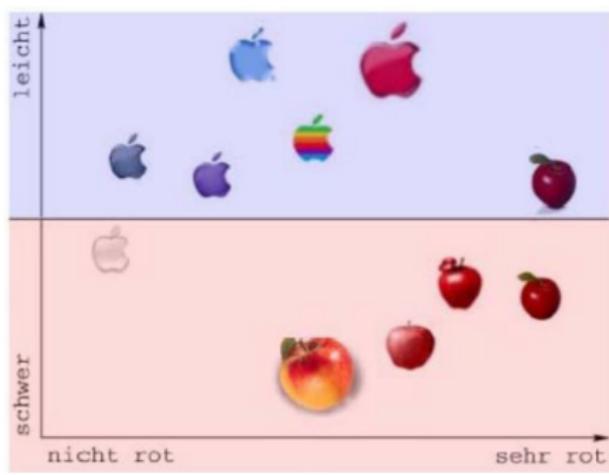


Fit a weak Learner and compute error

7: $h_r := \text{FitWeakLearner}(\mathcal{D}, \mathbf{w}_r)$

8: $\epsilon_r := \sum_i w_r(i) \mathbf{1}(h_r(i) \neq y_i)$ [compute error]

- Error (ϵ): 2/11
 - $\epsilon = 0.18$



Estimador weight α and Weights update

9: if $\epsilon_r > 1/2$ then stop

10: $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$ [small if error is large and vice versa]

11: $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$

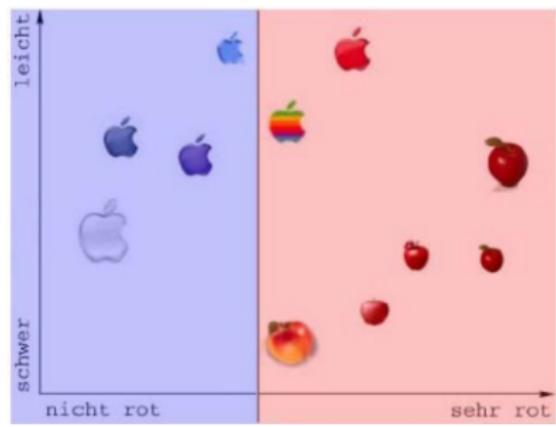
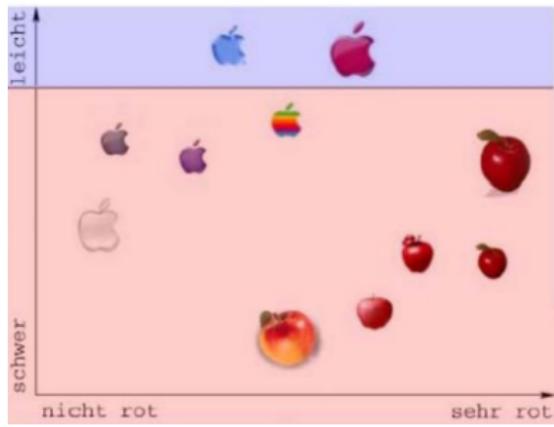


Repeat

```

5: for r=1 to k do
6:   For all  $i : \mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$  [normalize weights]
7:    $h_r := FitWeakLearner(\mathcal{D}, \mathbf{w}_r)$ 
8:    $\epsilon_r := \sum_i w_r(i) \mathbf{1}(h_r(i) \neq y_i)$  [compute error]
9:   if  $\epsilon_r > 1/2$  then stop
10:   $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$  [small if error is large and vice versa]
11:   $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$ 

```



Final classifier

12: Predict: $h_{ens}(\mathbf{x}) = \arg \max_j \sum_r^k \alpha_r \mathbf{1}[h_r(\mathbf{x}) = j]$

- $\alpha_1 = 0,75$
- $\alpha_2 = 0,75$
- $\alpha_3 = 0,29$
- $\alpha_4 = 0,75$

$$\mathbf{1}(h_r(i) = y_i) = \begin{cases} 0 & \text{if } h_r(i) \neq y_i \\ 1 & \text{if } h_r(i) = y_i \end{cases}$$

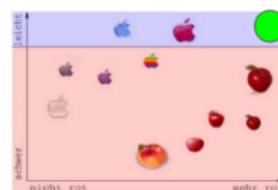
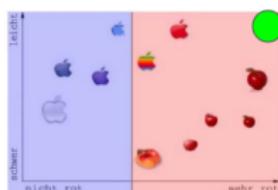


$$\arg \max_{\cdot,1} = 0,75*1(0) + 0,75*1(1) + 0,29*1(0) + 0,75*1(1) = 1,50$$



$$\arg \max_{\cdot,1} = 0,75*1(1) + 0,75*1(0) + 0,29*1(1) + 0,75*1(0) = 1,04$$

$$h_{ens} = -1$$



$$h_m(\mathbf{x}) = \text{sign} \left(\sum_{j=1}^m w_j h_j(\mathbf{x}) \right) \quad \text{for } h(\mathbf{x}) \in \{-1, 1\}$$

$$h_m(\mathbf{x}) = \arg \max_i \left(\sum_{j=1}^m w_j \mathbf{1}[h_j(\mathbf{x}) = i] \right) \text{ for } h(\mathbf{x}) = i, \quad i \in \{1, \dots, n\}$$

Final classifier

$$\text{sign} = 0,75*1 + 0,75*-1 + 0,29*1 + 0,75*-1 = -0,46$$

$$h_{\text{ens}} = -1$$



$$\arg \max_{-1} = 0,75*1(0) + 0,75*1(1) + 0,29*1(0) + 0,75*1(1) = 1,50$$



$$\arg \max_1 = 0,75*1(1) + 0,75*1(0) + 0,29*1(1) + 0,75*1(0) = 1,04$$

$$h_{\text{ens}} = -1$$

