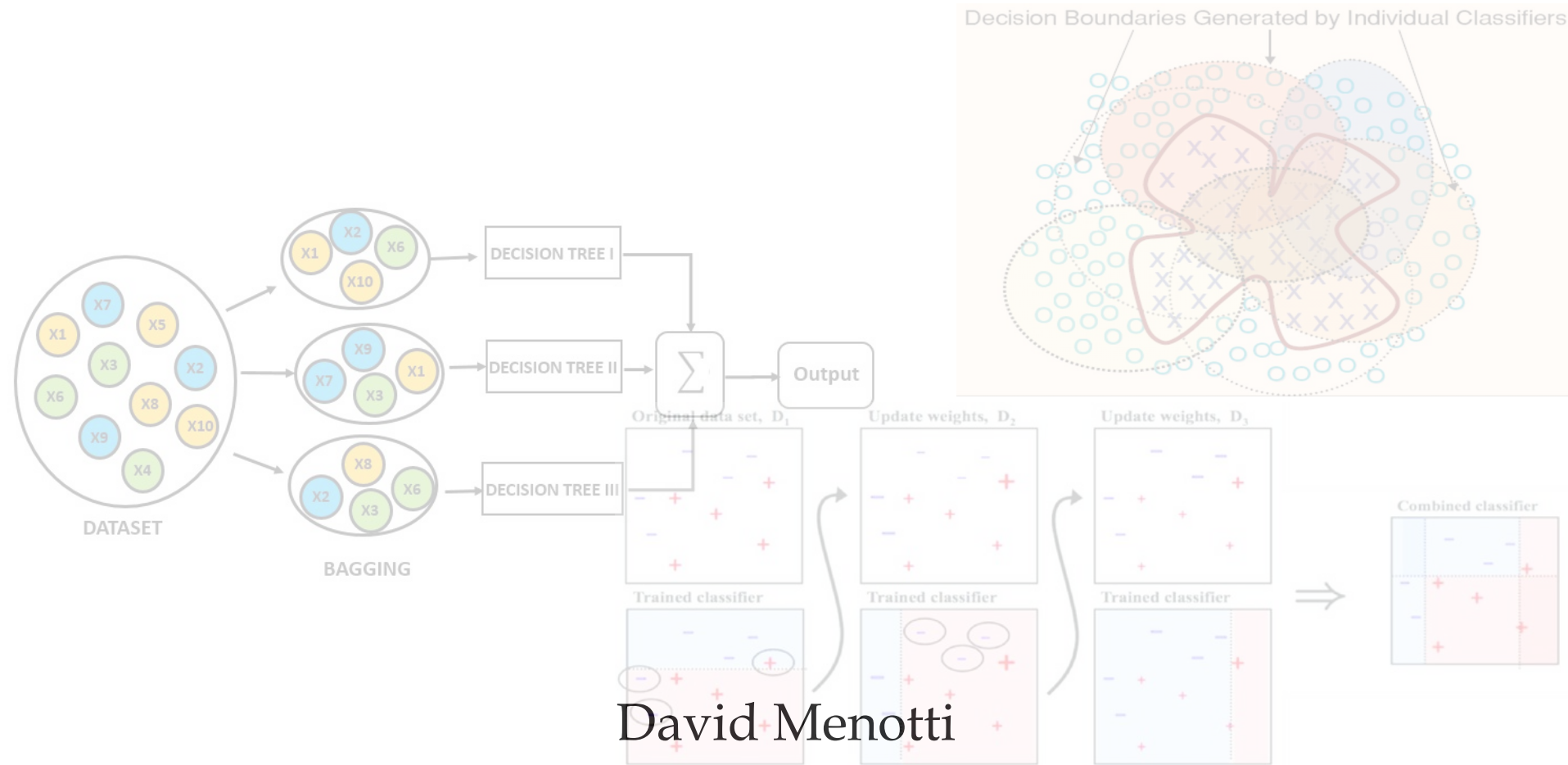


Múltiplos Classificadores



David Menotti

www.inf.ufpr.br/menotti/am-191

Hoje

- Múltiplos classificadores
 - Combinação de classificadores
 - Fusão de Classificadores
 - Comitês de Classificadores (*ensembles*)
 - *Bagging & Random forest*
 - *Boosting & Weak classifiers (Adaboost)*

Múltiplos Classificadores

- O uso de vários classificadores é uma estratégia bastante utilizada para aumentar o desempenho de sistemas de reconhecimento de padrões.
- A ideia é que os erros sejam minimizados através do uso de múltiplos classificadores ao invés de um único classificador.
- Resultados teóricos e experimentais demonstram a viabilidade do uso de múltiplos classificadores
 - Diferentes arquiteturas
 - Diferentes características (*subsets*)
 - Instâncias, atributos e classes

Múltiplos Classificadores

- O uso de múltiplos classificadores aparece na literatura com diferentes nomes:
 - *Multiple Classifier Systems* (MCS)
 - Fusão de classificadores
 - Combinação de classificadores
 - Mistura de classificadores
 - *Ensembles*
 - Comitês

Múltiplos Classificadores

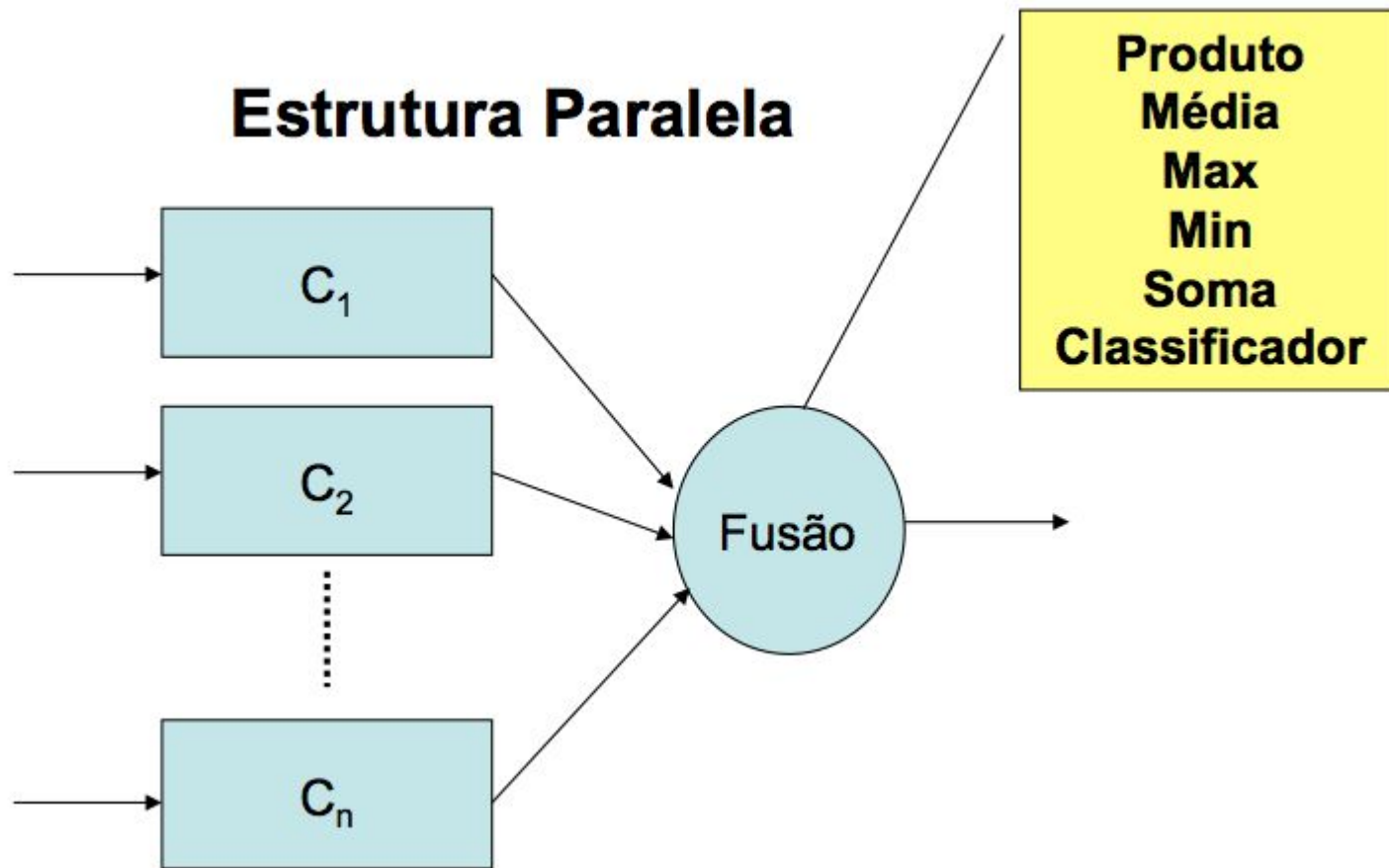
Podemos dividir o estudo de múltiplos classificadores

- Combinação de classificadores
 - Define regras de como combinar resultados produzidos por diferentes classificadores
- *Ensembles*
 - Como gerar diferentes classificadores para reduzir o erro de classificação
 - *Bagging, boosting, subspaces, etc..*
- Seleção dinâmica de classificadores

Combinando Classificadores

- Múltiplos classificadores podem ser combinados **paralelamente** ou em **série**.
 - Paralelo
 - Os classificadores C_1, C_2, \dots, C_n produzem decisões sobre um padrão desconhecido.
 - Todas essas decisões são então enviadas para um método de fusão que produzirá o resultado final.

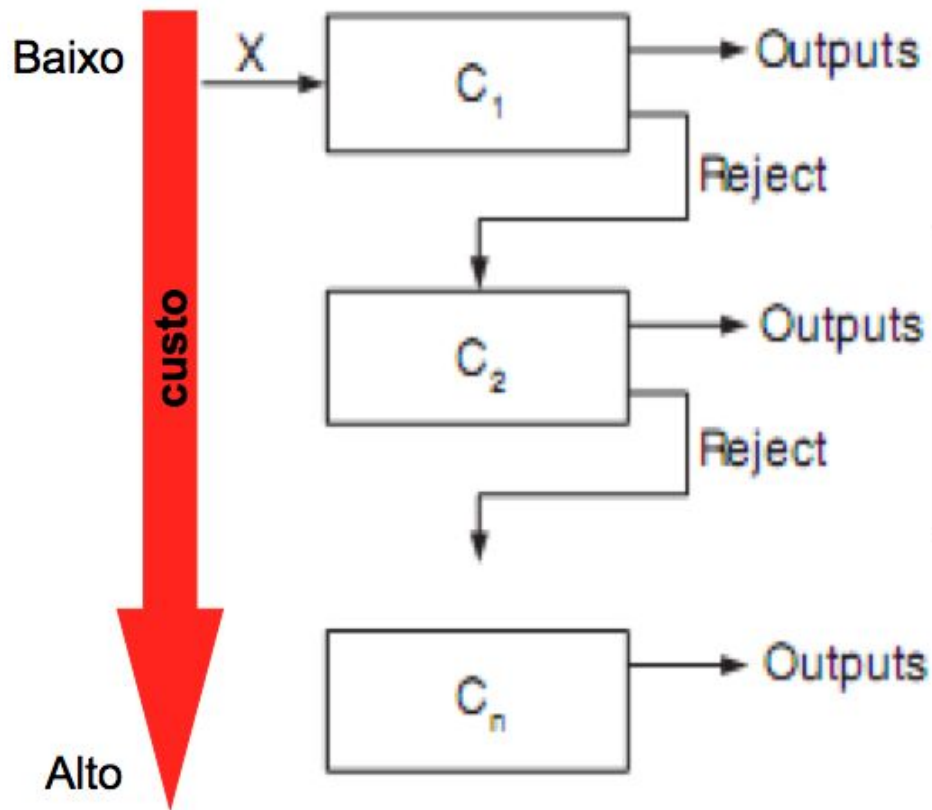
Combinando Classificadores



Combinando Classificadores

- Uma alternativa para a combinação paralela é a combinação em série
 - A cada estágio do sistema existe somente um classificador atuando no sistema
- A combinação em série pode ser dividida em duas abordagens
 - Redução de classes
 - Re-avaliação

Combinando Classificadores



Redução do custo computacional

A idéia é que a maioria dos padrões sejam classificados nos primeiros níveis, onde o custo computacional é mais baixo.

Combinando Classificadores

- Em geral, as saídas produzidas por classificadores podem ser divididas em três níveis:
 - **Abstrato**: o classificador produz como saída apenas o rótulo da classe escolhida.
 - e.g., C4.5
 - **Ranking**: O classificador produz uma lista ordenada com as possíveis classes.
 - e.g., kNN, pageRank
 - **Probabilidades/scores**: Além da lista ordenada, as probabilidades ou scores também são geradas.
 - e.g., NaiveBayes, SVM, MLP, etc.

Combinando Classificadores

Métodos de Fusão

- O método de fusão mais simples de ser implementado é o **voto majoritário**
 - Bastante utilizado com classificadores abstratos
 - Fácil implementação

		Probabilidade				
Classe		c1	c2	c3	c4	
	1	0.8	0.56	0.1	0.6	3
	2	0.2	0.44	0.9	0.4	1

Combinando Classificadores

Métodos de Fusão

- Se o classificador fornecer uma lista ordenada, outros métodos estão disponíveis
 - **Borda count**
 - O uso de lista de hipóteses ordenadas (*ranking*) é interessante quando a classe correta não aparece como a principal escolha (**Top 1**) do classificador, mas está próxima da saída correta Top 2 ou 3.
 - Bastante interessante para problemas com grandes léxicos (classes).

Combinando Classificadores

Exemplo **BordaCount**

- Considere os três classificadores abaixo, os quais fornecem uma lista ordenada com cinco classes
- Para cada classes, atribui-se um valor (rank). Nesse exemplo, foi atribuído 5 para o Top1, 4 para o Top2 e assim por diante.

Classificador 1

Classe	Rank
1	5
2	4
3	3
4	2
5	1

Classificador 2

Classe	Rank
4	5
2	4
5	3
1	2
3	1

Classificador 3

Classe	Rank
2	5
4	4
3	3
1	2
5	1

Combinando Classificadores

Exemplo **BordaCount**

- O resultado final é dado então pela soma dos *rankings*

Classificador 1

Classe	Rank
1	5
2	4
3	3
4	2
5	1

Classificador 2

Classe	Rank
4	5
2	4
5	3
1	2
3	1

Classificador 3

Classe	Rank
2	5
4	4
3	3
1	2
5	1

Resultado

Classe	Rank
2	13
4	11
1	9
3	7
5	5

Combinando Classificadores

Exemplo **BordaCount**

- Note que nesse exemplo, apenas um classificador (C_3) escolheu a classe 2
- Se o voto fosse utilizado, não haveria consenso.
- Pesos podem ser utilizados, caso existam classificadores com desempenhos bastante diferentes no conjunto.

Classificador 1

Classe	Rank
1	5
2	4
3	3
4	2
5	1

Classificador 2

Classe	Rank
4	5
2	4
5	3
1	2
3	1

Classificador 3

Classe	Rank
2	5
4	4
3	3
1	2
5	1

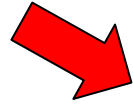
Resultado

Classe	Rank
2	13
4	11
1	9
3	7
5	5

Combinando Classificadores

- Se os classificadores produzem uma estimaco de probabilidade a posteriori, podemos utilizar outros mtodos de fuso, tais como:
 - Produto*
 - Soma*
 - Mdia
 - Mximo
 - Mnimo
- O score da combinao ainda pode ser utilizado para implementar um mtodo de **rejeio**.

Combinando Classificadores



Produto

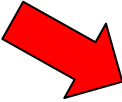
$$P(\omega_k) \prod P(\omega_k | x_i)$$

- É uma regra bastante severa para combinar classificadores.
- Basta que um dos classificadores não esteja de acordo com a decisão dos outros que o resultado final será penalizado.
- Quando usar ?
- Baseado no resultados desses quatro classificadores, qual seria a classe escolhida com base na regra do produto?

		Probabilidade				
Classe		c1	c2	c3	c4	
	1	0.8	0.56	0.1	0.6	0,027
	2	0.2	0.44	0.9	0.4	0,032

Combinando Classificadores

Soma


$$(1 - R)P(\omega_k) \sum_i P(\omega_k | x_i)$$

- Como o nome diz, a soma consiste em somar todas as estimativas de todos os classificadores envolvidos.
- Não é drástica como o produto.
- Resultados experimentais [Kittler 98 - On combining classifiers] mostram que a **soma** apresenta resultados bastante consistentes.
- Baseado no resultados desses quatro classificadores, qual seria a classe escolhida com base na regra da soma?

		Probabilidade				
Classe		c1	c2	c3	c4	
	1	0.8	0.56	0.1	0.6	0,515
	2	0.2	0.44	0.9	0.4	0,485

Combinando Classificadores

Média

$$\frac{1}{R} \sum P(\omega_k | x_i)$$

- Resultado médio de todos os classificadores
- Pode ser afetada por “*outliers*”
- Baseado no resultados desses quatro classificadores, qual seria a classe escolhida com base na regra da média?

Classe	Probabilidade				
	c1	c2	c3	c4	
1	0.8	0.56	0.1	0.6	4,502
2	0.2	0.44	0.9	0.4	5,557

Combinando Classificadores

Mínimo

$$\min(P(\omega_k | x_i))$$

- Os resultados encontrados pelo Min, são bastante similares aos resultados encontrados com o produto

Máximo

$$\max(P(\omega_k | x_i))$$

- A regra do máximo considera somente o classificador que maximiza a estimacão de probabilidade.
- Baseado no resultados desses quatro classificadores, qual seriam as classe escolhidas por essas duas regras?

		Probabilidade					
Classe		c1	c2	c3	c4		
	1	0.8	0.56	0.1	0.6	0,1	0,8
	2	0.2	0.44	0.9	0.4	0,2	0,9

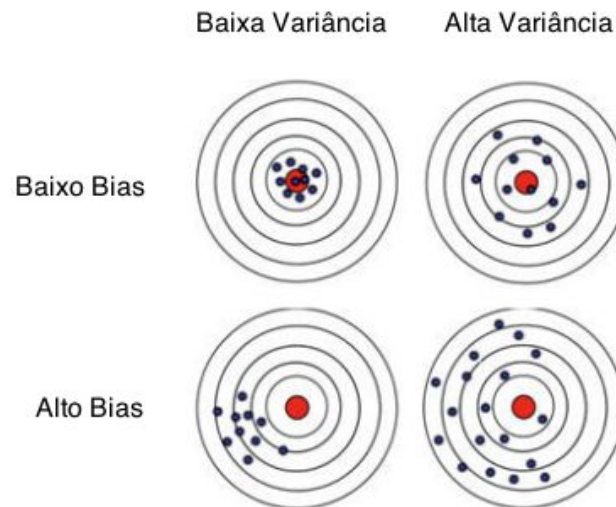
Combinando Classificadores

- Como a fusão com probabilidade se compara com o **voto majoritário**, supondo que a classe correta nesse caso é a classe 2?
- Porque a **média** se comportou diferentemente dos outros métodos baseado em fusão de probabilidades?

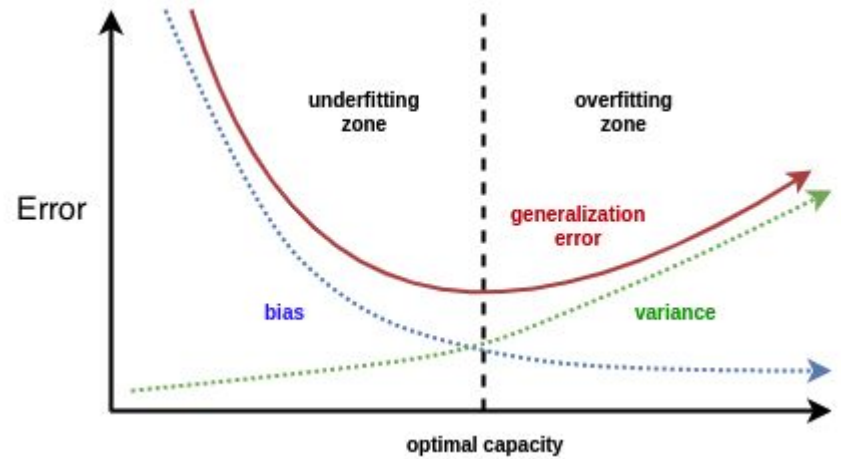
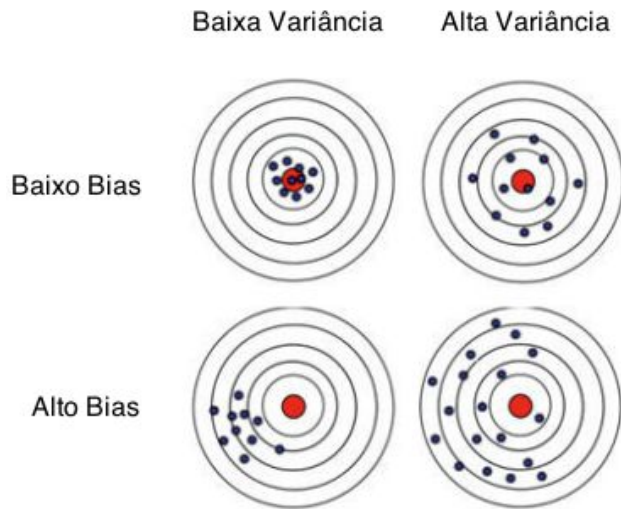
Classe	Probabilidade				Prod	Media	Min	Max	Soma
	c1	c2	c3	c4					
1	0.8	0.56	0.1	0.6	0.027	0.515	0.100	0.800	4.502
2	0.2	0.44	0.9	0.4	0.032	0.485	0.200	0.900	5.557

Ensembles

- Um conjunto de classificadores (geralmente do mesmo tipo) **gerados automaticamente** combinados de alguma forma.
- Resultados teóricos e práticos mostram que melhor desempenho do que um único classificador.
- Baseado na ideia de que o erro de um classificador pode ser decomposto em duas componentes:
 - **Bias**: Distância do resultado do classificador do seu objetivo
 - **Variância**: Quão dispersos são os resultados.



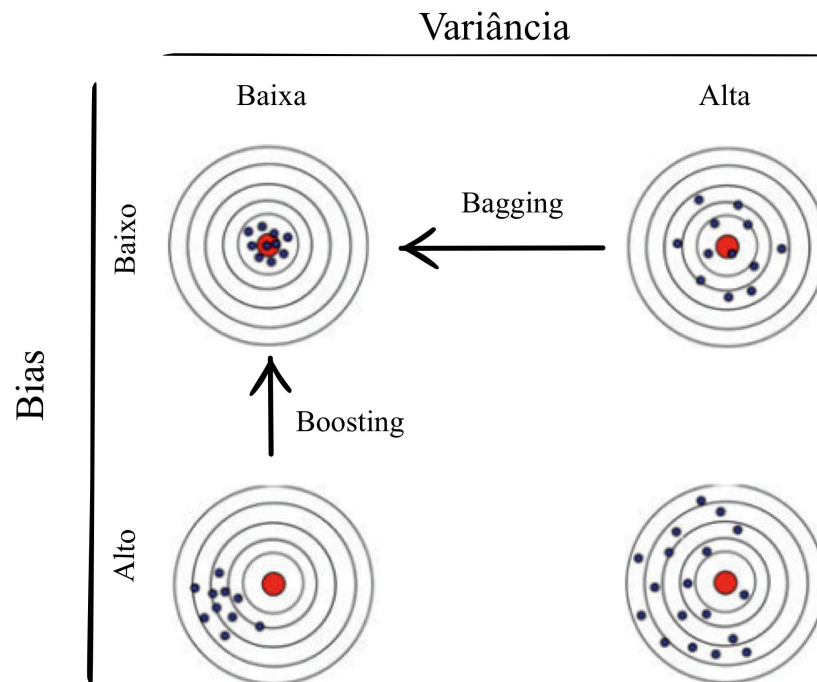
Ensembles



Ensembles

Bagging e Boosting são as duas principais técnicas de *ensemble* para lidar com os problemas de alta variância e alto bias.

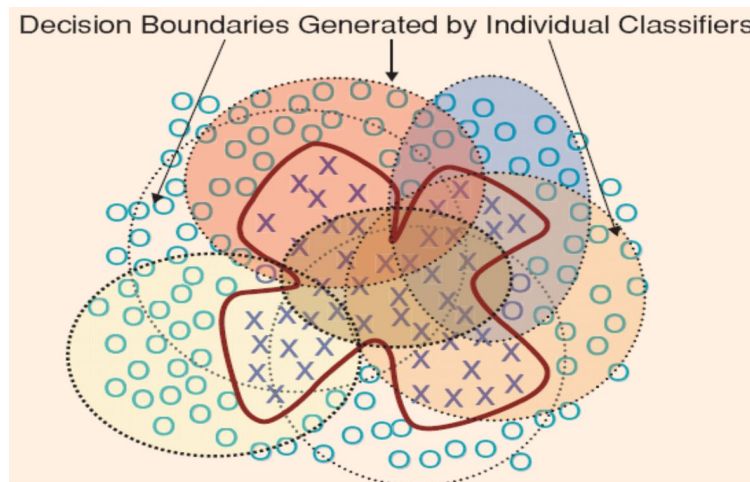
- **Bagging** é capaz de reduzir a **variância**.
- **Boosting** tem como objetivo reduzir o **bias**.



Ensembles

Bagging

- **Bagging** = *Bootstrap Aggregation*
 - Aprender diversos classificadores, utilizando para isso diferentes partições dos dados
 - Combinar o resultado dos modelos através da média
 - Média reduz a variância.



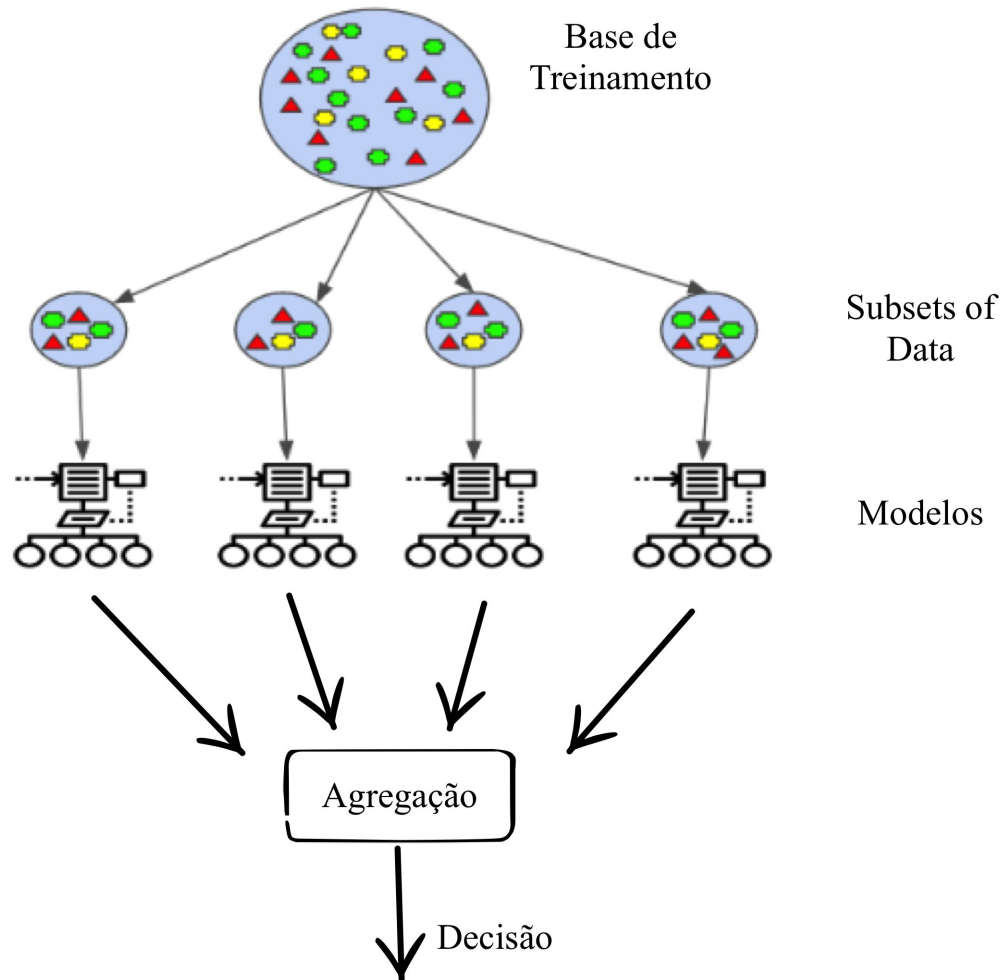
Ensembles

Bagging

- **Bootstrap**
 - Criar um conjunto aleatório de dados
 - Selecciona N' exemplos do conjunto N com **reposição**.
- **Bagging**
 - Repetir K vezes
 - Criar um conjunto de treinamento com $N' < N$ exemplos
 - Treinar um classificador com esse conjunto
- Para testar, execute cada classificador na base de teste
 - Utilize uma regra de fusão para agregar os resultados (voto, média)
 - No caso de regressão, a média é usada.

Ensembles

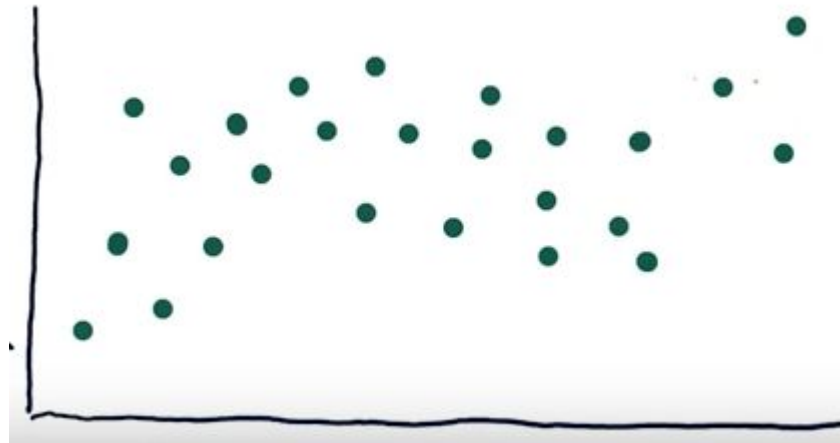
Bagging



Ensembles

Bagging - Exemplo (em regressão)

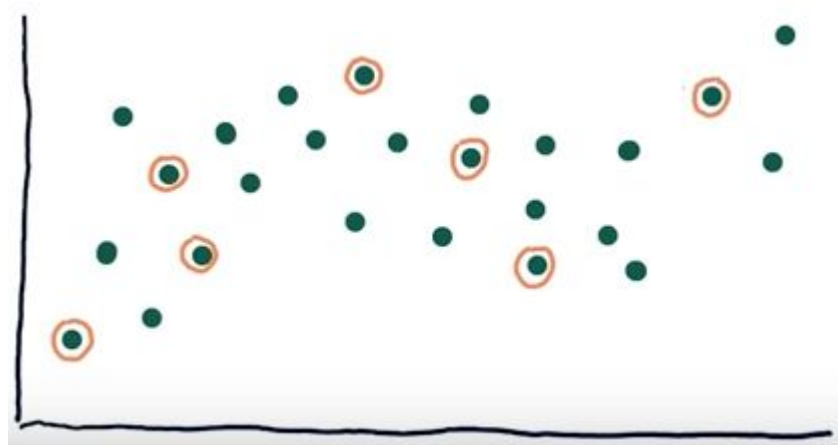
- Considere o conjunto de dados abaixo.



Ensembles

Bagging - Exemplo

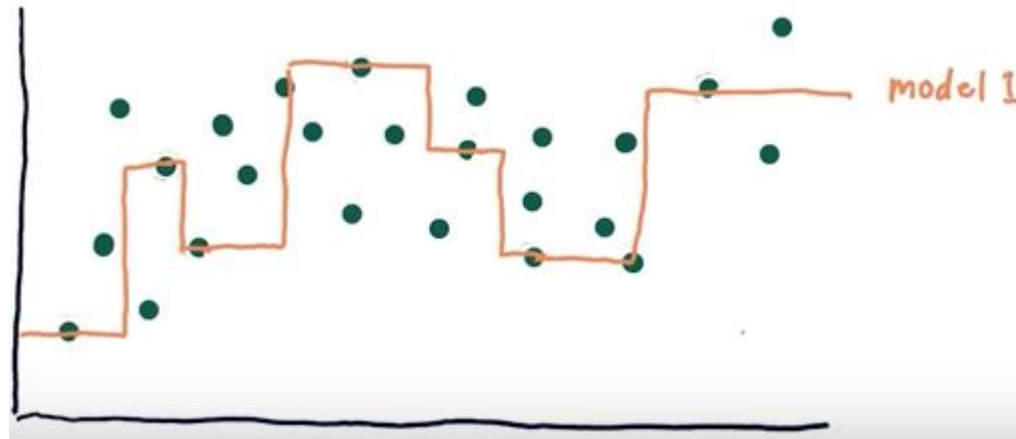
- Inicialmente selecionamos alguns dados aleatoriamente.



Ensembles

Bagging - Exemplo

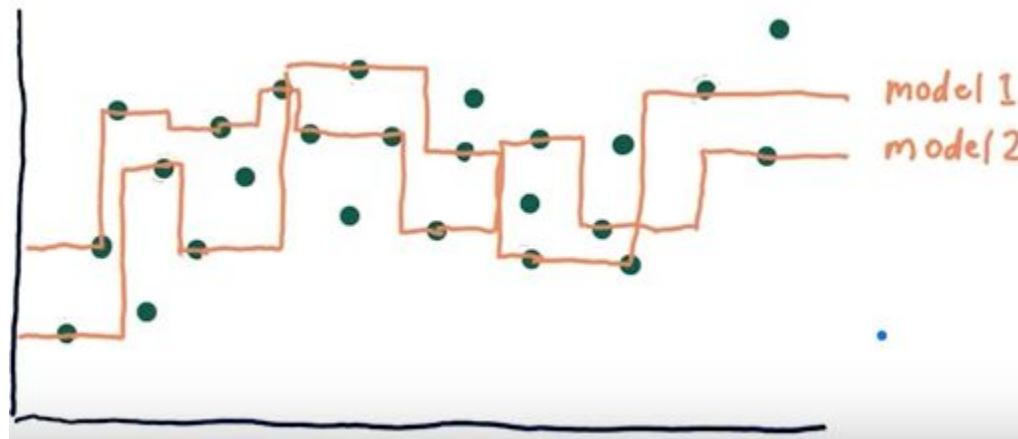
- Para esse conjunto de dados, construímos um classificador.
- Nesse exemplo, considere um k NN ($k = 1$) mostrado pela linha que passa sobre os pontos.
- Classificador com overfit.



Ensembles

Bagging - Exemplo

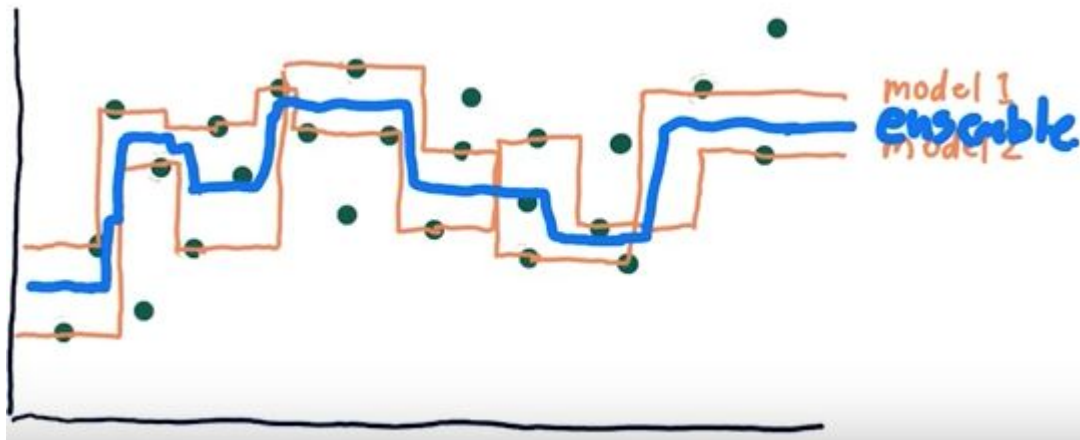
- Criando um segundo classificador com outro conjunto de dados.



Ensembles

Bagging - Exemplo

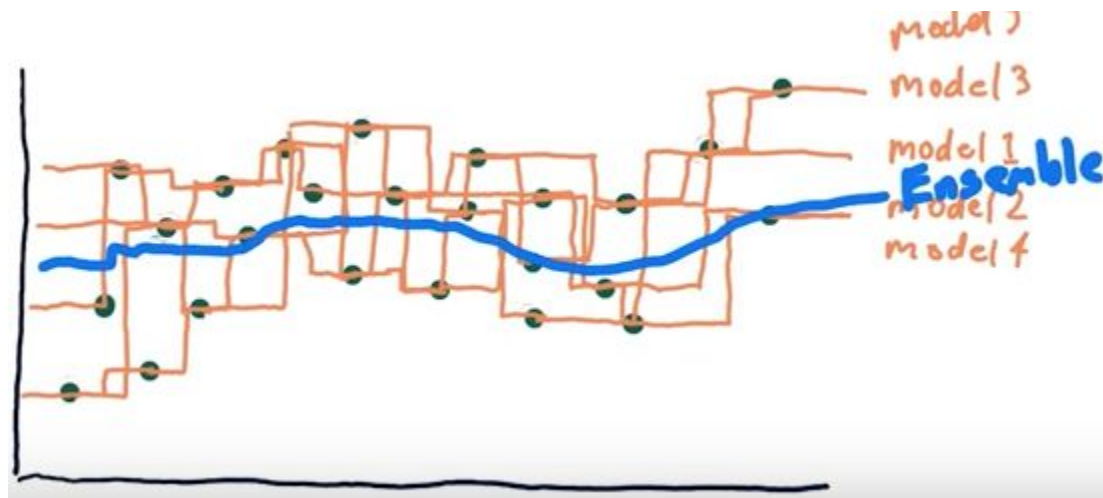
- Fazendo a média, temos um **ensemble**.
- O qual, como pode-se perceber, descreve melhor os dados.



Ensembles

Bagging - Exemplo

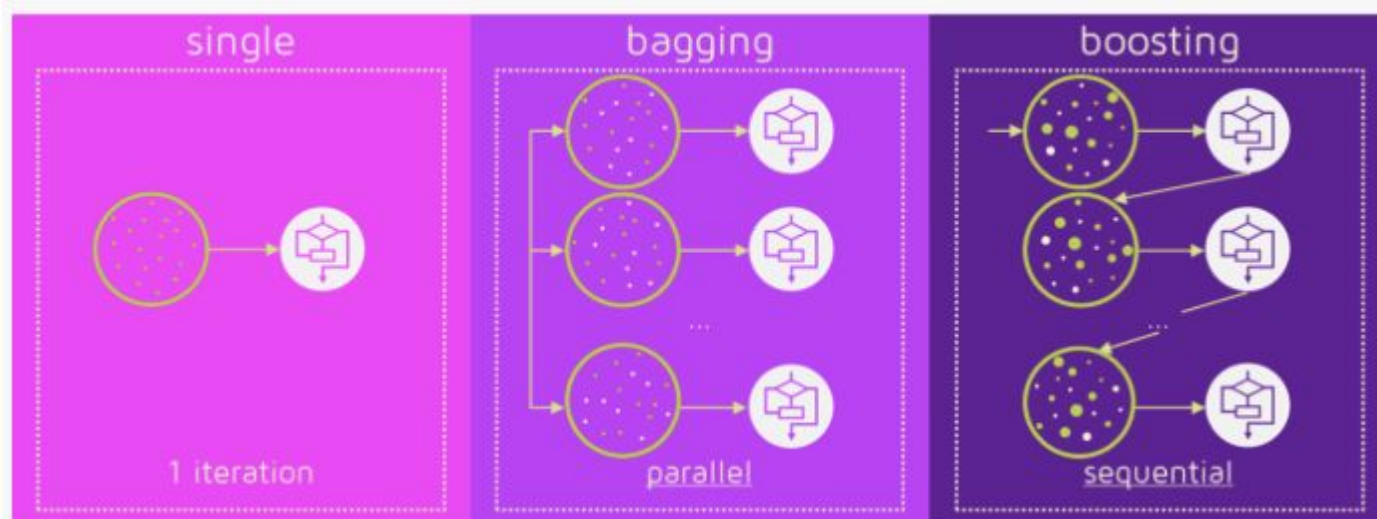
- Considerando mais classificadores, temos ainda um outro ensemble.



Ensembles

Boosting

- Treina uma sequência de classificadores (fracos) dando ênfase para exemplos **classificados erroneamente** pelo classificador anterior.
- Todos esses classificadores são então combinados, gerando assim um classificador robusto.
- Diferentemente do *bagging*, o *boosting* é sequencial.



Ensembles

Boosting

- O algoritmo original de *boosting* tem a desvantagem de precisar de um conjunto muito grande de aprendizagem.
- Em função disso, várias implementações do algoritmo de *boosting* têm sido desenvolvidas nos últimos anos.
- Uma implementação que teve bastante impacto na comunidade foi o **AdaBoost** (*Adaptive Boosting*)
- Mais recentemente, variantes como **Gradient Boosting** e **Extreme Gradient Boosting** (XGBoost) têm sido utilizadas com sucesso em diferentes tipos de problemas de classificação e regressão.

AdaBoost

- *AdaBoost* é um algoritmo que utiliza *Boosting* como método de aprendizagem
- A partir da combinação de vários classificadores com acurácia baixa (hipóteses fracas), uma regra de predição para um classificador com acurácia mais alta (hipótese forte) é produzida;
- Utiliza a mesma base de aprendizagem diversas vezes.
 - Por esse motivo a base não precisa ser muito grande.
- Classificadores devem ser fracos. Assim podemos garantir que eles não sofrem de *overfitting* (falta de generalização).

AdaBoost

Amostras / Instâncias

- Seja $\{ (x_i, y_i) \mid i = 1, 2, \dots, N \}$ um conjunto de N exemplos de treinamento, em que y_i é o rótulo da classe associado com a instância x_i
- $x_i \in \mathfrak{R}$ $y_i \in \{+1, -1\}$ ou (natural, plástico)



AdaBoost

Taxa de Erro e Importância do Classificador

- Dado um classificador aplicado ao *TrainSet*, é necessário estimar a taxa de erro da classificação realizada
- A **Taxa de Erro** de uma hipótese (classificador) h_f (fraco) é

$$\epsilon = PR(h_f(x_i) \neq y_i)$$

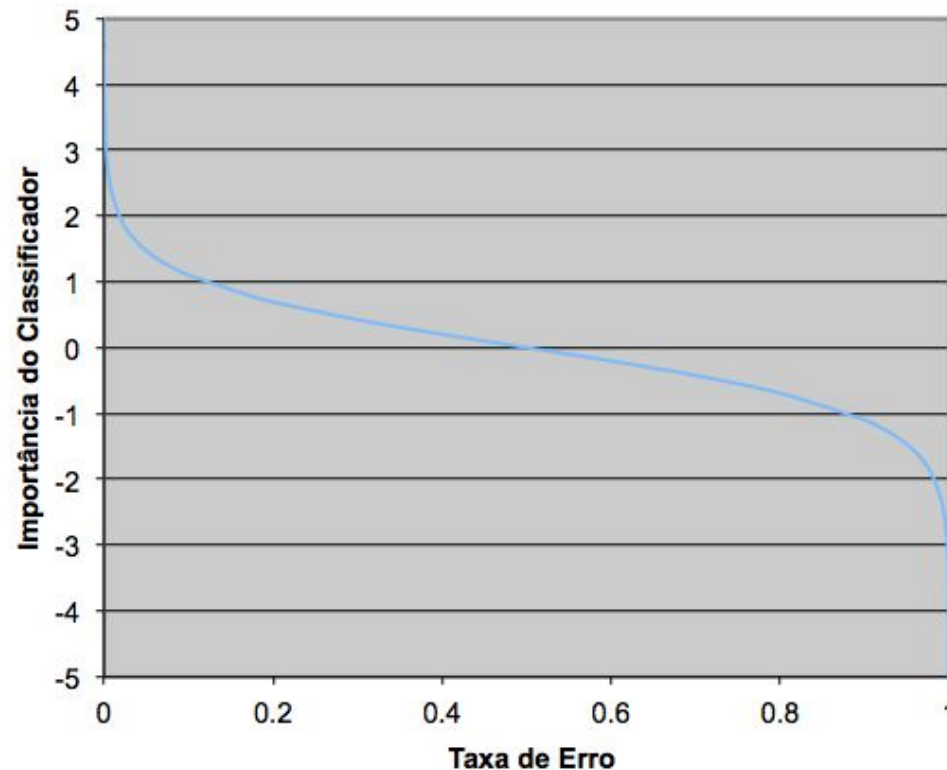
- A **Importância de um Classificador fraco** h_f , onde t é o índice da iteração, é

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1-e_t}{e_t} \right)$$

AdaBoost

Taxa de Erro e Importância do Classificador

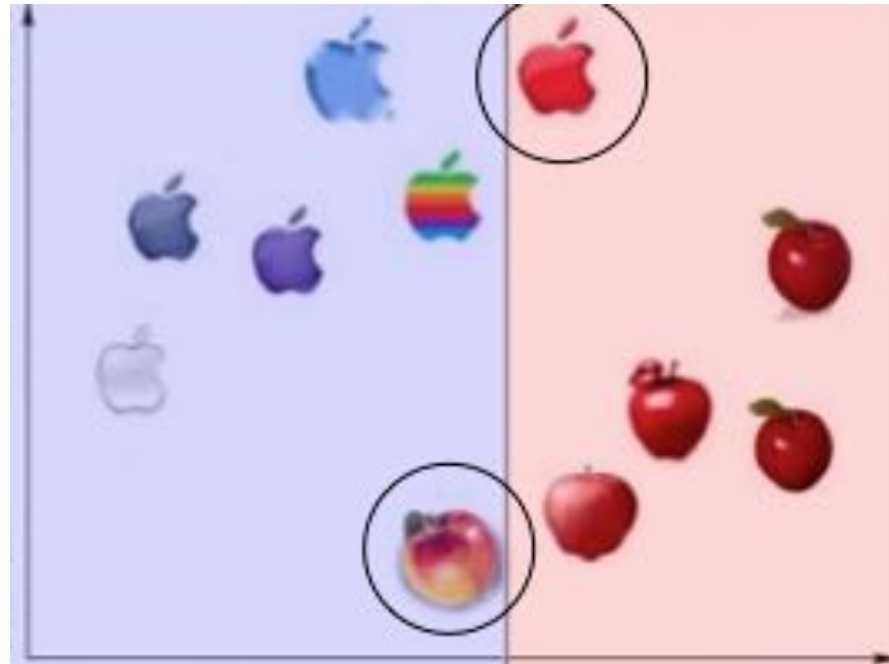
- Com isso é possível observar que quanto menor a taxa de erro, maior é a importância atribuída ao classificador.



AdaBoost

Cálculo de ϵ e α

- Erro (ϵ) = $2/11$ = 0,18
- Importância (α) = $\frac{1}{2} \ln\left(\frac{1-0,18}{0,18}\right) = 0,75$



AdaBoost

Pesos

- Usamos α_t para atualizar os pesos dos exemplos de treinamento.
- Considerando $\omega_i^{(t)}$ o peso atribuído ao exemplo (x_i, y_i) na iteração t , temos

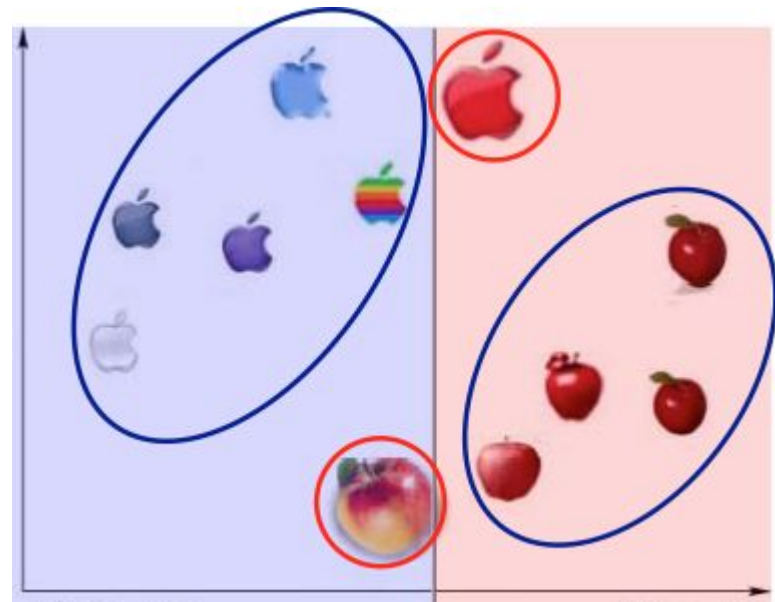
$$\omega_i^{(t+1)} = \frac{\omega_i^{(t)}}{Z_t} \times k \quad k = \begin{cases} \exp^{-\alpha_t} & \text{se } h_t(x_i) = y_i, \\ \exp^{\alpha_t} & \text{se } h_t(x_i) \neq y_i \end{cases}$$

- Em que Z_t é o fator de normalização usado para garantir que o somatório dos pesos seja 1.
- Desta forma,
 - O peso **diminui** para registros classificados corretamente
 - O peso **aumenta** para registros classificados erroneamente

AdaBoost

Exemplo do Cálculo dos Pesos ($t = 1$)

- $\omega_i^{(t=0)} = 1/11 = 0,091$
- Corretos ($h_t(x_i) = y_i$)
 - $\omega_i^{(t=1)} = 0,091 \times \exp^{-0.75}$
 - $0,091 \times 0,047 = \mathbf{0,043}$
- Incorretos ($h_t(x_i) \neq y_i$)
 - $\omega_i^{(t=1)} = 0,091 \times \exp^{0.75}$
 - $0,091 \times 2,117 = \mathbf{0,193}$



AdaBoost

Normalização

- Garantir que a soma seja 1
 - $(0,043 \times 9) + (0,193 \times 2) = 0,773$
 - Corretos: $0,043 / 0,773 = \mathbf{0,056}$
 - Incorretos: $0,193 / 0,773 = \mathbf{0,248}$
 - $(0,056 \times 9) + (0,248 \times 2) \cong 1,000$
- O resultado final é calculado com base nos resultados dos classificadores e seus respectivos pesos

$$H_f(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$$

AdaBoost

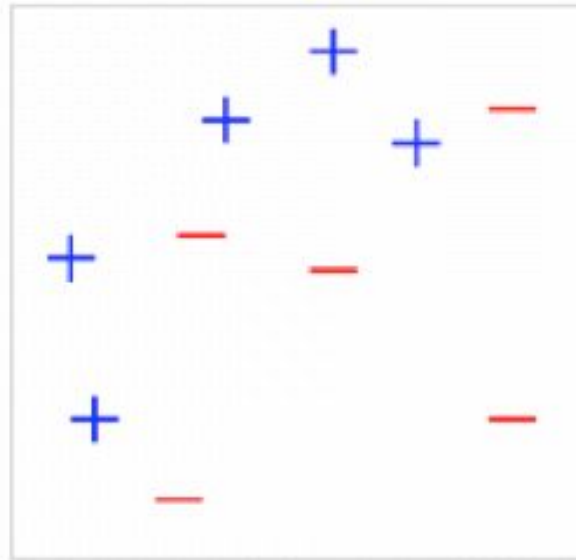
Algoritmo

1. Inicializar pesos para os N exemplos
2. Seja t o número de iterações
3. Para $i = 1..t$ faça
 - a. Crie *TrainSet* D_i pela amostragem de D de acordo com o peso
 - b. Treine um classificador base h_t utilizando D_i
 - c. Aplique $h_t \quad \forall$ exemplos em D
 - d. Calcule a taxa de erro de $h_t(\epsilon)$
 - e. Se a taxa de error for maior que **50%** então
 - i. Re-inicialize os pesos para os N exemplos
 - ii. Vá para o passo 3.a
 - f. Calcule a importância do classificador (α)
 - g. Calcule o novo peso para cada exemplo (w)
4. Construa o classificador final (**forte**) ponderando as importâncias (α) de cada classificador fraco.

AdaBoost

Exemplo - Inicialização

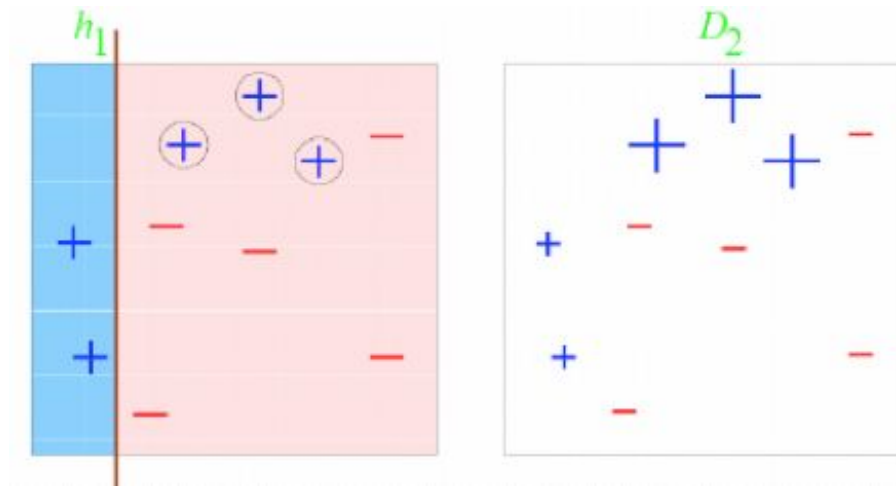
- Considere um *TrainSet* com 10 amostras de duas classes (+,-)
- Inicialmente todas as amostras de treinamento possuem o mesmo peso (1/10).



AdaBoost

Exemplo - Primeira Iteração

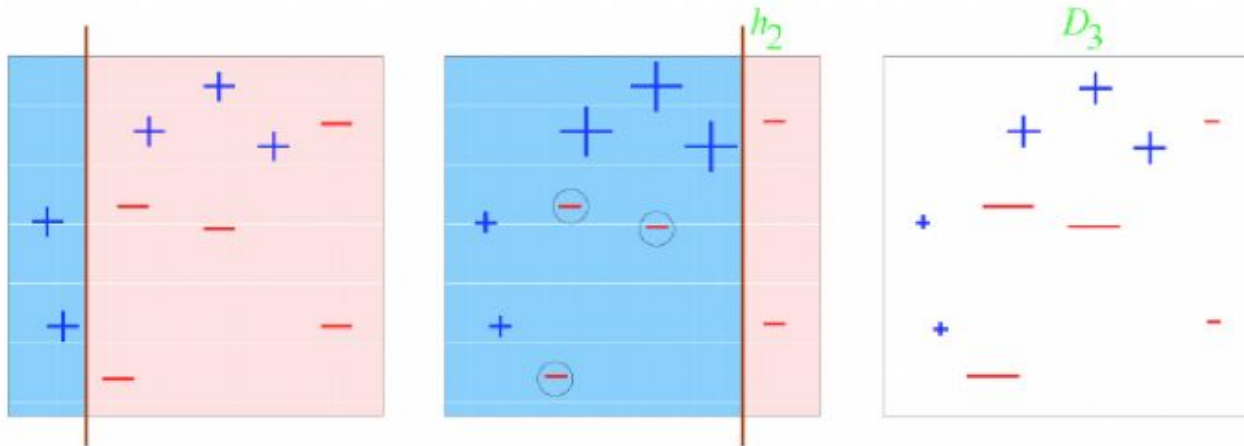
- Três amostras **positivas** são classificadas incorretamente pelo classificador h_1
- Os pesos para essas amostras são aumentados (*boosted*).
 - $\epsilon_1 = 0,30$ (30%) e $\alpha_1 = 0,42$ (importância)



AdaBoost

Exemplo - Segunda Iteração

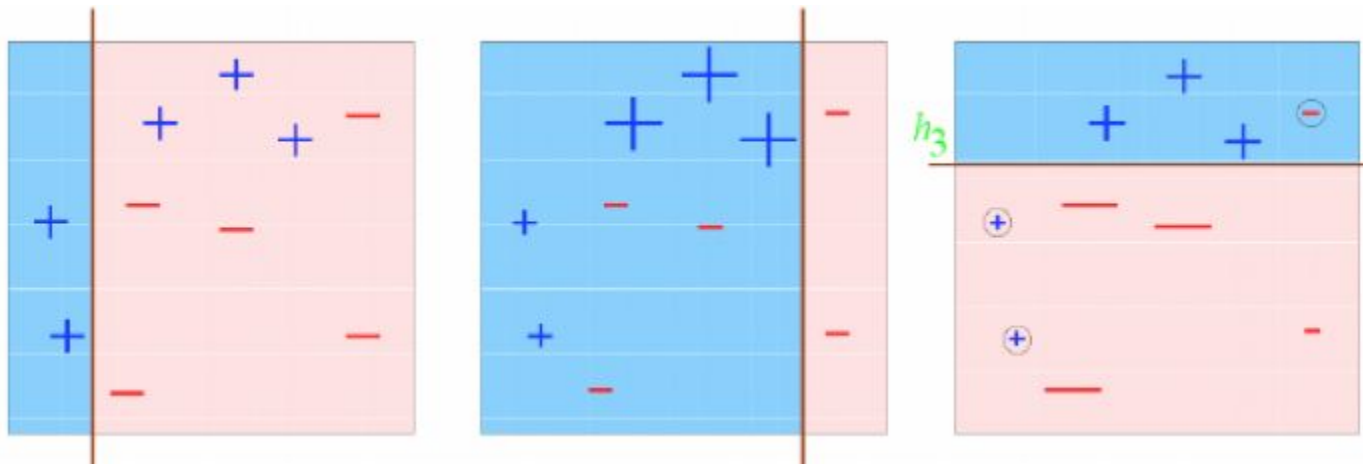
- Três amostras **negativas** são classificadas incorretamente pelo classificador h_2
- Os pesos para essas amostras são aumentados (*boosted*).
 - $\epsilon_2 = 0,21$ (21%) e $\alpha_2 = 0,62$ (importância)



AdaBoost

Exemplo - Terceira Iteração

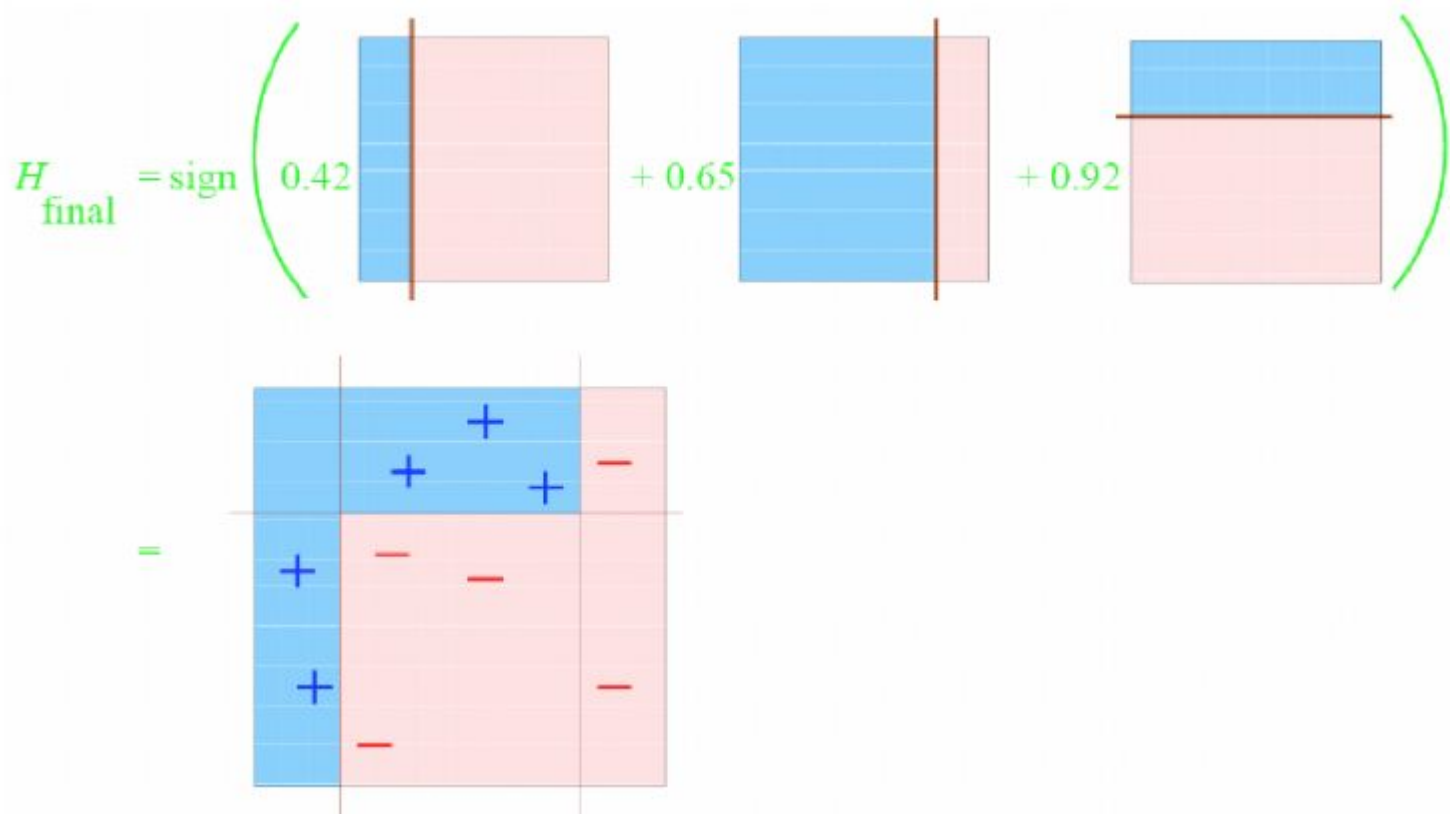
- Uma amostra **negativas** e três **positivas** são classificadas incorretamente pelo classificador h_3
- Os pesos para essas amostras são aumentados (*boosted*).
 - $\epsilon_2 = 0,14$ (14%) e $\alpha_3 = 0,92$ (importância)



AdaBoost

Exemplo - Classificador final

- Integra os três classificadores fracos e obtém um classificador final robusto



AdaBoost

Vantagens:

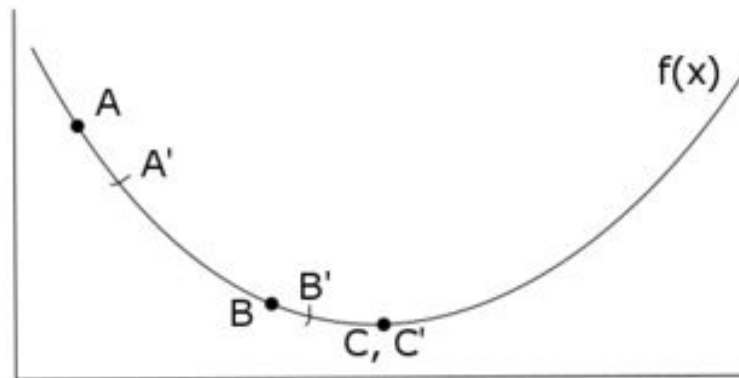
- Simples e fácil de implementar.
- Diferentes algoritmos de base podem ser usados como classificadores fracos.
- Como o conjunto de dados é sempre o mesmo, não existe a necessidade de uma grande base de treinamento.

Desvantagens:

- Depende do desempenho dos classificadores fracos. (**> 50%**)
- Desempenho é ruim se a quantidade de ruídos for alta (desempenho ruim dos classificadores fracos).

Gradient Boosting

- O desempenho do *Boosting* pode ser melhorado dando pesos para os classificadores.
- No caso do *Gradient Boosting*, os pesos são encontrados através do algoritmo *Gradient Descent*.
- O algoritmo de otimização é dividido em duas partes:
 - Determinar a direção do gradiente
 - Tamanho do passo
- **XGBoost** segue exatamente a mesma ideia, mas foi o processo de otimização da descida do gradiente que foi modificado para ganhar em desempenho e também evitar *overfitting*.



Random Subspace Method (RSM)

- RSM é bastante parecido com *Bagging*, porém as características (**atributos**) são selecionados aleatoriamente com reposição.
- Esse processo é também conhecido como “*Feature Bagging*”.
- Isso evita que características fortes (alto poder de discriminação) sejam sempre utilizadas.
- Gera maior **diversidade**.
- RSM utilizado com **árvores de decisão** é conhecido como
 - Florestas Aleatórias (**Random Forests**)

Random Forests

- Árvores muito **profundas** tendem a ter baixo poder de generalização (baixo **bias** e alta **variância**).
- *Random Forests* é a maneira de se fazer a média de diversas árvores treinadas em **diferentes partições** da mesma base.
- Nesse caso existe um pequeno **aumento no bias**
 - um pouco de perda de interpretabilidade das decisões (ponto forte da árvore de decisão)



Referências

