Chapitre : Recherche d'information et apprentissage :

Apprentissage automatique de fonctions d'ordonnancement

(Learning To rank)

Les modèles de RI conventionnels

- Dépendants de la requête (Query-dependant)
  - Boolean model, extended Boolean model
  - Vector space model, latent semantic indexing (LSI), etc.
  - BM25 model, statistical language model, etc.

- Indépendants de la requête (Query-independent)
  - PageRank, TrustRank, BrowseRank, Toolbar Clicks, readability, document length, etc.

Critères de tri des modèles conventionnels

- Plusieurs critères
  - Dépendants de la requête
    - Tf, idf, BM25, LM, terme de la requête dans le titre, dans l'ancre, en grans, …
  - Indépendants de la requête
    - *Document length, PgeRank, number of unique Word, document trust, ...*
- On peut en avoir quelques centaines
  - (voire un millier) comment les combiner?

Critères de tri des modèles conventionnels

- On peut en avoir quelques centaines
  - (voire un millier) comment les combiner?
  - Arbitrary useful features – not a single unified model
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains “~”?
  - Page edit recency?
  - Page loading speed

ECIR’2010, Amit Singhal quoted that Google was using over 200 such features (“signals”)

L'Apprentissage automatique peut aider

- Apprentissage automatique propose des outils (techniques) permettant
  - D'ajuster automatiquement des paramètres
  - De combiner des centaines de paramètres

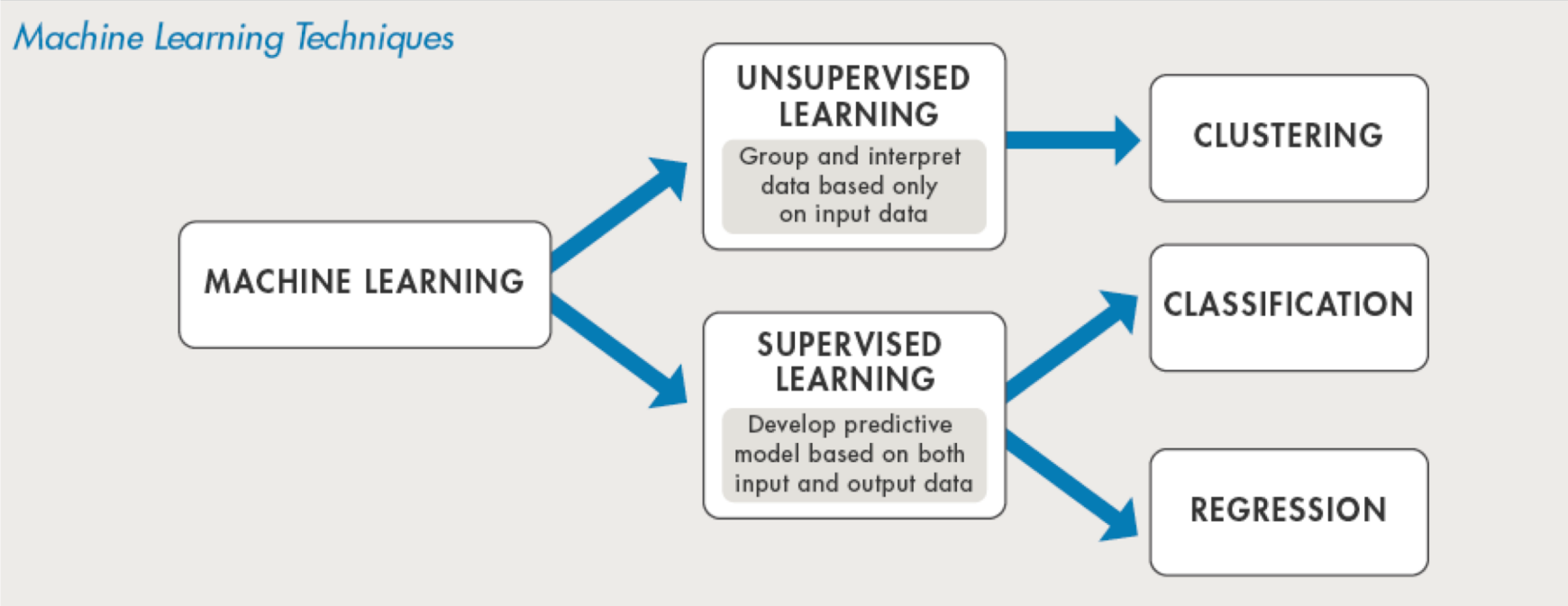
- Apprentissage de l'ordonnancement ("Learning to Rank")
  - Exploiter les techniques d'apprentissage pour répondre au problème de tri en RI
  - → Learning to rank.

Brève introduction l'apprentissage automatique

Apprentissage automatique ?

- L'**apprentissage automatique** (en anglais *machine learning*, littéralement « l'apprentissage machine ») ensemble d'approches permettant la conception de techniques capables d'apprendre à) partir de données
- Deux classes d'apprentissage :
  - Apprentissage supervisé (Supervised Learning)
    - On dispose d'exemples connus (données d'entrées, données sorties)
    - Objectif : apprendre une fonction permettant de mapper des données d'entrée vers les données de
    - a.k.a. Regression, Classification…
  - Apprentissage non supervisée (Unsupervised Learning)
    - Apprendre directement à partir des données
    - a.k.a. data manipulation, clustering …

Apprentissage automatique ?



Apprentissage supervisé (Supervised Learning)

- **Données d'entrainement (Training Data) :** $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(1)})$
  - Data: $\mathbf{x^{(i)}}$ (souvent des vecteurs)
  - **Label :** $y^{(i)}$ (une valeur, une ou plusieurs catégories, …)

- **Ex:**

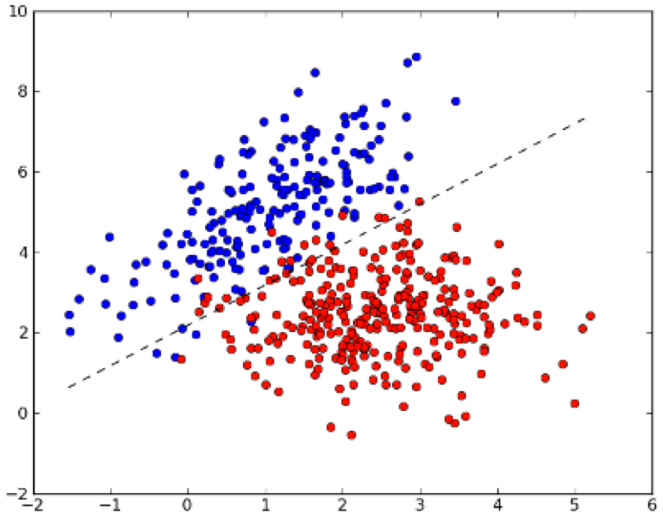
- But : On cherche à apprendre une fonction f**(w,x) (on cherche les w)** qui prédit la sortie y d'une nouvelle observation x

- Ex:

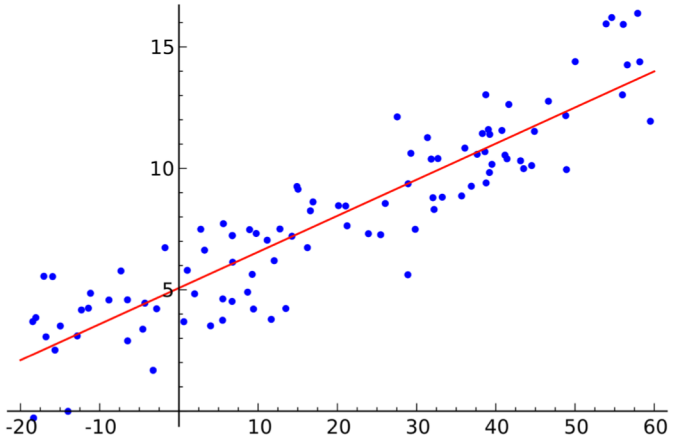
Apprentissage supervisé (Supervised Learning)

- Choix de la fonction : f(w,x)
  - Différents types de fonctions: fonctions linéaires, polynomiales, …
  - Prendre en compte aussi la sortie
    - Si Y est un réel → Regression
    - Si Y est une variable catégorielle (ensemble fini de catégories)→ classification

Classification



Regression



Apprentissage supervisé (Supervised Learning)

- Formulation:
  - Soient $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(21)})), \dots (x^{(m)}, y^{m})$): ensemble d'entrainement
    - $le\ ième\ exemple, x^{(i)}(x_1^{(i)}, x_2^{(i)}, .. x_n^{(i)})$ → $x_1^{(i)}$: number of words, $x_2^{(i)}$: occurrence of each word, …
    - $y^{(i)}$:
      - Binary: Relevant non relevant
      - Numeric: Very relevant(5), somewhat relevant(4), marginal relevant(3), somewhat irrelevant(2), very irrelevant(1)
    - f : linéaire(/polynomiale)→ $f_w(x^{(i)})$
  - Apprendre f (i.e. → w_i)
    - Minimiser la différence entre la sortie $y^{(i)} et\ f_w(x^{(i)})$
    - $J(w) = \sum_{i=1}^{m} loss(y^{(i)}, f_w(x^{(i)}))$ (m est l'ensemble des exemples)

Apprentissage supervisé (Supervised Learning)

- Comment choisir la meilleure fonction à minimiser ? (On parle aussi de fonction de coût (cost function), fonction objectif)
  - Loss: difference entre vraies sorties y_i et sortie prédite $w^T x_i$
    - Moindres carrés (Least square) (regression): $\sum_{i=1}^{m}(y^{(i)} - w^T x^{(i)})^2$
    - Hinge (Hinge Loss) (classification): $\max(0, 1 - y^{(i)} . w^T x^{(i)})$
    - Logistique (Logistic Loss) (classification): $\log(1 + \exp(- y^{(i)} . w^T x^{(i)}))$

Apprentissage supervisé

- Descente du gradient
  - Algo
    - Initialiser aléaoitement les w
    - Répéter
      - $w_j = w_j - \alpha \frac{\partial J(f_w,x,y)}{\partial w_j}$ (notation $\nabla \mathrm{J} = \left(\frac{\partial J(f_w,x,y)}{\partial w_j}\right), \mathrm{j} = 1, \mathrm{n}$
    - Jusqu'à convergence $J(w)$ soit très petit

Sur-apprentissage vs sous-apprentissage

- Sur apprentissage (Overfitting)
  - On colle trop aux données
  - Pose des problèmes de généralisation

- **Sous-apprentissage (Underfitting)**
  - **On ne colle pas du tout aux données**

Apprentissage supervisé: Regression linéaire

- Objectif prédire une fonction réelle
- Ensemble d'apprentissage:
  - $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(21)})), \ldots (x^{(m)}, y^{m}))$:
  - $x^{i} \in R^{n}$ et $y^{(i)} \in R$ $(une\ valeur\ réelle)$
  - L'hopothèse :
    - $f_w(x^{(i)}) = \sum_{i=0}^{n} x_i w_i$ $avec$ $x^0 = 1$
  - La fonction de côut (moindres carrés)

Apprentissage supervisé: Regression linéaire

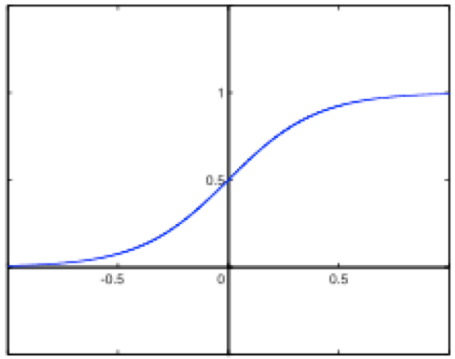
- Objectif prédire une function réelle
- Ensemble d'apprentissage
  - $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)}))$
  - $x^{(i)} \in \mathrm{R}^n , y^{(i)} \in \mathrm{R}$
  - L'hypothèse:
    - $f_w(x^{(i)}) = \sum_{j=0}^{n} x_j^{(i)} * w_j \ \ (avec\ x^0 = 1)$
  - La function de coût (objectif minimiser J(w))
    - $J(w) = \sum_{i=1}^{m} (y^{(i)} - f_w(x^{(i)}))$
- Il existe une solution exacte :
  - $\widehat{w} = (X^T X)^{-1} X^T Y$

Cassification et regression logistique

- Classification prédit des sorties binaires (y=1 ou y=0) (ou un ensemble fini den valeurs discrètes)
- Formulation
  - $((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)} \ldots, (x^{(m)}, y^{(m)})$
  - $x^{(i)} \in R \; et \; y^{(i)} un \; label \; (0 \; ou \; 1; on \; se \; limite \; \text{à} \; ce \; cas \; ici)$
- Algos
  - regression logistique (adaptation de la regression linéaire)
  - SVM, random forest, …

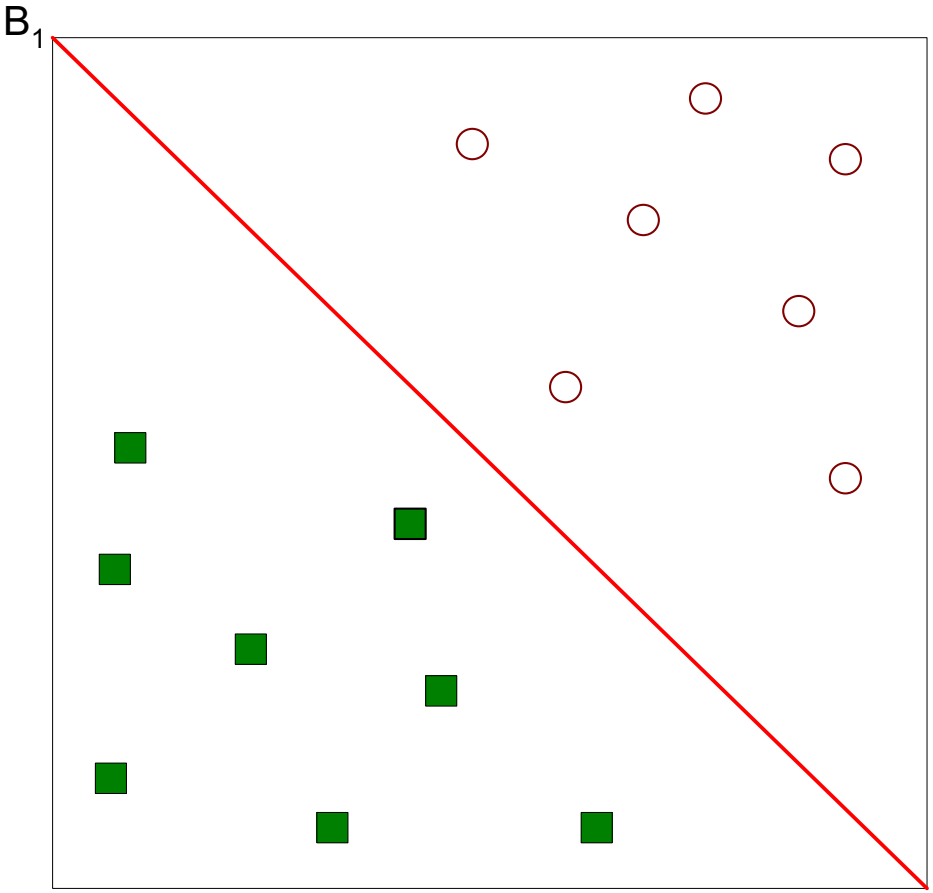
Classification par regression logistique

- On est sur la même formulation que la regression linéaire
- Formulation
  - $\left(\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right)$
  - $x^{(1)} \in R \; et \; y^{(i)} \; un \; label \; (0 \; ou \; 1, on \; se \; limite \; à \; ce \; cas)$
- Fonction de score, on applique une function logistique (par exemple une sigmoïde) à la sortie donnée par la regression linéaire
  - $g_w(x) = sigmoide\left(f_w(x)\right) = \frac{1}{1+\exp(w^T * x)}$



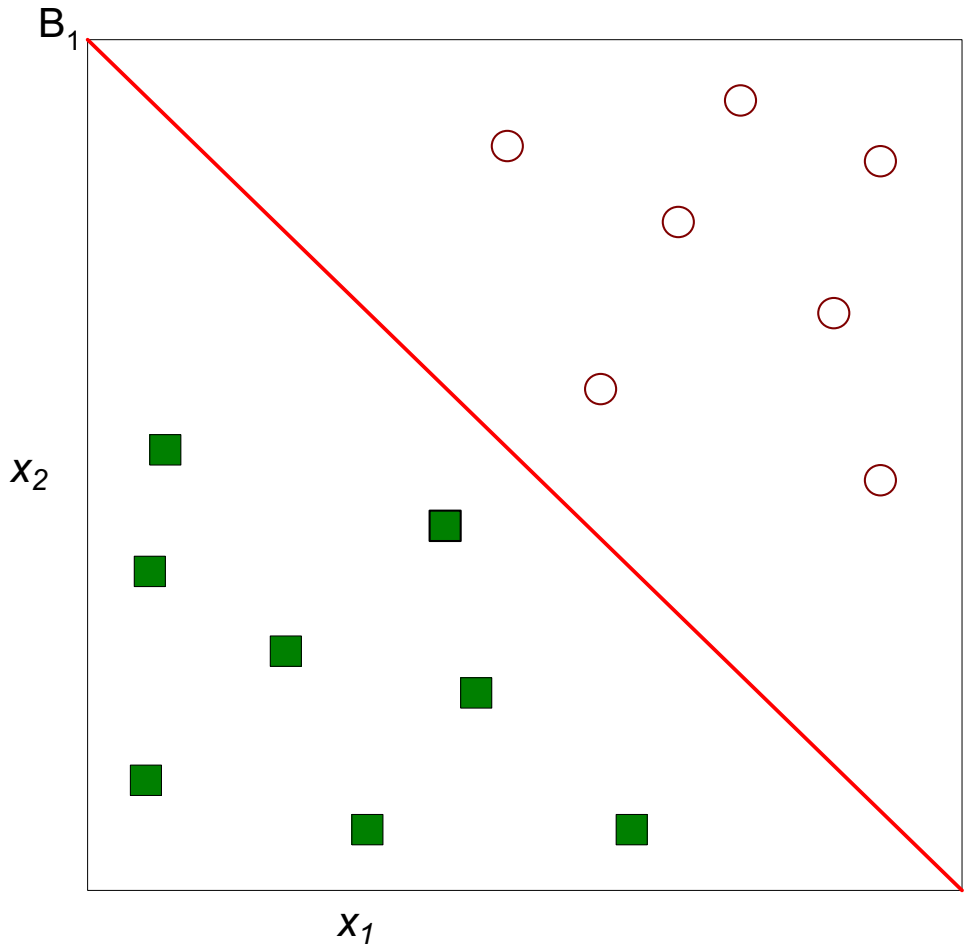
Machine à Vecteurs de Support

- Placer correctement les documents dans la catégorie qui leur correspond (P(+), NP(-) Puis maximiser la marge



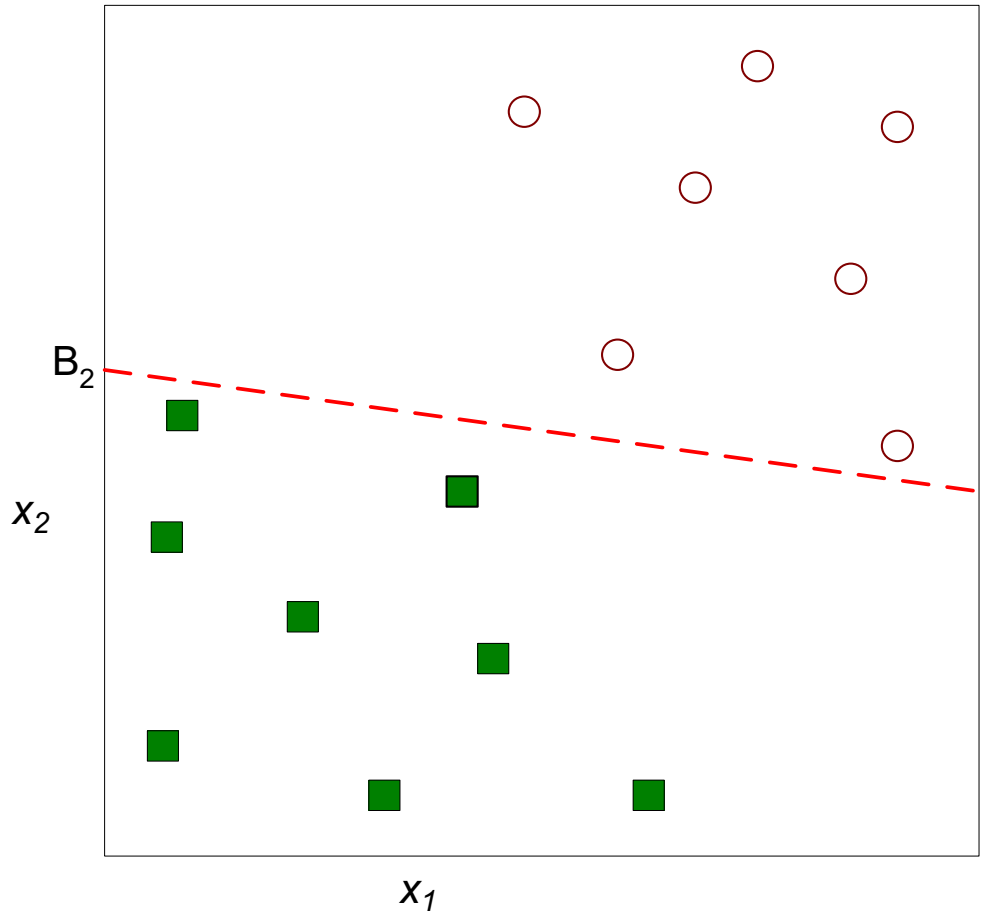
Machine à Vecteurs de Support

- Trouver un hyperplan linéaire qui permet de séparer les données
- Plusieurs solutions possibes



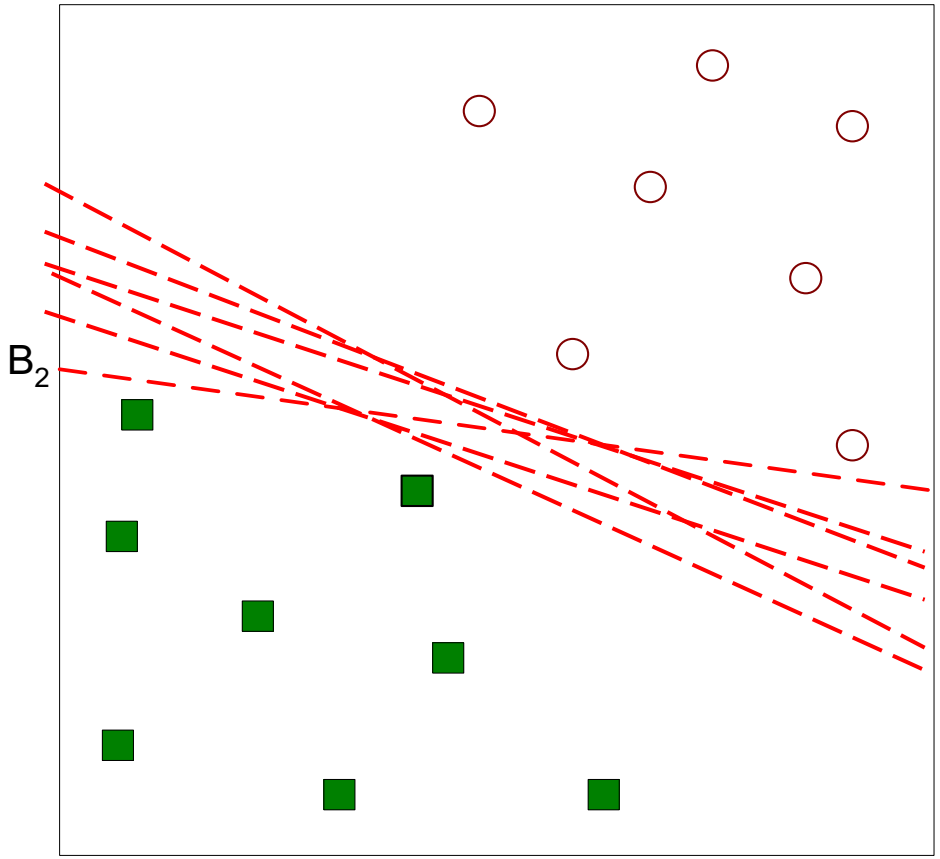
Machine à Vecteurs de Support

- Plusieurs solutions



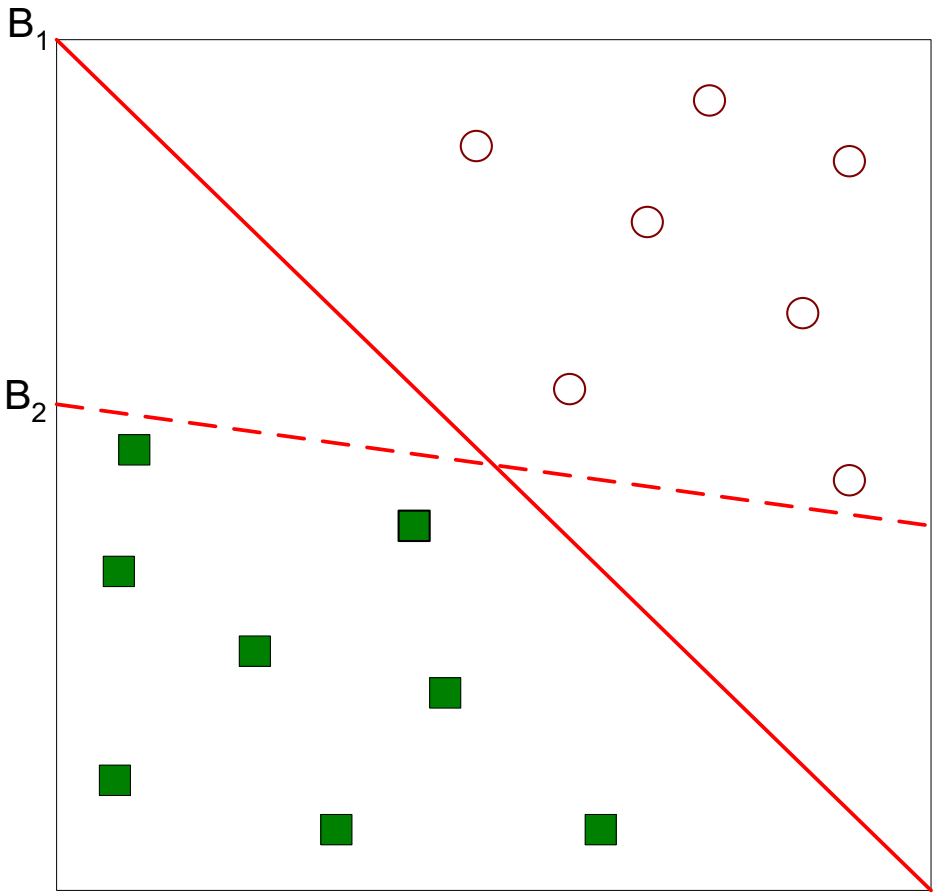
Machine à Vecteurs de Support

- Plusieurs solutions



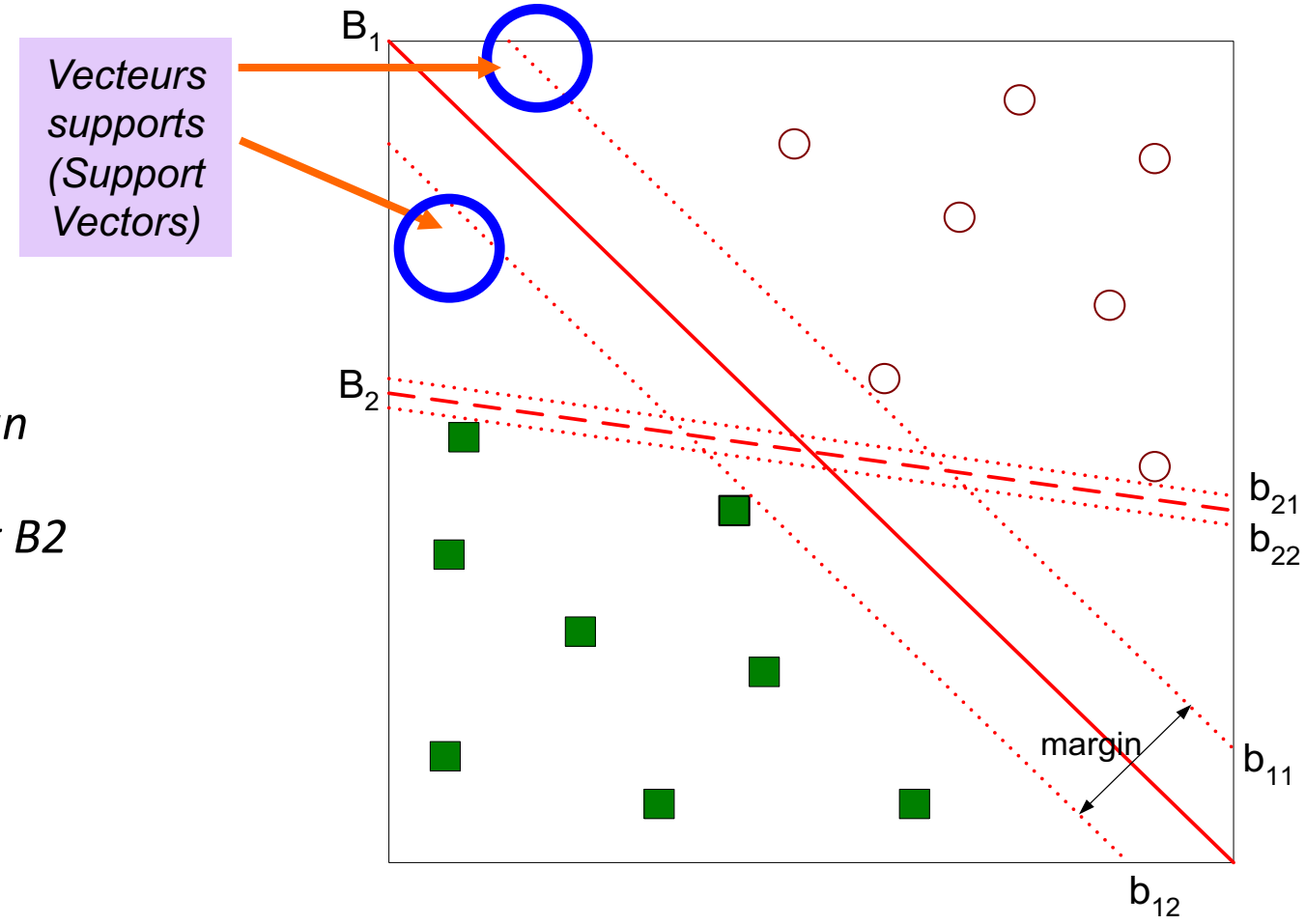
Machine à Vecteurs de Support

- Quel est le meilleur hyperplan? B1 or B2?
- Comment identifier le “meilleur”

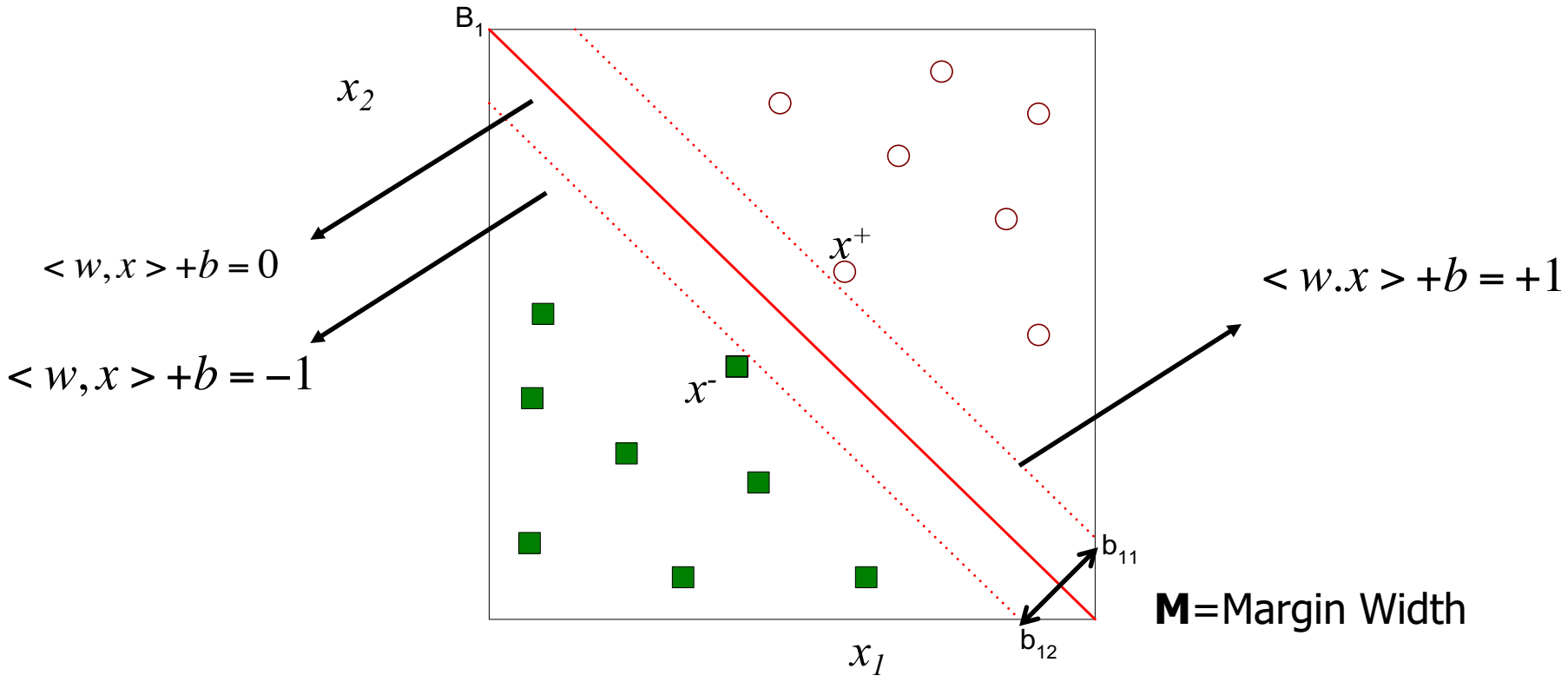


Machine à Vecteurs de Support

- *Trouver l'hyperplan quoi maximise la marge entre B1 et B2*



Machine à Vecteurs de Support



M=Margin Width

La distance d'un point à l'hyperplan est:

$$\frac{|\boldsymbol{w}.\boldsymbol{x} + b|}{\|\boldsymbol{w}\|}$$

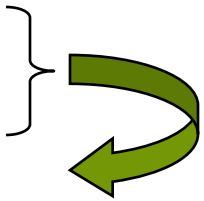
On cherche un hyperplan de marge M maximale :

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

SVM

- But: 1) Classer correctement toutes les données d'entrée

$$w.x_i + b \geq 1 \quad \text{if } y_i = +1$$
$$wx_i + b \leq 1 \quad \text{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$



2) Maximiser la marge $M = \frac{2}{|w|}$

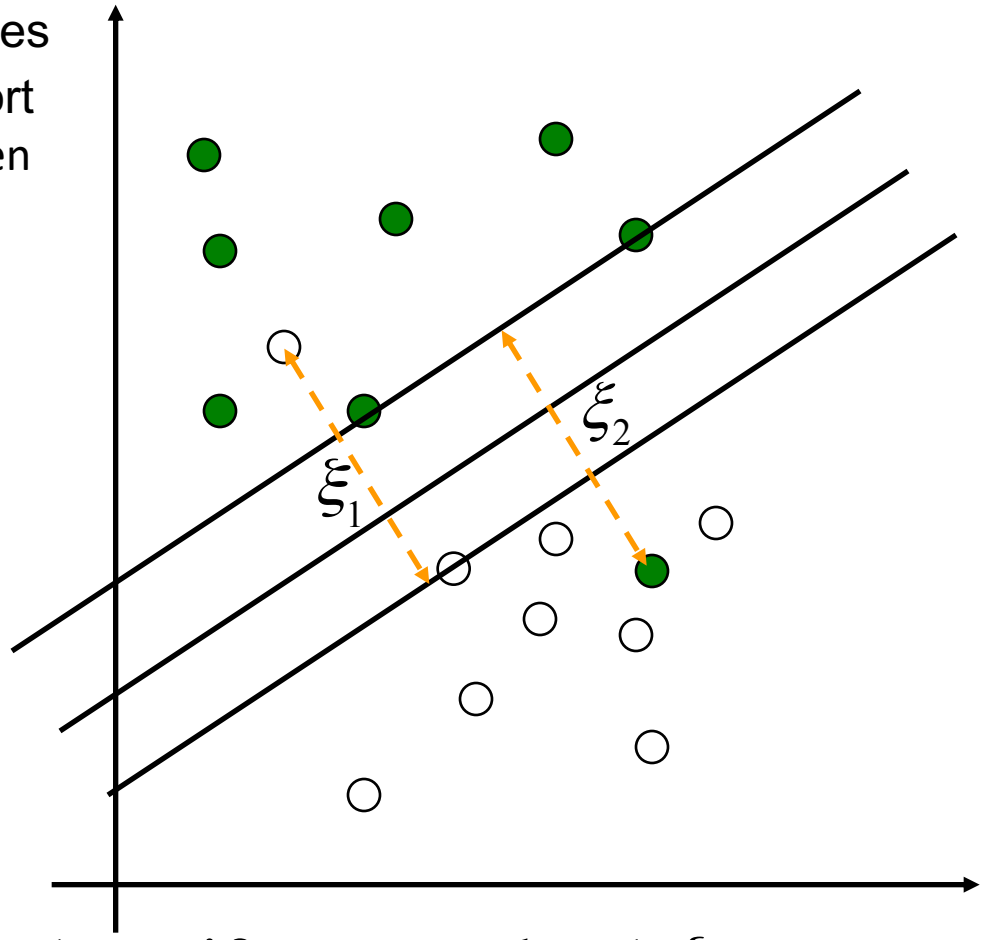
revient à minimiser $\frac{1}{2} w^t w$

- On peut le formuler différemment
- Minimize $\frac{1}{2} w^t w$

subject to $y_i (wx_i + b) \geq 1 \quad \forall i$

Marge souple

- Données bruitées
- Variables ressort (*slack variables* en anglais) ξ_i



$$f(x) = \begin{cases} 1 & \text{if } \langle w,x \rangle + b \geq 1-\xi_i \\ -1 & \text{if } \langle w,x \rangle + b \leq -1+\xi_i \end{cases}$$

SVM : Hard Margin v.s. Soft Margin

- **Ancienne formulation:**

Find $\mathbf{w}$ *and b such that*
Minimize $\frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ *and for all* $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- Nouvelle formulation avec les "slack variables"

Find $\mathbf{w}$ *and b such that*
Minimize $\frac{1}{2}\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ *for all* $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ *and* $\xi_i \geq 0$ *for all i*

Etapes de construction d'un algo d'apprentissage

- 1. choix des entrées et des sorties.
- 2. choix du codage entrées et sorties .
  - Espace des X et des Y.
- 3. Choisir une classe d'hypotyhèse (regression, classification) .
- 4. choisir une fonction objective à minimiser
- 5. Choisir un algorithme approprié

Panoplie de sites qui proposent des algos prêts à utiliser : Weka, scikit-learn, librairies sous python (Keras, Pytorch, …)

Recherche d’Information et apprentissage automatique

Apprentissage automatique et RI

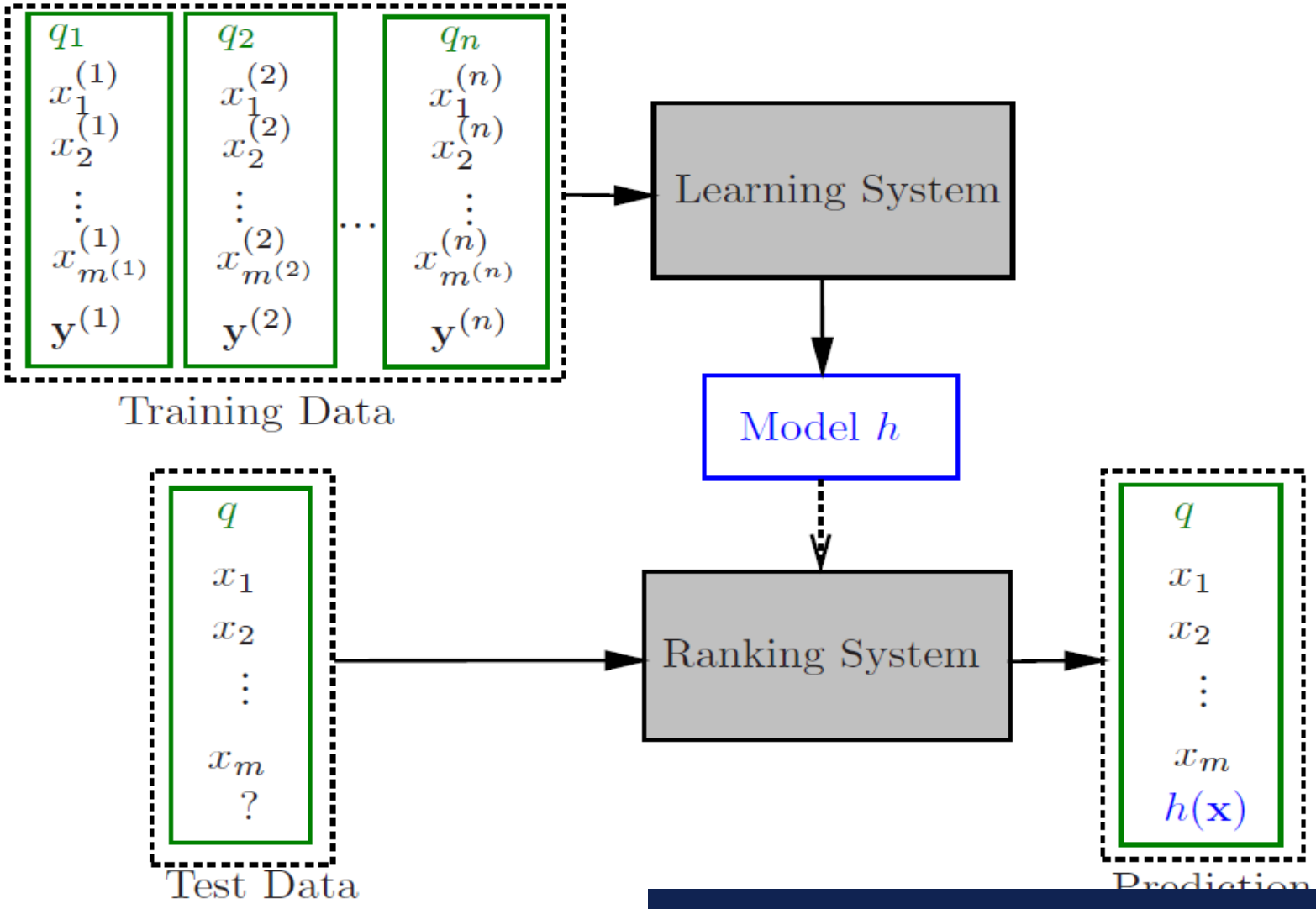
- L'idée de base
  - Etant donné un ensemble d'exemples d'apprentissage : des requêtes et les documents pertinents pour chaque requête
  - Extraire des critères (features) (données d'entrées) à partir de ces exemples
- Apprendre une fonction qui permet de combiner ces critères et trier les documents en fonction de leur pertinence vis-à-vis de la requête

Learning to rank: general approach

- 1. Choix des entrées et des sorties
  - Collecte des données d'entrainements (requêtes et documents pertinents)
- 2. Choix du codage des entrées et des sorties.
  - X: caractéristiques extraites de la requête et ou des documents
  - Y : sortie (pertinence) (Yes/No), rank d1>d2, ...
- 3. Choix de la de classe de la fonction hypothèse (classification/Régression) : SVM, Regression, Naive Bayes, Random Forest
- 4. Choix de la fonction d'erreur

Utiliser le modèle appris pour inférer le les résultats (documents) répondant aux nouvelles requêtes.

LTR : approche générale



Examples de caractéristiques

Table 2.3: Example Features of Learning to Rank for Web Search

Feature	Type	Explanation
Number of occurrences	Matching	number of times query exactly occurs in title, anchor, URL, extracted title, associated query, and body
BM25	Matching	BM25 scores on title, anchor, URL, extracted title, associated query, and body
N-gram BM25	Matching	BM25 scores of n-grams on title, anchor, URL, extracted title, associated query, and body
Edit Distance	Matching	edit distance scores between query and title, anchor, URL, extracted title, associated query, and span in body (minimum length of text segment including all query words [94])
Number of in-links	Document	number of in-links to the page
PageRank	Document	importance score of page calculated on web link graph
Number of clicks	Document	number of clicks on the page in search log
BrowseRank	Document	importance score of page calculated on user browsing graph
Spam score	Document	likelihood of spam page
Page quality score	Document	likelihood of low quality page

Types de sorties désirées

- Degré de pertinence l_k
  - Binaire: relevant vs. irrelevant
  - Pertinence graduée: Perfect > Excellent > Good > Fair > Bad
  - → Apprentissage par point (pointwise)
- Préférence par paire $l_{u,v}$
  - Le document u est plus pertinent que v
  - → Apprentissage par paires(Pairwise)
- Ordre total
  - Les documents sont triés {A,B,C,..} en fonction de leur pertinence
  - → Apprentissage par liste (Listwise)

Apprentissage en fonction du type de la sortie (apprise)

- Par point (Pointwise)
  - Entrée: caractéristiques d'une requête et un SEUL
  - Sortie: scores ou classe de pertinence de par document
- Par paire (Pairwise)
  - Entrée: caract. requêteet paire de document (d1>d2)
  - Output: préférence
- Par liste (Listwise)
  - Input: une requête et une liste de documents (triés par préférence)
  - Output: Liste triée de documents

Classes d'approches de LTR

- On retrouve les deux grandes classes d'algorithmes
  - Combinaison linéaire de critères (régression)
    - Apprendre des scores de pertinence
  - Classification
    - Apprendre des catégories (Pertinent, Non pertinent)

LTR par Régression linéaire (1)

- Collecter des exemples d'entraînement (q, d, y) triplets
  - Pertinence y est binaire (peut être graduée)
  - Identifier/construire les caractéristiques (Document, requête) : $x(x_1, x_2, \ldots x_n)$
  - Exemple $x=(x_1, x_2)$, deux caractéristiques :
    - x_1 est la similarité (entre q et d) ,
    - x_2 est la proximité entre les termes de la requête dans le document

LTR par Régression linéaire (2)

- La pertinence est vue comme une valeur de score
- → apprendre la fonction de score qui combine les différentes caractéristiques
  - $f_w\left(x^{(i)}\right) = \sum_{j=0}^{n} x_j^{(i)} * w_j \;\; (avec\ x^0 = 1)$
    - w les poids ajustés par apprentissage
    - (x_1, ..x_n) les caractéristiques du document-requête

- Trouver les w_i qui minimisent l'erreur suivante (moindres carrées):
  - $J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(y^{(i)} - f_w\left(x^{(i)}\right)\right)^2$
  - → pertinence (y=1), non pertinence (y=0)

LTR par Régression linéaire (3)

- Apprendre une fonction de score qui combine les deux « features » (x_1,x_2)

$$f(d, q) = w_1 * x_1(d, q) + w_2 * x_2(d, q) + w_0$$

example	docID	query	x_1	x_2	judgment
$x^{(1)}$	37	linux operating system	0.032	3	*relevant*
$x^{(2)}$	37	penguin logo	0.02	4	*nonrelevant*
$x^{(3)}$	238	operating system	0.043	2	*relevant*
$x^{(4)}$	238	runtime environment	0.004	2	*nonrelevant*
$x^{(5)}$	1741	kernel layer	0.022	3	*relevant*
$x^{(6)}$	2094	device driver	0.03	2	*relevant*
$x^{(7)}$	3191	device driver	0.027	5	*nonrelevant*

LTR par Classification

- Ramène la RI à un problème de classification
  - Une requête, un document, une classe (Pertinent, non pertinent) (plusieurs catégories)
- Différents classifieurs
  - Logistique régression, SVM (Machine à Vecteurs de Support), Random Forest, Bayésien…
- Exemple SVM
  - On cherche une fonction de décision qui permet de séparer les documents pertinents des documents non pertinents.
  - La fonction est de la forme :
    - f(x)=sign <(x.w)+b>
  - On souhaite que
    - $f(x) \leq -1$ pour non pertinent
    - et $f(x) \geq 1$ pour pertinent

Apprendre à ordonnancer (learning to rank)

- Apprendre la pertinence individuelle des documents (pointwise) n'est pas le on moyen moyen pour aborder le «ranking »:

- Cette formulation ne traduit pas la notion de tri « chère à la RI »
  - L'idéal est de fournir en entrée des préférences entre documents

Approche par paire

- Au lieu de juger chaque document seul, on juge des paires de documents.
- Nous écrivons $d_u \prec d_v$ pour "d_u précède d_v dans la liste des résultats »
- Nous construisons à nouveau un vecteur de critères $x_u = (d_u, q)$ pour chaque paire document-requête
- PUIS, pour chaque paire de documents d_u et d_v, nous formons le vecteur de différences de critères
- $$\Phi(d_u , d_v , q) = x_u - x_v$$
- L'entrée de l'algorithme d'apprentissage sera alors le vecteur $\Phi(d_u , d_v , q)$ en prenant des combinaisons de u et v.

Approche par paire: RankSVM

- Si d_u est jugé plus pertinent que d_v, alors on assigne au vecteur $\Phi(d_u , d_v , q)$ la classe $y_{u,v} = +1$; sinon−1.
- Cela nous donne un ensemble de données d'apprentissage
- Nous pouvons alors utiliser un SVM sur cet ensemble d'apprentissage

Find $\mathbf{w}$ and b such that

Minimize $\frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma_{u,v}\xi_{u,v}$ and for all $\{(x_u, x_v, y_{u,v})\}$

$y_{u,v}\,(\mathbf{w}^T(\mathbf{x_u} - \mathbf{x_v}) + b) \geq 1 - \xi_{u,v}$ and $\xi_{u,v} \geq 0$ for all u,v

SVM: phase de test

- Comment utiliser SVM lors de la phase de test
  - On récupère les documents qui comportent les termes de la requête, soient (d_1, d_2, d_3, ..)
  - Comment les trier ?
  - Construire toutes les combinaisons entre les docs puis vérifier si d_u plus pertinent que d_v ssi $(\mathbf{w}^T (\mathbf{x_u} - \mathbf{x_v}) + b) \geq 1$

- Cette utilisation est toutefois coûteuse. On utilise en fait en pratique directement le score « SVM »

$$RSV(q, d_u) = (w^*.x_u)$$

Conclusion

- Il existe une panoplie de sites listant tous ces algorithmes