Démocratie à géométrie variable
(à l'usage d'algorithmes)

Théo Delemazure, François Durand & Fabien Mathieu

ALGOTEL 2021 - La Rochelle

Des algorithmes qui votent ?

Les élections sont partout

	C#1	C#2	C#3	C#4
E#1				
E#2				
E#3				
E#4				
E#5				
E#6				
Total				

Election humaine typique

- Des candidats et des électeurs

Les élections sont partout

	C#1	C#2	C#3	C#4
E#1	1			
E#2		1		
E#3	1			
E#4			1	
E#5			1	
E#6	1			
Total				

Election humaine typique

- Des candidats et des électeurs
- Les électeurs votent anonymement

Les élections sont partout

	C#1	C#2	C#3	C#4
E#1	1	0	0	0
E#2	0	1	0	0
E#3	1	0	0	0
E#4	0	0	1	0
E#5	0	0	1	0
E#6	1	0	0	0
Total				

Election humaine typique

- Des candidats et des électeurs
- Les électeurs votent anonymement

Les élections sont partout

	C#1	C#2	C#3	C#4
E#1	1	0	0	0
E#2	0	1	0	0
E#3	1	0	0	0
E#4	0	0	1	0
E#5	0	0	1	0
E#6	1	0	0	0
Total	3	1	2	0

Election humaine typique

- Des candidats et des électeurs
- Les électeurs votent anonymement
- On compte les votes

Les élections sont partout

	C#1	C#2	C#3	C#4
E#1	1	0	0	0
E#2	0	1	0	0
E#3	1	0	0	0
E#4	0	0	1	0
E#5	0	0	1	0
E#6	1	0	0	0
Total	3	1	2	0

Election humaine typique

- Des candidats et des électeurs
- Les électeurs votent anonymement
- On compte les votes
- Le meilleur score l'emporte

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne				
Belgique				
Suède				
Serbie				
Macédoine				
Croatie				
Total				

Eurovision

- Des candidats et des électeurs

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
Total				

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
Total	36	44	37	32

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
Total	36	44	37	32

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
Total	36	44	37	32

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Comment faire gagner la France à l'Eurovision ?

Les élections sont partout

	“5”	“8”	“6”	“4”
Moyenne				

Ensemble Learning

- Différentes catégories

Les élections sont partout

	“5”	“8”	“6”	“4”
Moyenne				

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés

Les élections sont partout

	“5”	“8”	“6”	“4”
CNN (SGD, ƛ = 0.01)	23	**58**	18	1
CNN (SGD, ƛ = 0.1)	20	**64**	12	4
CNN (Adam, ƛ = 0.01)	26	**63**	11	0
CNN (Adam, ƛ = 0.1)	31	**55**	14	0
Moyenne				

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés

Les élections sont partout

	“5”	“8”	“6”	“4”
CNN (SGD, ƛ = 0.01)	23	58	18	1
CNN (SGD, ƛ = 0.1)	20	64	12	4
CNN (Adam, ƛ = 0.01)	26	63	11	0
CNN (Adam, ƛ = 0.1)	31	55	14	0
VGG (α = 0.4)	**71**	18	11	0
VGG (α = 0.05)	**78**	16	5	1
Moyenne				

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés

Les élections sont partout

	“5”	“8”	“6”	“4”
CNN (SGD, ƛ = 0.01)	23	58	18	1
CNN (SGD, ƛ = 0.1)	20	64	12	4
CNN (Adam, ƛ = 0.01)	26	63	11	0
CNN (Adam, ƛ = 0.1)	31	55	14	0
VGG (α = 0.4)	71	18	11	0
VGG (α = 0.05)	78	16	5	1
monHeuristique.py	15	15	**60**	10
Moyenne				

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés

Les élections sont partout

	“5”	“8”	“6”	“4”
CNN (SGD, ƛ = 0.01)	23	58	18	1
CNN (SGD, ƛ = 0.1)	20	64	12	4
CNN (Adam, ƛ = 0.01)	26	63	11	0
CNN (Adam, ƛ = 0.1)	31	55	14	0
VGG (α = 0.4)	71	18	11	0
VGG (α = 0.05)	78	16	5	1
monHeuristique.py	15	15	60	10
Moyenne	38	**41**	19	2

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés
- Comment agréger les scores ?

Les élections sont partout

	“5”	“8”	“6”	“4”
CNN (SGD, ƛ = 0.01)	23	58	18	1
CNN (SGD, ƛ = 0.1)	20	64	12	4
CNN (Adam, ƛ = 0.01)	26	63	11	0
CNN (Adam, ƛ = 0.1)	31	55	14	0
VGG (α = 0.4)	71	18	11	0
VGG (α = 0.05)	78	16	5	1
monHeuristique.py	15	15	60	10
Moyenne des groupes	**38**	31	27	4

Ensemble Learning

- Différentes catégories
- Algorithmes donnent des scores
- Certains algos sont liés
- Comment agréger les scores ?

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
Total (groupé)				

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Comment faire gagner la France à l'Eurovision ?

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	7	7	5	10
Macédoine				
Croatie				
Total (groupé)				

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Comment faire gagner la France à l'Eurovision ?

Les élections sont partout

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	7	7	5	10
Macédoine				
Croatie				
Total (groupé)	23	31	28	22

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd toujours

Comment faire gagner la France à l'Eurovision ?

Agrégation d'algorithmes

Etat de l'*Ensemble Learning*

- Techniques de ML avancées
- Techniques de votes basiques
- Nécessite une **diversité** des algorithmes

Agrégation d'algorithmes

Etat de l'*Ensemble Learning*

- Techniques de ML avancées
- Techniques de votes basiques
- Nécessite une **diversité** des algorithmes

Notre objectif

- S'adapter à toutes les situations
- Exploiter la diversité
- Prendre en compte les corrélations

Agrégation d'algorithmes

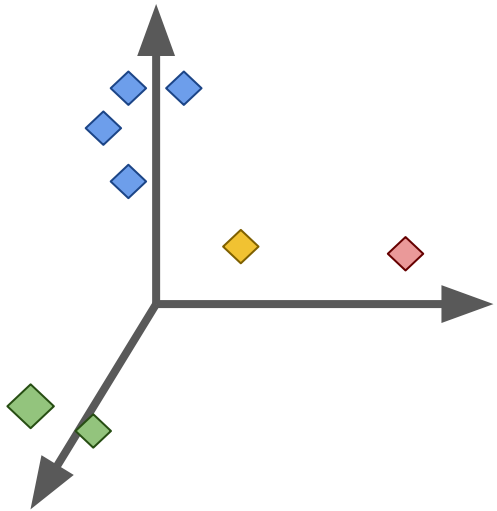
Etat de l'*Ensemble Learning*

- Techniques de ML avancées
- Techniques de votes basiques
- Nécessite une **diversité** des algorithmes

Notre objectif

- S'adapter à toutes les situations
- Exploiter la diversité
- Prendre en compte les corrélations

Notre approche : Représentation spatiale des algorithmes



Agrégation d'algorithmes

Etat de l'*Ensemble Learning*

- Techniques de ML avancées
- Techniques de votes basiques
- Nécessite une **diversité** des algorithmes

Notre objectif

- S'adapter à toutes les situations
- Exploiter la diversité
- Prendre en compte les corrélations

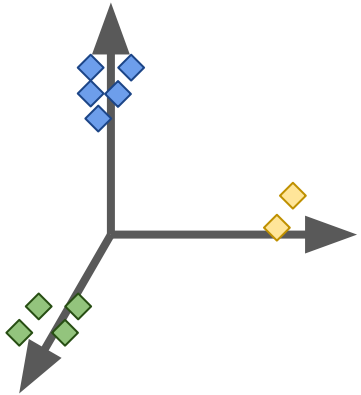
Notre approche : Représentation spatiale des algorithmes

Avertissement

Nous ne cherchons pas à déterminer le meilleur algorithme

Notre méthode : *SVD-Nash*

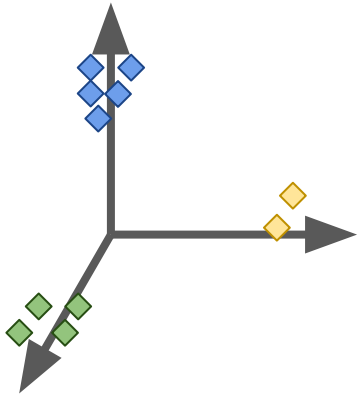
Cas idéal



p groupes d'algorithmes

- Les algorithmes **du même groupe** sont très similaires
- Les algorithmes **de différents groupes** sont indépendants
- Les groupes sont de **tailles très variables**

Cas idéal

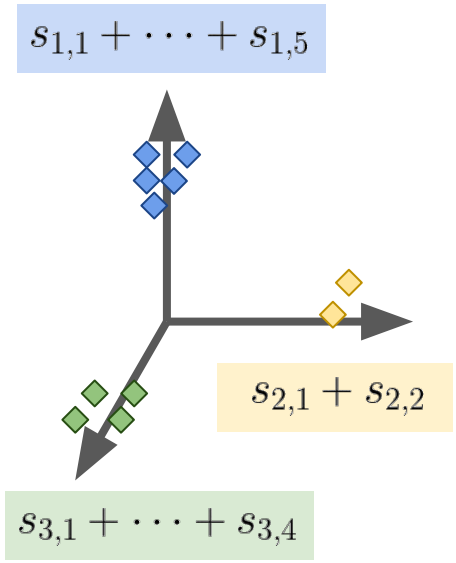


p groupes d'algorithmes

- Les algorithmes **du même groupe** sont très similaires
- Les algorithmes **de différents groupes** sont indépendants
- Les groupes sont de **tailles très variables**

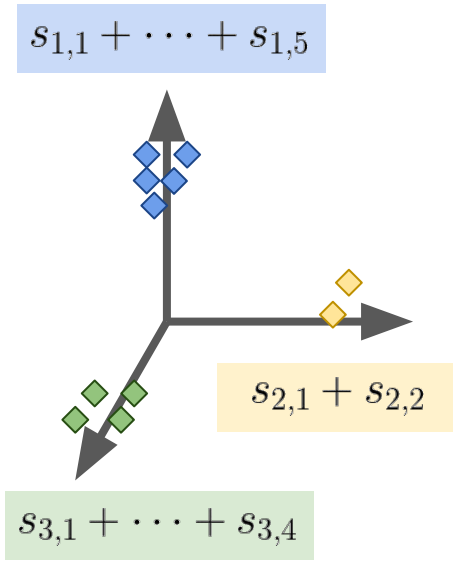
Objectif : Agréger les scores en ignorant la taille des groupes

Cas idéal



Objectif : Agréger les scores en ignorant la taille des groupes

Cas idéal



Objectif : Agréger les scores en ignorant la taille des groupes

Vote classique (Range Voting)

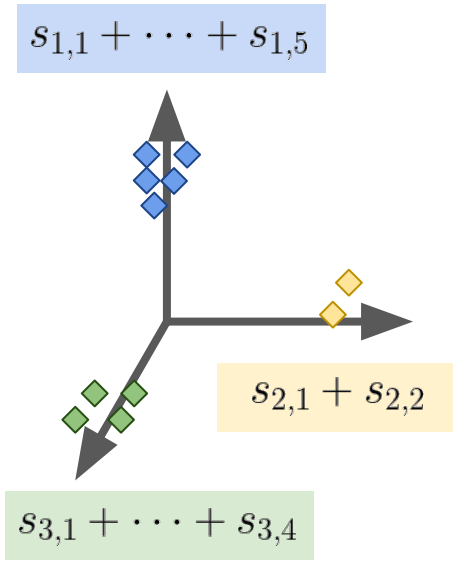
$$score(c) = \sum_{i=1}^{p} \sum_{j=1}^{|G_i|} s_{i,j}(c)$$

$s_{1,1} + \cdots + s_{1,5}$

\+ $s_{2,1} + s_{2,2}$

\+ $s_{3,1} + \cdots + s_{3,4}$

Cas idéal



Objectif : Agréger les scores en ignorant la taille des groupes

Vote classique (Range Voting)

$$score(c) = \sum_{i=1}^{p} \sum_{j=1}^{|G_i|} s_{i,j}(c)$$

$s_{1,1} + \cdots + s_{1,5}$

\+ $s_{2,1} + s_{2,2}$

\+ $s_{3,1} + \cdots + s_{3,4}$

Nash Product

$$score(c) = \prod_{i=1}^{p} \sum_{j=1}^{|G_i|} s_{i,j}(c)$$

$s_{1,1} + \cdots + s_{1,5}$

x $s_{2,1} + s_{2,2}$

x $s_{3,1} + \cdots + s_{3,4}$

La *matrice de score*

$$M = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

La Recette

- Une matrice d'*embeddings* (une colonne = un groupe)

Retrouver le *Nash Product*

$$M(c) = \begin{pmatrix} \sqrt{s_{1,1(c)}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{s_{1,|G_1|}(c)} & 0 & \cdots & 0 \\ 0 & \sqrt{s_{2,1}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \sqrt{s_{2,|G_2|}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,1}(c)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,|G_p|}(c)} \end{pmatrix}$$

La Recette

- Une matrice d'*embeddings* (une colonne = un groupe)
- On multiplie par la racine des scores

Retrouver le *Nash Product*

$$M(c) = \begin{pmatrix} \sqrt{s_{1,1(c)}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{s_{1,|G_1|}(c)} & 0 & \cdots & 0 \\ 0 & \sqrt{s_{2,1}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \sqrt{s_{2,|G_2|}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,1}(c)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,|G_p|}(c)} \end{pmatrix}$$

La Recette

- Une matrice d'*embeddings* (une colonne = un groupe)
- On multiplie par la racine des scores
- On calcule par **SVD** les valeurs singulières $\lambda_1, \ldots, \lambda_p$

Retrouver le *Nash Product*

$$M(c) = \begin{pmatrix} \sqrt{s_{1,1(c)}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{s_{1,|G_1|}(c)} & 0 & \cdots & 0 \\ 0 & \sqrt{s_{2,1}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \sqrt{s_{2,|G_2|}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,1}(c)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,|G_p|}(c)} \end{pmatrix}$$

La Recette

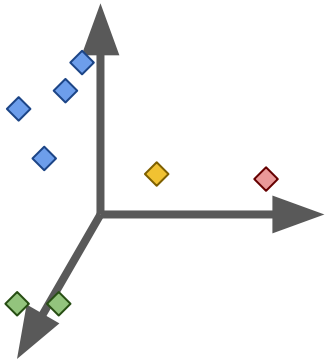
- Une matrice d'*embeddings* (une colonne = un groupe)
- On multiplie par la racine des scores
- On calcule par **SVD** les valeurs singulières $\lambda_1, \ldots, \lambda_p$
- Abracadabra : $\lambda_i = \sum_{j=1}^{|G_i|} s_{i,j}$

Retrouver le *Nash Product*

$$M(c) = \begin{pmatrix} \sqrt{s_{1,1(c)}} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{s_{1,|G_1|}(c)} & 0 & \cdots & 0 \\ 0 & \sqrt{s_{2,1}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \sqrt{s_{2,|G_2|}(c)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,1}(c)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{s_{p,|G_p|}(c)} \end{pmatrix}$$

La Recette

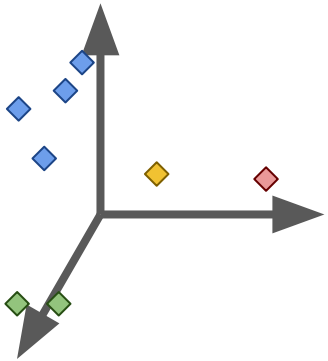
- Une matrice d'*embeddings* (une colonne = un groupe)
- On multiplie par la racine des scores
- On calcule par **SVD** les valeurs singulières $\lambda_1, \ldots, \lambda_p$
- Abracadabra : $\lambda_i = \sum_{j=1}^{|G_i|} s_{i,j}$
- Moyenne : $\sum_i \lambda_i$
- Nash Product : $\prod_i \lambda_i$



Et quand le cas n'est pas idéal ?

Les *embeddings* peuvent être plus complexes

- Cas idéal : $\vec{v_{i,j}} = \delta_i = (0, \ldots, 1, \ldots, 0)$
- On veut une méthode qui fonctionne pour n'importe quels *embeddings* $\vec{v_1}, \ldots, \vec{v_n}$
- Par exemple $\vec{v_i} = \alpha\delta_1 + \beta\delta_2 = (\alpha, \beta, 0, \ldots, 0)$

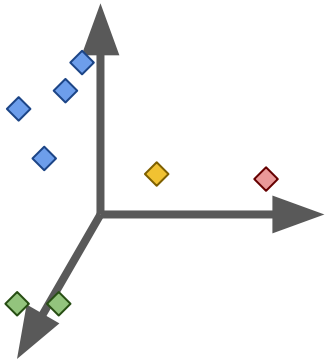


Et quand le cas n'est pas idéal ?

Les *embeddings* peuvent être plus complexes

Pourquoi ?

- On dispose rarement de *groupes* pour séparer les algorithmes
- Même au sein des groupes, il y a de la variabilité



Et quand le cas n'est pas idéal ?

Les *embeddings* peuvent être plus complexes

Pourquoi ?

- On dispose rarement de *groupes* pour séparer les algorithmes
- Même au sein des groupes, il y a de la variabilité

Comment ?

- On récupère les scores donnés sur **un set d'entraînement**
- On réduit la matrice à l'aide d'une **PCA**
- Et paf, ça fait des *embeddings*

Retrouver le *Nash Product*

$$M = \begin{pmatrix} \vec{v_1} \\ \vdots \\ \vec{v_n} \end{pmatrix}$$

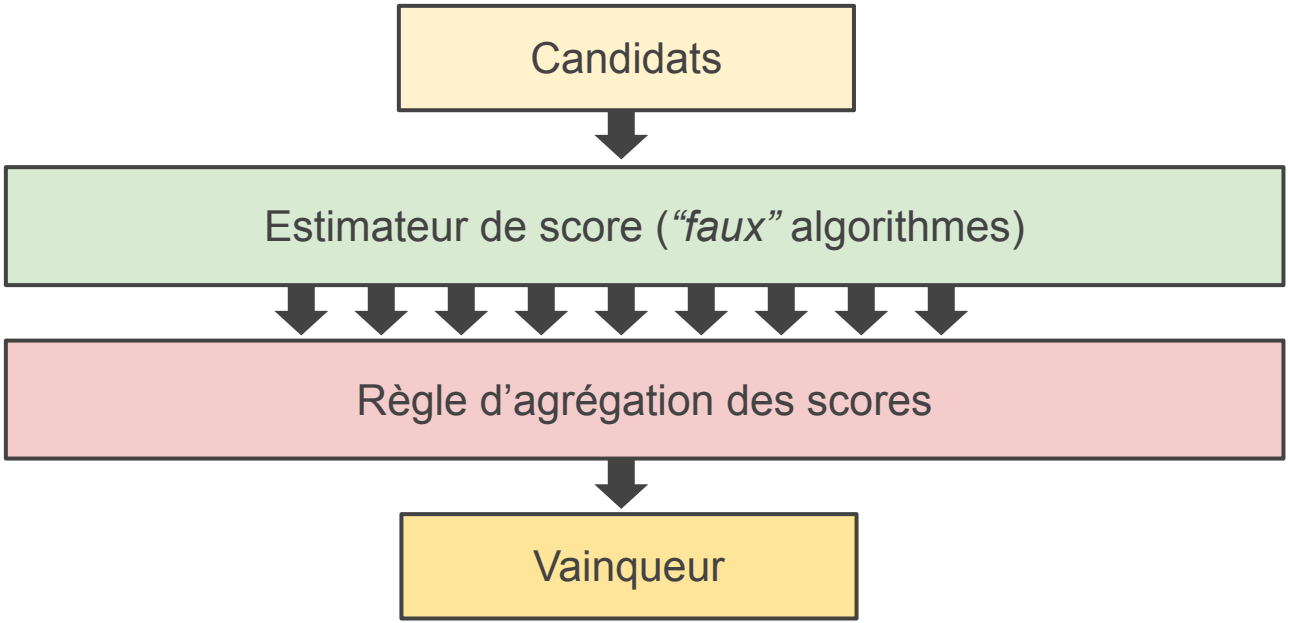
$$M(c) = \begin{pmatrix} \sqrt{s_1(c)}\vec{v_1} \\ \vdots \\ \sqrt{s_n(c)}\vec{v_n} \end{pmatrix}$$

La Recette

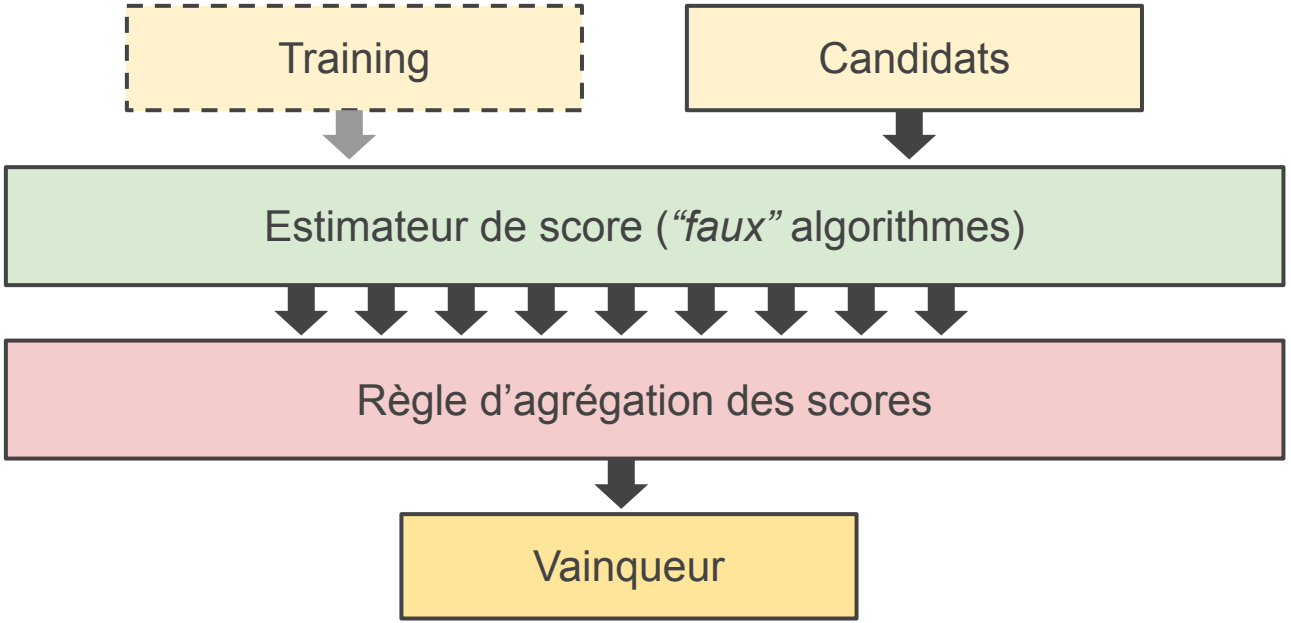
- Une matrice d'*embeddings* (une colonne = un groupe)
- On multiplie par la racine des scores
- On calcule par **SVD** les valeurs singulières $\lambda_1, \ldots, \lambda_p$
- Abracadabra : $\lambda_i = \sum_{j=1}^{|G_i|} s_{i,j}$
- Moyenne : $\sum_i \lambda_i$
- Nash Product : $\prod_i \lambda_i$

Validation expérimentale

Validation expérimentale



Validation expérimentale



Setting

- 20 candidats avec des **scores véritables** allant de 10 à 20
- 3 groupes d'algorithmes:
  - **G1** : 30 algorithmes
  - **G2** : 2 algorithmes
  - **G3 = 0.3G1 + 0.7G2** : 5 algorithmes

Estimation = score véritable + bruit de groupe + bruit individuel

$s(c)$

Setting

- 20 candidats avec des **scores véritables** allant de 10 à 20
- 3 groupes d'algorithmes:
  - **G1** : 30 algorithmes
  - **G2** : 2 algorithmes
  - **G3 = 0.3G1 + 0.7G2** : 5 algorithmes

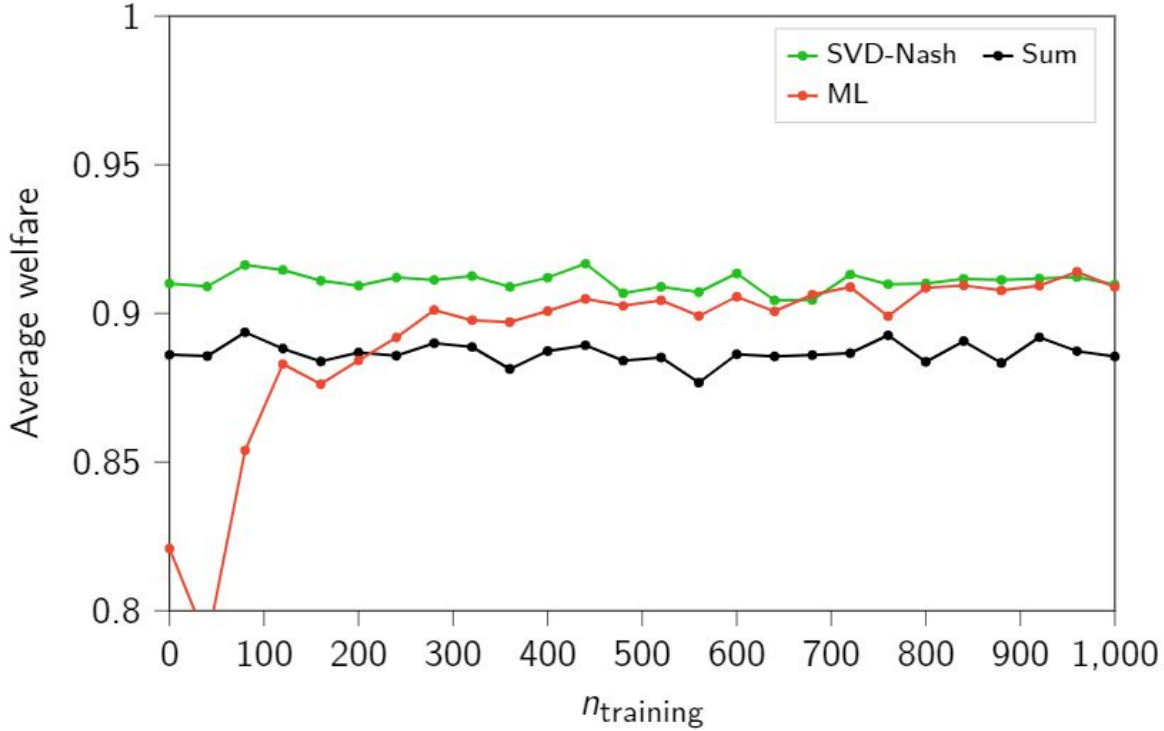
Objectif : Maximiser le *welfare*

$$w(c) = \frac{s(c) - s_{min}}{s_{\max} - s_{min}}$$

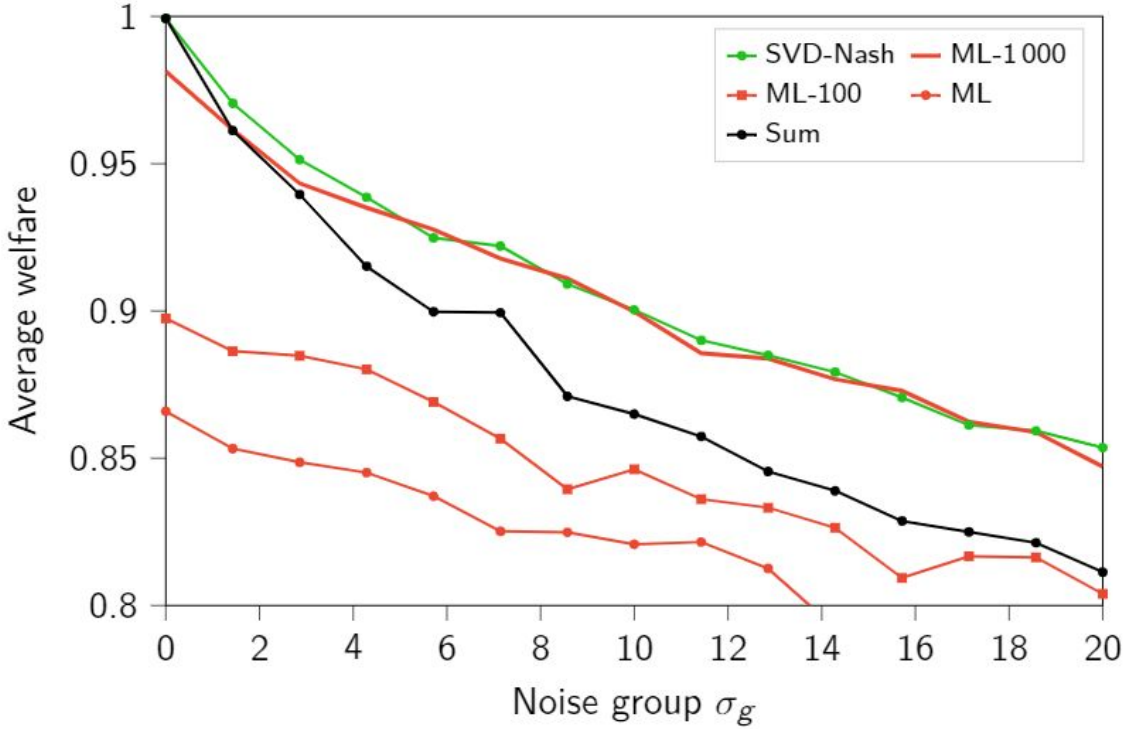
Concurrent : *Maximum Likelihood*

- Hypothèse : Bruit gaussien
- Méthode : Utilise l'entraînement pour construire la covariance entre algos

Résultats


SVD-Nash
Sum
ML
Average welfare
1
0.95
0.9
0.85
0.8
0
100
200
300
400
500
600
700
800
900
1,000
n_{training}

Résultats


SVD-Nash
ML-1 000
ML-100
ML
Sum
Average welfare
1
0.95
0.9
0.85
0.8
0
2
4
6
8
10
12
14
16
18
20
Noise group σ_g

Résultats : retour à l'Eurovision

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
SVD-Nash				

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Comment faire gagner la France à l'Eurovision ?

Résultats : retour à l'Eurovision

	France	Italie	Suisse	Chypre
Allemagne	10	8	12	4
Belgique	0	12	6	6
Suède	6	4	5	2
Serbie	8	10	5	12
Macédoine	7	4	6	8
Croatie	5	6	3	10
SVD-Nash	7	31	28	25

Eurovision

- Des candidats et des électeurs
- Chaque groupe d'électeurs donne un certain nombre de points
- On somme les points
- La France perd

Comment faire gagner la France à l'Eurovision ?

Conclusion

Quand ?

- Besoin d'agréger des algorithmes
- Pas d'a-priori sur les liens entre algorithmes

Comment ?

- Théorie du vote
- Algèbre linéaire

Pourquoi ?

- Pas de *"meilleur"* algorithme
- Techniques d'*Ensemble learning* sensibles à la diversité

http://embedded-voting.readthedocs.io

Merci de votre attention !