Invited Talk:

Object Ranking

Toshihiro Kamishima

http://www.kamishima.net/

National Institute of Advanced Industrial Science and Technology (AIST), Japan

Preference Learning Workshop (PL-09) @ ECML/PKDD 2009, Bled, Slovenia

START

Today, we'd like to talk about object ranking tasks and methods for this task.

Introduction

- **Object Ranking:** Task to learn a function for ranking objects from sample orders
- Discussion about methods for this task by connecting with the probabilistic distributions of rankings
- Several properties of object ranking methods

Order / Ranking

object sequence sorted according to a particular preference or property

ex. an order sorted according to my preference in *sushi*
“I prefer fatty tuna to squid” but “The degree of preference is not specified”

An object ranking task is to learn a function for ranking objects from given sample orders. We will discuss methods for this task by connecting with the probabilistic distributions of rankings.
We begin with what is an order or ranking. An order is an object sequence sorted according to a particular preference or property. For example, this is an order sorted according to my preference in sushi. This order indicates that “I prefer a fatty tuna to squid”, but “The degree of preference is not specified.”

Outline

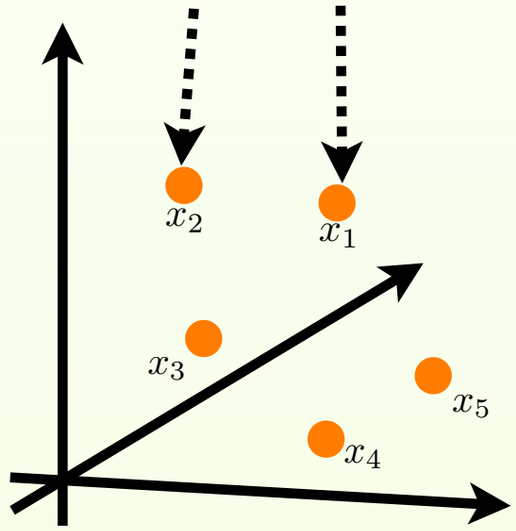
- **What's object ranking**
  - Definition of an object ranking task
  - Connection with regression and ordinal regression
  - Measuring the degree of preference
- **Probability distributions of rankings**
  - Thurstonian, paired comparison, distance-based, and multistage
- **Six methods for object ranking**
  - Cohen's method, RankBoost, SVOR (a.k.a. RankingSVM), OrderSVM, ERR, and ListNet
- **Properties of object ranking methods**
  - Absolute and relative ranking
- **Conclusion**

This is an outline of our talk. We begin by talking about definition of an object ranking task, its connection to regression or ordinal regression, and relation to the way for measuring the degree of preference. Next, we will introduce probability distributions of rankings, because these are closely connected with object ranking methods. Then, we briefly review six methods for object ranking: Cohen's method, RankBoost, SVOR a.k.a Ranking SVM, OrderSVM, expected rank regression, and ListNet. Additionally, we will show several properties of these methods.

Object Raking Task

sample order set

$$O_1 = x_1 \succ x_2 \succ x_3$$
$$O_2 = x_1 \succ x_5 \succ x_2$$
$$O_3 = x_2 \succ x_1$$



feature space

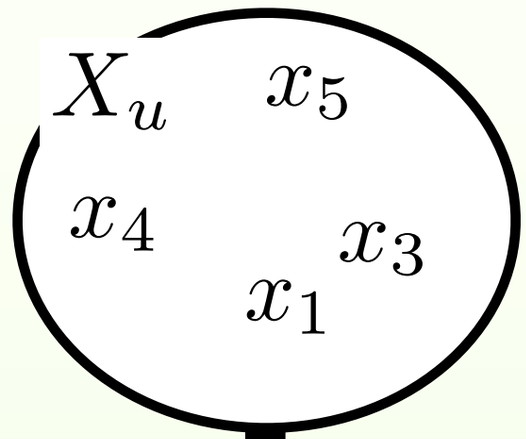
objects are represented by feature vectors

object ranking method

ranking function

unordered objects

feature values are known



$$\hat{O}_u = x_1 \succ x_5 \succ x_4 \succ x_3$$

estimated order

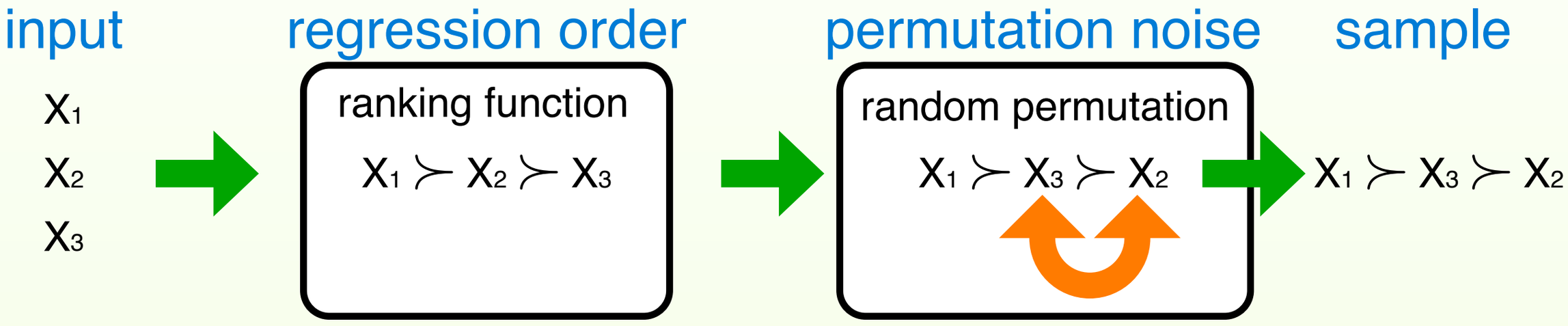
- objects that don't appeared in training samples have to be ordered by referring feature vectors of objects

First of all, we'd like to show an object ranking task. Training sample orders are sorted according to the degree of the target preference to learn. Objects in these orders are represented by feature vectors. From these samples, an object ranking method acquires a ranking function. By applying this learned function, unordered objects can be sorted according to the degree of the target preference. Note that objects that don't appeared in training samples have to be ordered by referring feature vectors of objects.

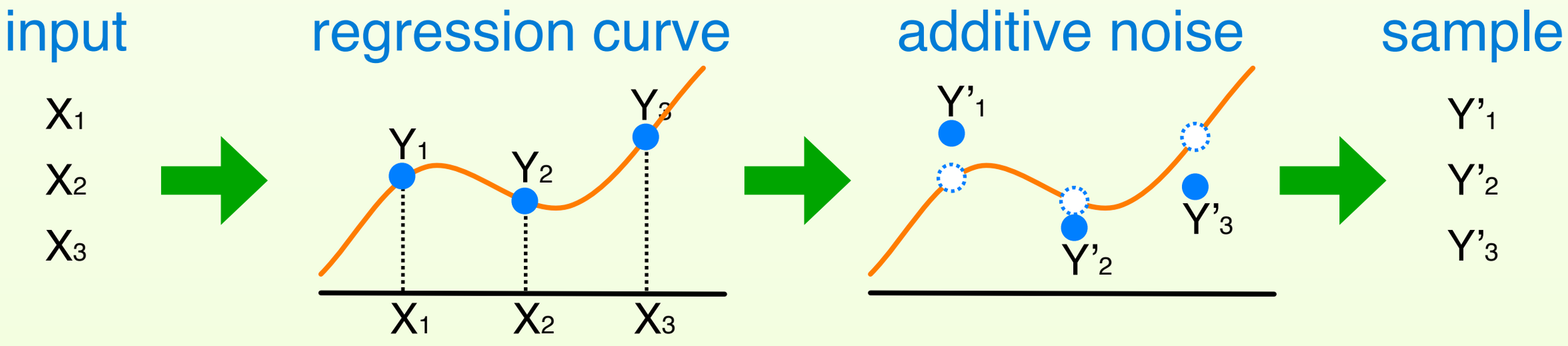
Object Ranking vs Regression

Object Ranking: regression targeting orders

generative model of object ranking



generative model of regression



Next, we will show the connection between object ranking and regression. Object ranking can be considered as regression targeting orders. This is a generative model of object ranking. A ranking function sorts input objects according to the degree of the target preference and a regression order is generated. This order is then affected by permutation noise, and a sample order is generated. This model is very similar to a generative model of regression, like this.

Ordinal Regression

Ordinal Regression [McCullagh 80, Agresti 96]
Regression whose target variable is ordered categorical

Ordered Categorical Variable
Variable can take one of a predefined set of values that are ordered
ex. { good, fair, poor}

Differences between "ordered categories" and "orders"	
Ordered Category	**Order**
The # of grades is **finite**	The # of grades is **infinite**
ex: For a domain {good, fair, poor}, the # of grades is limited to three	
Absolute Information is contained	It contains purely **relative** information
ex: While "good" indicates absolutely preferred, "$x_1 > x_2$" indicates that x_1 is relatively preferred to x_2	

Object ranking is more general problem than ordinal regression as a learning task

Ordinal regression is also analogous to object ranking. Ordinal regression is regression whose target variable is ordered categorical. Ordered categorical variables can take one of predefined a set of values that are ordered. For example, good, fair, poor.
This is a summary of the differences between "ordered categories" and "orders." The number of grades of an ordered category is finite, and ordered categorical values provide absolute information.
Due to these differences, object ranking is more general problem than ordinal regression as a learning task.

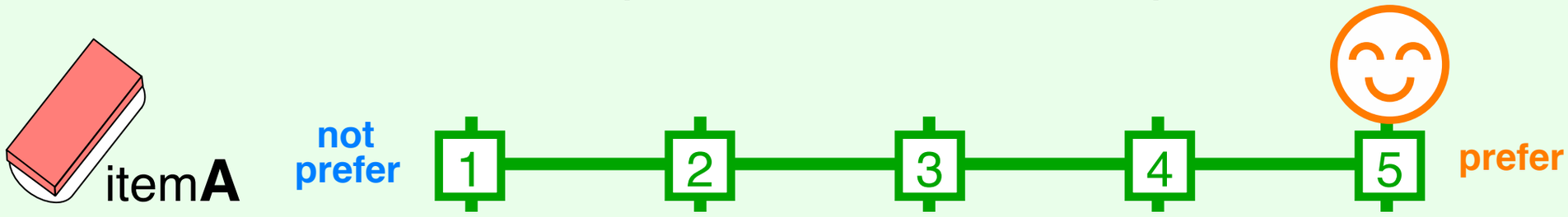
Measuring Preference

ordinal regression (ordered categories)

Scoring Method / Rating Method

Using scales with scores (ex. 1,2,3,4,5) or ratings (ex. gold, silver, bronze)

The user selects “5” in a five-point scale if he/she prefers the item A



object ranking (orders)

Ranking Method

Objects are sorted according to the degree of preference

The user prefers the item A most, and the item B least

These two tasks are related to schemes for measuring the degree of preference.
Ordinal regression is related to scoring and rating methods. The user selects “5” in a five-point-scale, if he/she prefers the item A.
On the other hand, object ranking is related to ranking method. By the user, objects according to the degree of preference.
In next two slides, we will compare these schemes.

Demerit of Scoring / Rating Methods

Difficulty in caliblation over subjects / items

Mappings from the preference in users' mind to rating scores differ among users

- Standardizing rating scores by subtracting user/item mean score is very important for good prediction [Herlocker+ 99, Bell+ 07]
- Replacing scores with rankings contributes to good prediction, even if scores are standardized [Kmaishima 03, Kamishima+ 06]

presentation bias

The wrong presentation of rating scales causes biases in scores

- When prohibiting neutral scores, users select positive scores more frequently [Cosley+ 03]
- Showing predicted scores affects users' evaluation [Cosley+ 03]

Scoring and rating methods have the following demerits.
First, mappings from the preference in users' mind to rating scores differ among users. Standardizing rating scores is important for good prediction, and replacing scores with rankings contributes to good prediction.
Second, the wrong presentation of rating scales causes biases in scores. These are examples of such biases.

Demerit of Ranking Methods

Lack of absolute information

Orders don't provide the absolute degree of preference

- Even if "$x_1 > x_2$" is specified, x_1 might be the second worst

Difficulty in evaluating many objects

Ranking method is not suitable for evaluating many objects at the same time

- Users cannot correctly sort hundreds of objects
- In such a case, users have to sort small groups of objects in many times

On the other hand, demerits of ranking methods are as follows.
First, orders don't provide the absolute degree of preference. Even if x1 is preferred to x2 is specified, x1 might be the second worst
Second, ranking method is not suitable for evaluating many objects at the same time. Because users cannot correctly sort hundreds of objects
That's all we have to say about an object ranking task and its connections to regression and ordinal regression.

Distributions of Rankings

generative model of object ranking

regression order + permutation noise

The permutation noise part is modeled by using probabilistic distributions of rankings

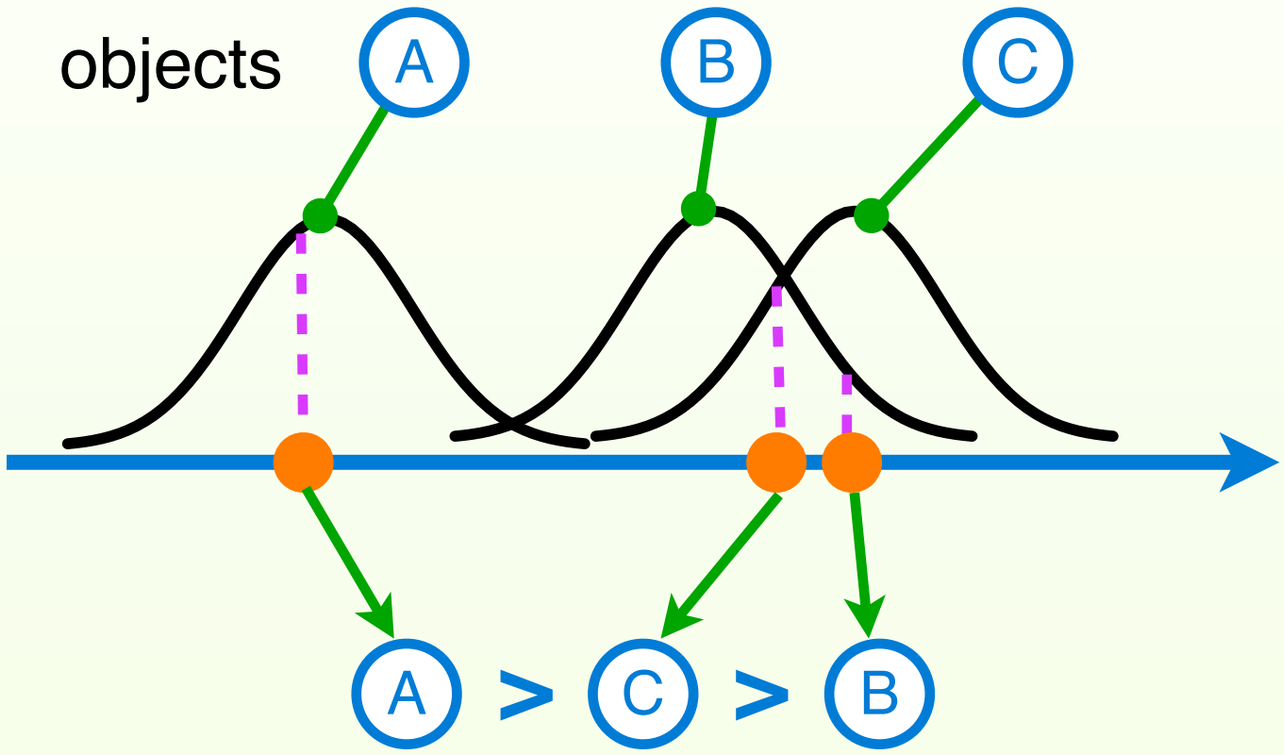
4 types of distributions for rankings [Crichlow+ 91, Marden 95]

- **Thurstonian:** Objects are sorted according to the objects' scores
- **Paired comparison:** Objects are ordered in pairwise, and these ordered pairs are combined
- **Distance-based:** Distributions are defined based on the distance between a modal order and sample one
- **Multistage:** Objects are sequentially arranged top to end

Now, we move on to the probabilistic distributions of rankings, because these are closely connected object ranking methods.
Let's remind that a generative model of object ranking composed of two parts: regression order and permutation noise. The later is modeled by using distributions of rankings. These distributions can be classified into four types. We will introduce these distributions one by one.

Thurstonian

Thurstonian model (a.k.a Order statistics model)
Objects are sorted according to the objects' scores



For each object, the corresponding scores are sampled from the associated distributions

Sort objects according to the sampled scores

distribution of scores

- **Normal Distribution:** Thurstone's law of comparative judgment [Thurstone 27]
- **Gumbel Distribution:** CDF is $1 - \exp(-\exp((x_i - \mu_i)/\sigma)$

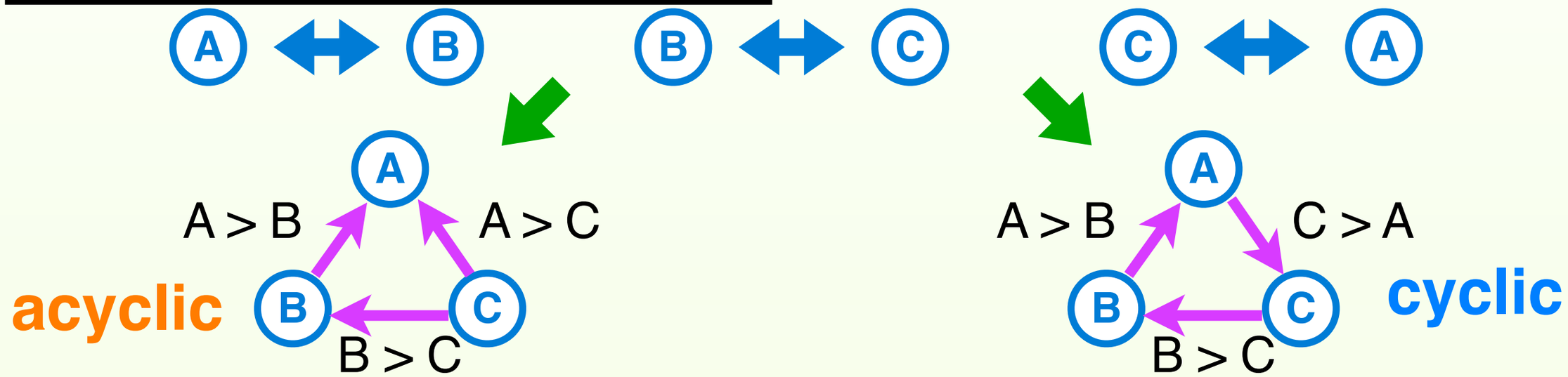
First, in a Thurstonian model, objects are sorted according to the objects' scores.
For each object, the corresponding scores are sampled from the associated distributions. Then, objects are sorted according to the sampled scores.
If its score distribution is normal, the model is known as "Thurstone's law of comparative judgment." Another choice is Gumbel distribution, which is related to order statistics.

Paired Comparison

Paired comparison model

Objects are ordered in pairwise, and these ordered pairs are combined

Objects are ordered in pairwise



generate the order: A > B > C

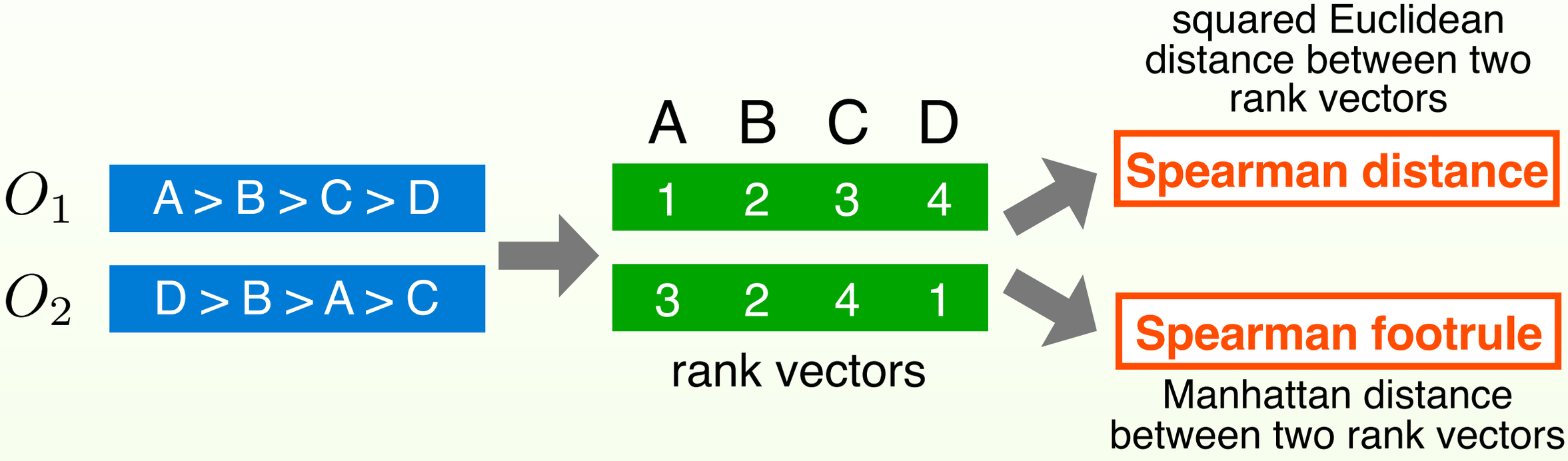
Abandon and retry

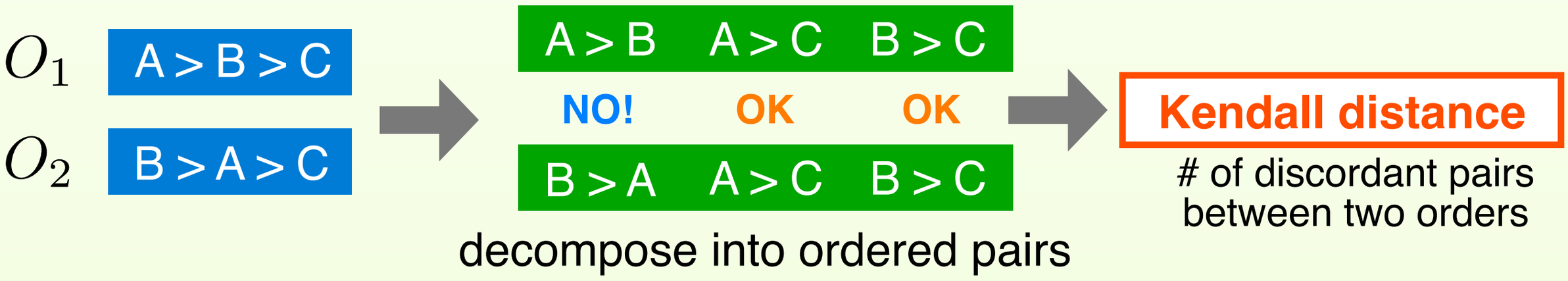
parameterization

- **Babinton Smith model:** saturated model with ${}_nC_2$ paramaters [Babington Smith 50]
- **Bradley-Terry model:** $\Pr[x_i \succ x_j] = \frac{v_i}{v_i + v_j}$ [Bradley+ 52]

Second, in a paired comparison model, objects are compared, and these ordered pairs are combined.
In the first step, ordered pairs are generated independently. If these ordered pairs are cyclic, namely, these are contradicted each other, all pairs are abandoned and generated again. If these pairs are acyclic, all objects can be sorted without contradiction.
The most general saturated model is know as Babington Smith model, because Babington Smith firstly calculated the moments of distributions. A Bradley-Terry model has less parameters.

Distance between Orders





Before showing distance-based model, we briefly review the distances between orders.
Orders are first converted into rank vectors, whose entries are ranks of the corresponding objects. Spearman distance is squared Euclidean distance between rank vectors, and Spearman foot rule is Manhattan distance. Kendall distance is defined as the number of discordant pairs between two orders.

Distance-based

Distance-based model

Distributions are defined based on the distance between orders

consentration parameter　　modal order/ranking

$$\Pr[O] = C(\lambda)\exp(-\lambda d(O, O_0))$$

normalization factor　　distance

distance

- **Spearman distance:** Mallows' θ model
- **Kendall distance:** Mallows' ϕ model

These are the special cases of Mallows' model (ϕ=1 or θ=1),
which is a paired comparison model that defined as:

$$\Pr[x_i \succ x_j] = \frac{\theta^{i-j}\phi^{-1}}{\theta^{i-j}\phi^{-1}+\theta^{j-i}\phi}$$

[Mallows 57]

Third, in a distance-based model, distributions are defined based on the distance between orders like this formula. When orders are more distant from the modal ranking, the orders are less frequently generated. If Spearman distance is used, this model is called Mallows' θ model. If Kendall distance is used, this is called Mallows' ϕ model. These are special cases of this Mallows' model.

Multistage

Multistage model
Objects are sequentially arranged top to end

Plackett-Luce model [Plackett 75]

ex. objects {A,B,C,D} is sorted into A > C > D > B

$$\Pr[A] = \frac{\theta_A}{\theta_A + \theta_B + \theta_C + \theta_D}$$

θ_A: a param of the top object

denominator: total sum of params

$$\Pr[A>C \mid A] = \frac{\theta_C}{\theta_B + \theta_C + \theta_D}$$

θ_C: a param of the second object

denominator: params for A is eliminated

$$\Pr[A>C>D \mid A>C] = \frac{\theta_D}{\theta_B + \theta_D}$$

$$\Pr[A>C>D>B \mid A>C>D] = \theta_B / \theta_B = 1$$

The probability of the order, A > C > D > B, is

$$\Pr[A>C>D>B] = \Pr[A]\ \Pr[A>C \mid A]\ \Pr[A>C>D \mid A>C]\ 1$$

Finally, in a multistage model, objects are sequentially arranged top to end. Plackett-Luce model generates ranking as follows. The top object is generated with this probability. The numerator is a parameter of the top object, and the denominator is the total sum of parameters. The second object is generated with this probability. The numerator is a parameter of the second object, but the denominator is the sum of parameters of objects excluding already ranked objects. In consequence, the probability of this order is the product of these probabilities.
Now, we have completed to introduce four types of distributions for rankings.

Object Ranking Methods

Object Ranking Methods

- **permutation noise model:** orders are permutated accoding to the distributions of rankings
- **regression order model:** representation of the most probable rankings
- **loss function:** the definition of the “goodness of model”
- **optimization method:** tuning model parameters

connection between distributions and permutation noise model

- **Thurstonian:** Expected Rank Regression (ERR)
- **Paired comparison:** Cohen’s method
- **Distance-based:** RankBoost, Support Vector Ordinal Regression (SVOR, a.k.a RankingSVM), OrderSVM
- **Multistage:** ListNet

Next, I'd like to move on to the main topic: object ranking methods. Object ranking methods consist of these components. Like other ML methods, loss functions and optimization methods are of course important. Permutation noise models of object ranking methods are connected with the distributions of rankings as shown in this table. Regression order model represents the most probable rankings as in the next slide.

Regression Order Model

linear ordering: Cohen's method

1. Given the features of any object pairs, $\mathbf{x}_i$ and $\mathbf{x}_j$, $f(\mathbf{x}_i, \mathbf{x}_j)$ represents the preference of the object i to the object j
2. All objects are sorted so as to maximize: $\sum_{\mathbf{x}_i \succ \mathbf{x}_j} f(\mathbf{x}_i, \mathbf{x}_j)$

This is known as **Linear Ordering Problem** in an OR literature [Grötschel+ 84], and is NP-hard ➡ Greedy searching solution $O(n^2)$

sorting by scores: ERR, RankBoost, SVOR, OrderSVM, ListNet

1. Given the features of an object, $\mathbf{x}_i$, $f(\mathbf{x}_i)$ represents the preference of the object i
2. All objects are sorted according to the values of $f(\mathbf{x})$

Computational complexity for sorting is O(n log(n))

Regression order models can be classified into two types.
A "linear ordering" model is used only in Cohen's method. Score function represents the preference of the object i to the object j. All objets are sorted so as to maximize the sum of scores over all concordant ordered pairs. This is known as linear ordering problem in an operational research literature.
A "sorting by scores" model is used in all other methods. In this model, all objects are sorted according to the values of the score function.
We next show six object ranking methods one by one.

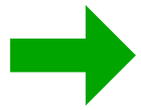
Cohen's Method [Cohen+ 99]

- **permutation noise model = paired comparison**
- **regression order model = linear ordering**

sample orders are decomposed into ordered pairs

$$A \succ B \succ C$$
$$D \succ E \succ B \succ C$$
$$A \succ D \succ C$$

training sample orders



$$A \succ B, A \succ B, B \succ C$$
$$D \succ E, D \succ B, D \succ C, \cdots$$
$$A \succ D, A \succ C, D \succ C$$

ordered pairs

the preference function that one object precedes the other

$$f(\mathbf{x}_i, \mathbf{x}_j) = \Pr[\mathbf{x}_i \succ \mathbf{x}_j; \mathbf{x}_i, \mathbf{x}_j]$$

Unordered objects can be sorted by solving linear ordering problem

Cohen's method adopts a paired comparison approach.
Sample orders are first decomposed into ordered pairs. From these pairs, the algorithm learns a preference function that one object precedes the other. Unordered objects can be sorted by solving linear ordering problem.

RankBoost [Freund+ 03]

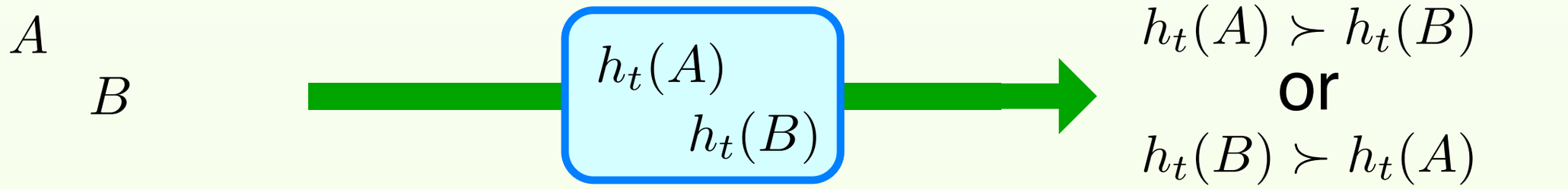
- **permutation noise model = distance based (Kendall distance)**
- **regression order model = sorting by scores**

find a linear combination of weak hypotheses by boosting

objects

weak hypotheses

partial information about the target order



score function: $f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

This function is learned so that minimizing the number of discordant pairs

⇓

minimizing the Kendall distance between samples and the regression order

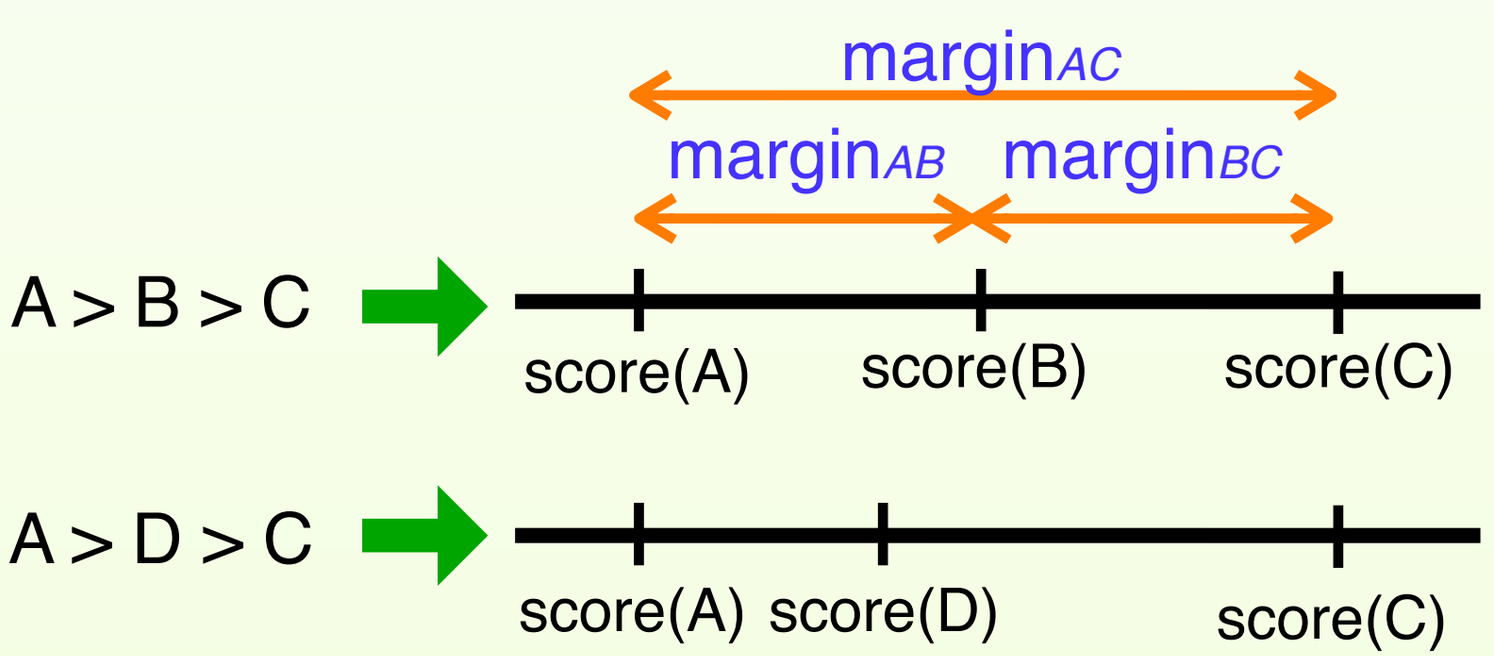
The RankBoost tries to find a score function, which is a linear combination of weak hypotheses. Given an object, weak hypotheses provides some partial information about the target order. This function is learned by boosting so that minimizing the number of discordant pairs. Therefore, this method is considered as minimizing the Kendall distance to the regression order.

Support Vector Ordinal Regression
(SVOR; a.k.a RankingSVM) [Herbrich+ 98, Joachims 02]

- **permutation noise model = distance based (Kendall distance)**
- **regression order model = sorting by scores**

find a score function that maximally separates preferred objects from non-preferred objects

sample orders | score & margin | Objective



maximize:

$$\sum_{X,Y} \mathrm{margin}_{XY}$$

Support Vector Ordinal Regression is also known as RankingSVM. SVOR tries to find a score function that maximally separates preferred objects from non-preferred ones. To maximize the separation, the margins between the closest pair is maximized.

OrderSVM [Kazawa+ 05]

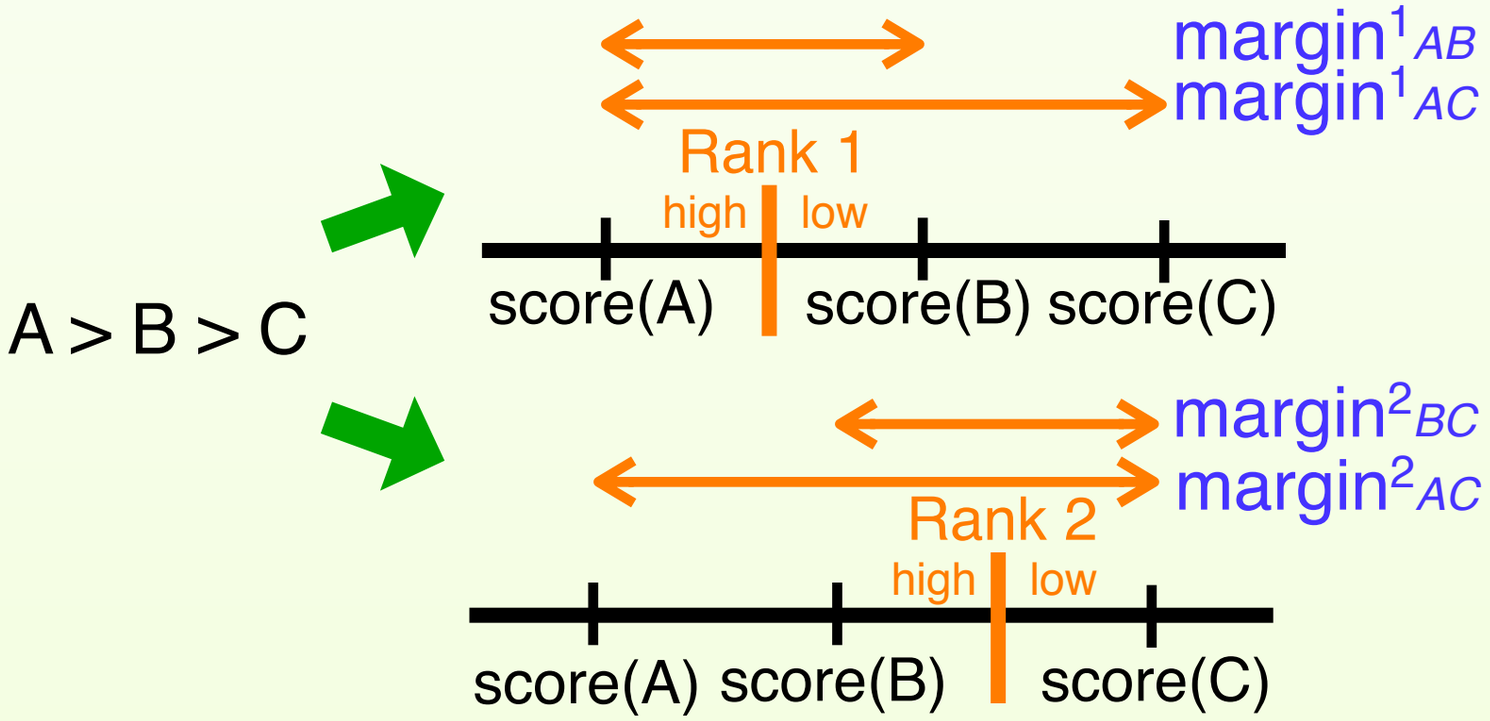
- **permutation noise model = distance based (Spearman footrule)**
- **regression order model = sorting by scores**

find a score function which maximally separates higher-ranked objects from lower-ranked ones on average

sample orders

score & margin

Objective



maximize:

$$\sum_j \sum_{X,Y} \mathrm{margin}^j_{XY}$$

Another SVM for object ranking is OrderSVM.
OrderSVM tries to find a score function which maximally separate higher-ranked from lower-ranked by comparing some threshold. These separations are performed for all thresholds between adjacent ranks.

SVM and Distance-based Model

SVOR (RankingSVM)

minimizing the # of misclassifications in orders of object pairs

↓

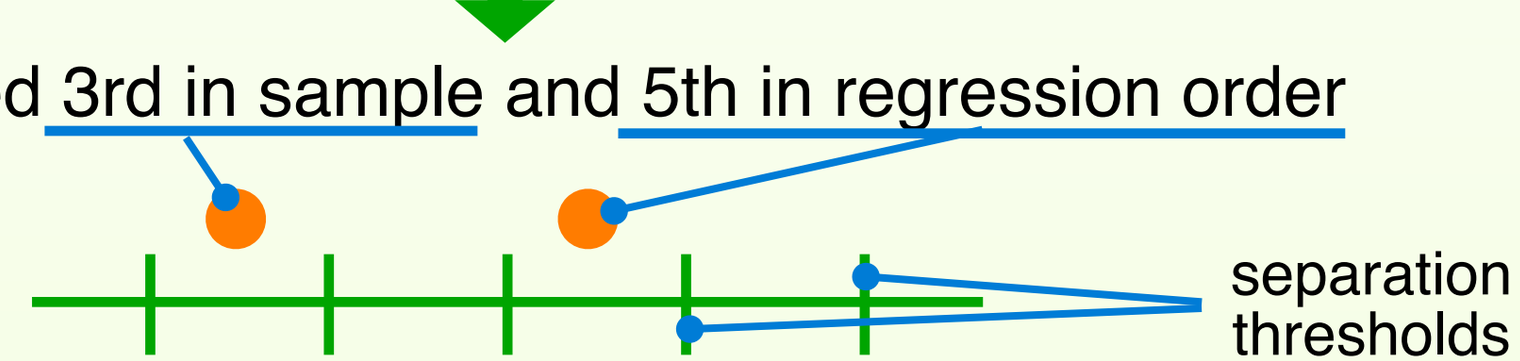
minimizing the Kendall distance between regression order and samples

OrderSVM

separate the objects that ranked lower than *j*-th from the higher ones,
and these separations are summed over all ranks *j*

↓

ex: object A is ranked 3rd in sample and 5th in regression order



# of misclassifications = absolute difference between ranks

↓

minimizing the Spearman footrule between regression order and samples

We here summarize the relations between two SVMs and a distance-based model.
SVOR minimizes the number of misclassifications in orders of object pairs. This is equivalent to minimizing the Kendall distance between regression order and samples.
OrderSVM separates the objects that ranked lower than *l*-th from the higher ones, and these separations are summed over all ranks *l*.
Consider the case that an object A is ranked 3rd in sample and 5th in regression order. In this case, classifications are failed at two thresholds. That is to say the number of misclassifications equals to absolute difference between ranks. Therefore, OrderSVM can be considered as minimizing the Spearman footrule between regression order and samples.
(Diaconis-Graham inequality: d_Ken+d_Cay ≦ d_Foot ≦ 2 d_Ken)

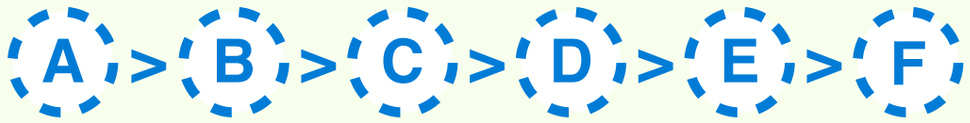
Expected Rank Regression (ERR)

[Kamishima+ 05]

- **permutation noise model = Thurstonian**
- **regression order model = sorting by scores**

expected ranks in a complete order are estimated from samples, and a score function is learned by regression from pairs of expected ranks and feature vectors of all objects

complete order



consisting of all possible objects,
free from permutation noise,
unobserved

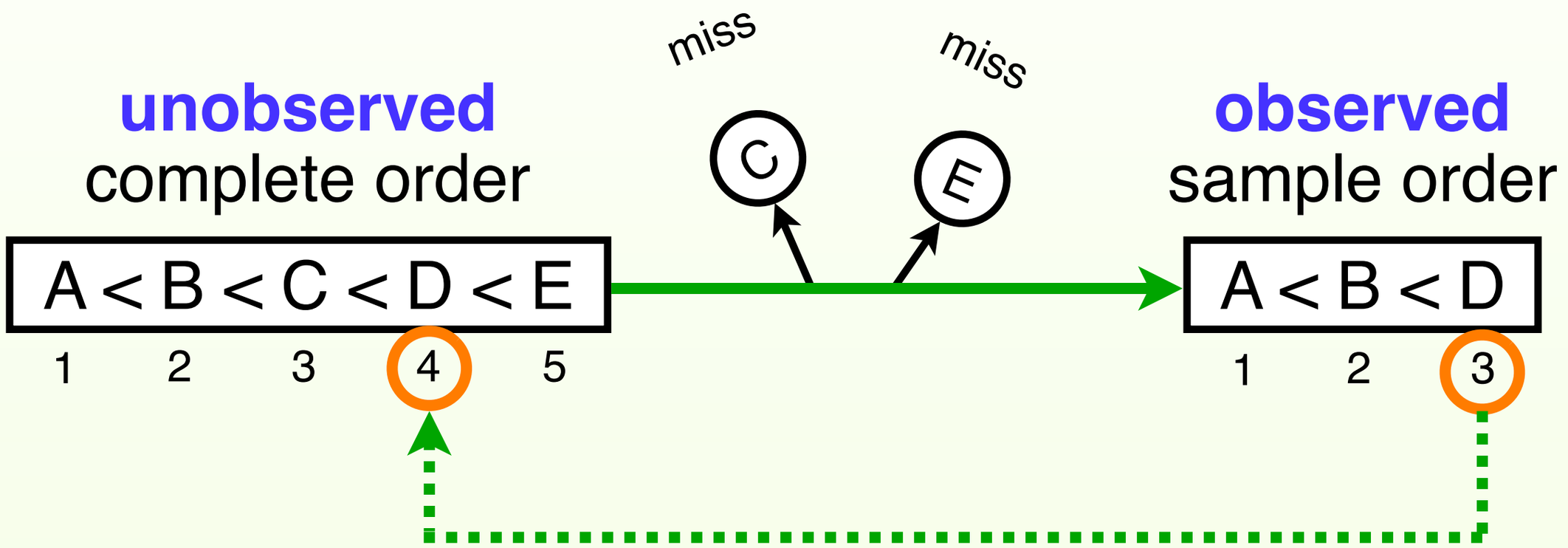
sample order

consisting of sub-sampled objects,
with permutation noise,
observed

Because expected ranks are considered as the location parameters of score distributions, this method is based on Thurstonian model

Our expected rank regression assumes the existence of complete order, which is consisting of all possible objects, free from permutation noise, but unobserved. Expected ranks in a complete order are estimated from samples for all observed objects. A score function is learned by regression from pairs of an expected rank and feature vector of all objects. Because expected ranks are considered as the location parameters of score distributions, this method is based on Thurstonian model.

Expected Rank



$$\text{expectation of ranks in a unobserved complete order} \propto \frac{\text{rank in a observed order}}{(\text{length of a observed order}) + 1}$$

[Arnold+ 92]

To perform ERR, expected ranks have to be computed. To do this, we use simple theorem in a order statistics literature. Objects are selected uniformly at random, and these are missed, then sample orders are observed. In this case, expectation of ranks in a unobserved complete order over all possible missing patterns is proportional to the rank in an observed order divided by the length of a observed order plus one. We adopted this quantity as a expected rank.

ListNet [Cao+ 07]

- **permutation noise model = Multistage**
- **regression order model = sorting by scores**

Straightforward modification of Plackett-Luce model, and parameters are optimized by using neural networks

Parameters of objects are replaced with score functions of object features

sum of scores for the not yet ranked objects

$$\frac{f(\mathbf{x}_i)}{\sum_j f(\mathbf{x}_j)}$$

score for the next ranked object

scores functions, $f(\mathbf{x}_i)$, are linear,
and these weights are estimated by maximum likelihood

ListNet is a straightforward modification of a Plackett-Luce model. Parameters of objects are replaced with functions of object features. The probability of the next ranked object is the score for the next ranked object divided by the sum of scores for the not yet ranked objects. Score functions are linear, and these weights are estmated by maximum likelihood and are optimized by using neural networks.
That covers what we want to say about six object ranking methods.

Absolute / Relative Ranking

absolute ranking function

objects {A,B,C} → **absolute ranking function** → are sorted as: A > B > C

C is replaced with D {A,B,D} → **absolute ranking function** → A must be always ranked higher than B

In other words, either D>A>B, A>D>B, or A>B>D is allowed

relative ranking function

Other than absolute ranking function

- If you know Arrow's impossibility theorem, this is related to its condition I

Finally, we'd like to discuss several properties of object ranking tasks.
We propose notions of absolute and relative ranking, which are the properties that the learned ranking function should have. Absolute ranking function satisfy the following condition. Consider the case that objects A, B, C are sorted as A>B>C by an absolute ranking function. In this case, even if C is replaced with D, A must be always ranked higher than B by the same ranking function. In other words, either D>A>B, A>D>B, or A>B>D is allowed. If a ranking function doesn't satisfy this condition, it is called relative ranking function.
If you know Arrow's impossibility theorem, this is related to its condition I.

Absolute / Relative Ranking

regression order model

sorting by scores

absolute ranking function

linear ordering

relative ranking function

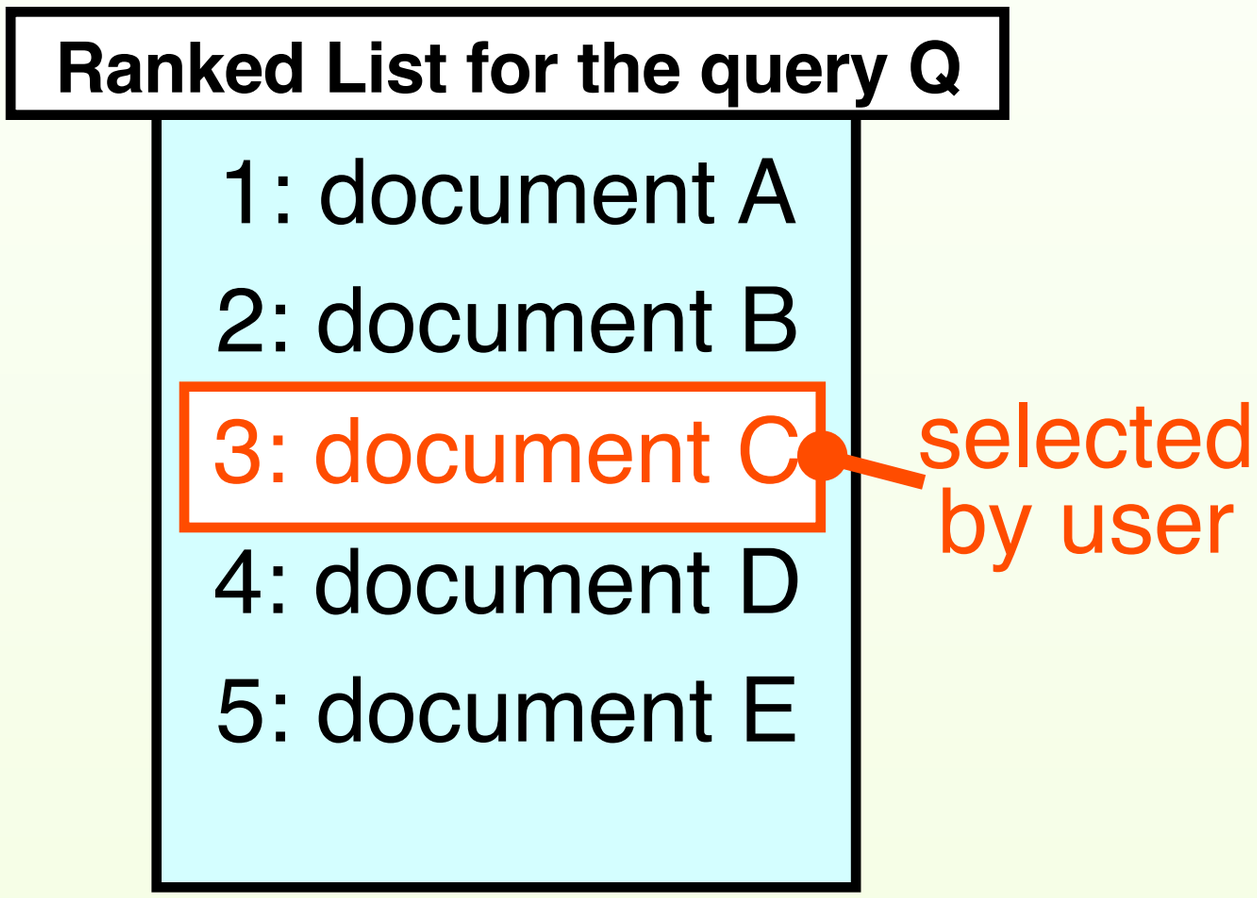
- For IR or recommendation tasks, absolute ranking functions should be learned. For example, the fact that an apple is preferred to an orange is independent from the existence of a banana.
- Only few tasks suited for relative ranking

This property is connected with the regression order model. If an object ranking method adopts a “sorted by scores” model, its ranking function becomes absolute, because the degrees of preference are determined independently. For information retrieval or recommendation tasks, absolute ranking functions should be learned. This is because, for example, the fact that an apple is preferred to an orange is independent from the existence of a banana. On the other hand, only few tasks suited for relative ranking.
In the next two slides, we will show two types of applications.

Relevance Feedbacks

[Joachims 02, Radlinski+ 05]

Leaning from relevance feedback is a typical absolute ranking task



The user scans this list from the top, and selected the third document C.

The user checked the documents A and B, but these are not selected.

This user's behavior implies relevance feedbacks: C>A and C>B.

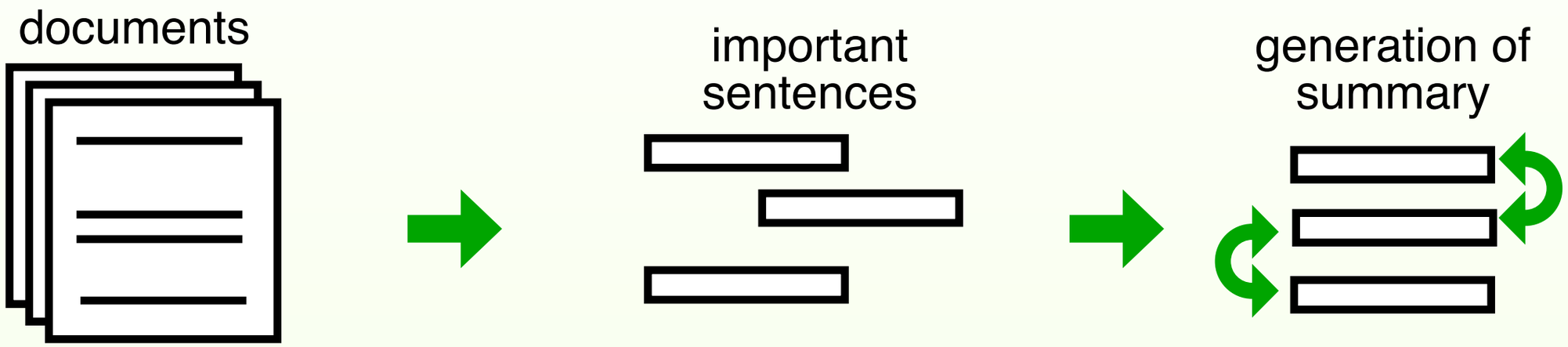
Object ranking methods can be used to update document's relevance based on these feedbacks

Learning from relevance feed back is a typical absolute ranking task. Joachims proposed a method for implicitly obtaining the relevance feedbacks. Given a ranked list for the query Q, the user scans this list from the top, and selected the third document C. The user checked the documents A and B, but these are not selected. This user's behavior implies relevance feedback: The document C is more relevant than the A or B. Object ranking methods can be used to update document's relevance based on these feedbacks.

Multi-Document Summarization

[Bollegala+ 05]

Example of relative ranking task: Multi-Document Summarization (MDS)



Generating summaries is sorting sentences appropriately

↓

From the samples of correctly sorted sentences, object ranking methods learns ranking functions

- features of sentences: chronological info, precedence, relevance among sentences

Appropriate order of sentences are influenced by the relevance to the other sentences or the importance relative to the other sentences

↓

Absolute ranking functions are not appropriate for this task

To my knowledge, this is an only example of relative ranking task: Multi-Document Summarization (MDS). Experimental results are not so good, but I think this is interesting application of object ranking. Given multiple documents, important sentences are picked-up, and a summary is generated from these sentences. Generating summary is sorting sentences appropriately. To achieve this, from the samples of correctly sorted sentences, object ranking methods learns ranking functions.
In this case, appropriate order of sentences are influenced by the relevance to the other sentences or the importance relative to other sentences. Therefore, absolute ranking functions are not appropriate for this task.

Attribute and Order Noise

Order Noise

noiseless order → observed sample

$A \succ B \succ C \rightarrow A \succ C \succ B$

order noise is the permutation in orders

Attribute Noise

objects are represented by attribute vectors

$$x_i = (x_{i1}, \ldots, x_{ik})$$

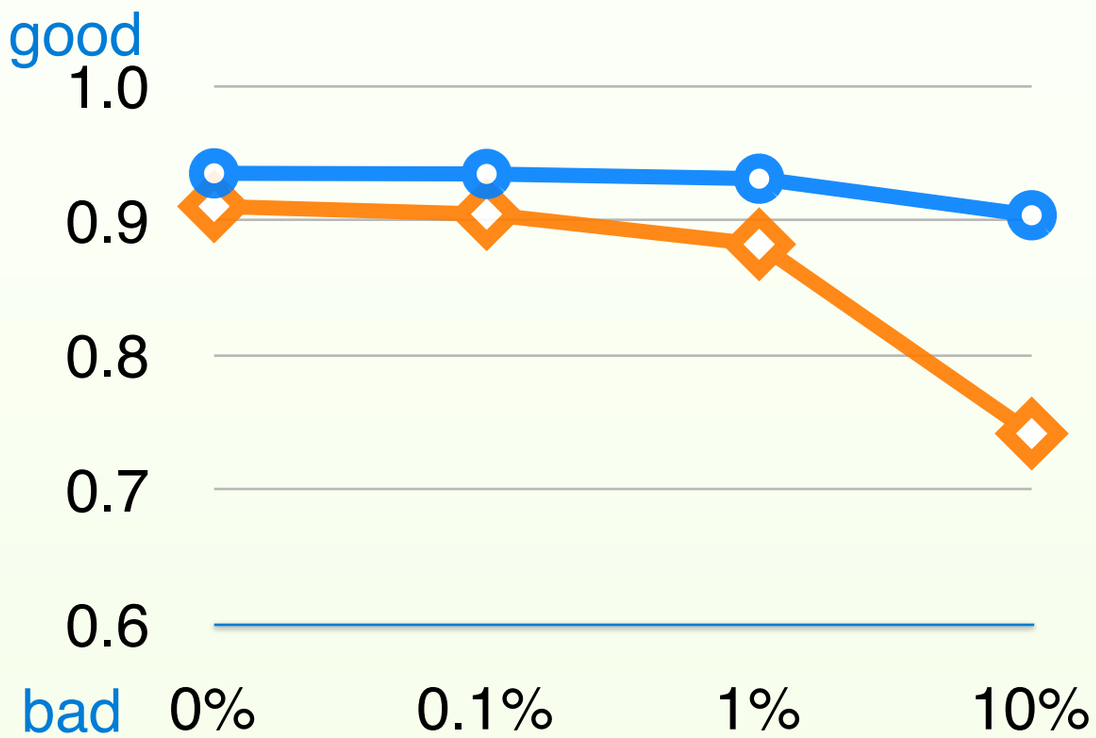
attribute noise is the perturbation in attribute values

Next, we'd like to show interesting experimental results about the difference between SVMs and non-SVMs. We investigated the robustness against two types of noise on synthetic data. One is order noise,which is the permutation in orders. The other is attribute noise, which is the perturbation in attribute values.

Robustness against Noises

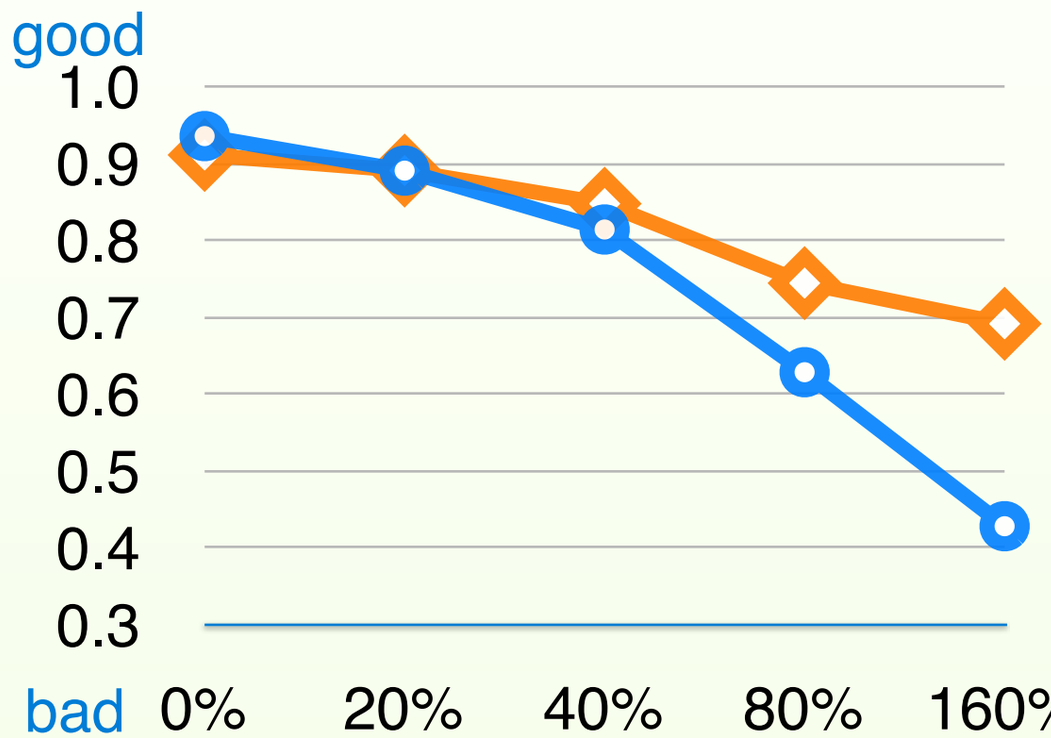
[Kamishima+ 05]

Order Noise



low high

Attribute Noise



low high

ERR SVOR

Vertical: prediction concordance Horizontal: noise level

SVM-based → robust against attribute noise

non-SVM-based → robust against order noise

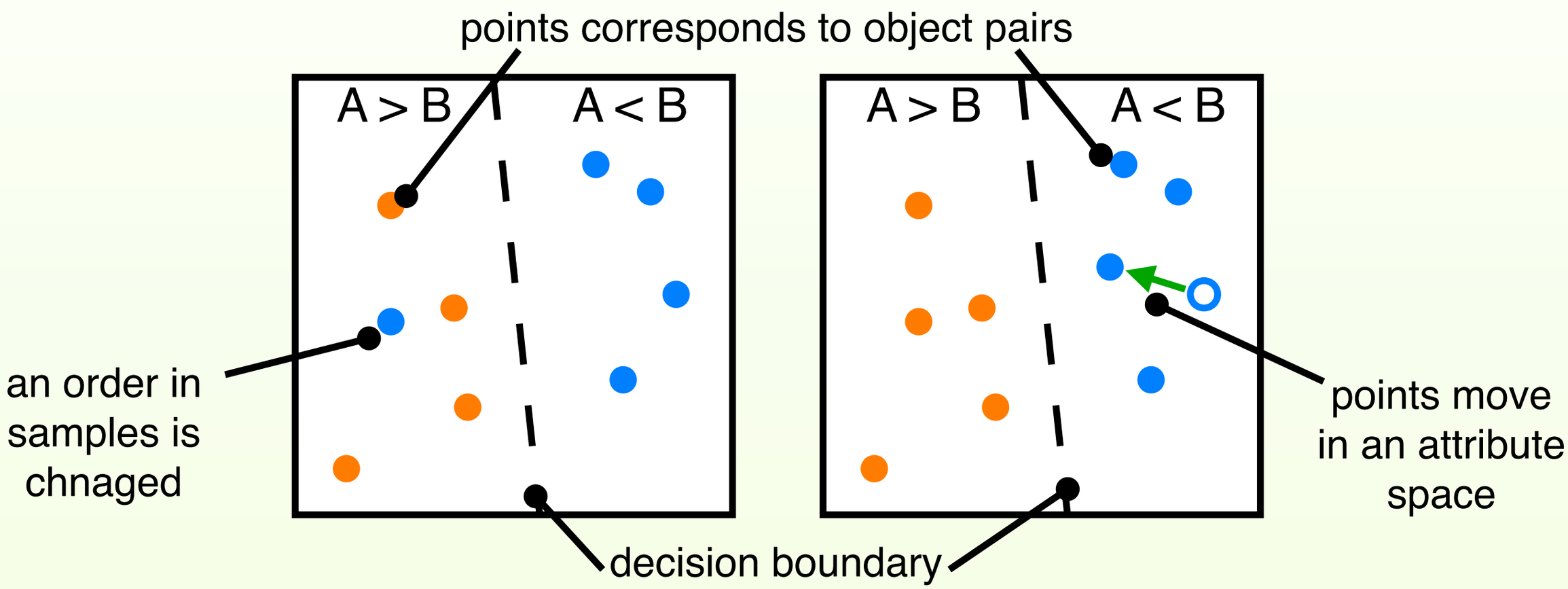
These figures show the variation of prediction concordance in accordance with the noise level. An orange curve is SVM-based results, and the other is non-SVM-based results. These two sets of curves are clearly contrasted. SVM-based methods are robust against attribute noise, but are not robust against order noise. On the other hand, non-SVM-based methods conversely behaves.

SVM-based Cases

SVM-based methods solves object ranking tasks as classification: A>B or A<B

Order Noise

Attribute Noise



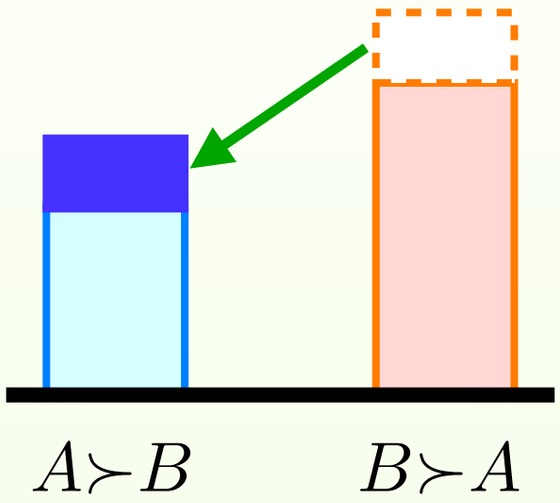
Changed points become support-vectors with high probability, and the results are seriously influened

Slight change in features never influences the results, if changing within decision boundary

We consider that this experimental results can be explained as follows:
SVM-based methods solves object ranking tasks as classification: A precedes to B or A succeeds to B. If orders in samples are permuted, changed points become support-vectors with high probability, and seriously affect the results. On the other hand, slight change in features never influences the results, if changing within decision boundary.

non-SVM-based Cases

Order Noise



samples are moved from B>A to A>B

Results are not influenced, if majority class between these two do not change

Attribute Noise

Any little changes in features influences the loss function, due to the lack of the robustness features like hinge loss of SVMs

We next discuss non-SVM-based cases.
In a case of order noise, if samples are moved from class B>A to A<B, results are not affected, if majority classes between these two do not change. In the other attribute noise case, any little changes in features influences the loss function, due to the lack of the robustness features like hinge loss of SVMs.

Performance of Object Ranking Methods

Accuracy

We compared the prediction accuracies of object ranking methods except for ListNet [Kamishima+ 05]. Though several differences are observed, we think that, like other ML tasks, the appropriate choices for the target task is primally important.

Efficiency

Two SVMs are slow than non-SVMs, and our ERR is fast in almost cases

Powerful linear model

Linear models for ranking functions are more powerful than in standard regression or classification. This is because any monotonic functions are equivalent to linear function as ranking score function.

Conclusion

- define object ranking task and discuss relation with regression and ordinal regression problems
- introduce four types of distributions for rankings: Thurstonian, paired comparison, distance-based, and multistage
- show six methods for object ranking tasks: Cohen's method, RankBoost, SVOR(=RankingSVM), OrderSVM, ERR, and ListNet
- propose the notion of absolute and relative ranking tasks
- discuss about the prediction accuracy of object ranking methods

SUSHI data: preference in sushi surveyed by ranking method
http://www.kamishima.net/sushi/

Bibliography

- [Agresti 96] A.Agresti, "Categorical Data Analysis", John Wiley & Sons, 2nd eds. (1996)
- [Arnold+ 92] B.C.Arnold et. al. "A First Course in Order Statistics", John Wiley & Sons, Inc. (1992)
- [Babington Smith 50] B.Babington Smith, "Discussion on Professor Ross's Paper", JRSS (B), vol.12 (1950)
- [Bell+ 07] R.M.Bell & Y.Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights", ICDM2007
- [Bladley+ 52] R.A.Bradley & M.E.Terry, "Rank Analysis of Incomplete Block Designs — I. The Method of Paired Comparisons", Biometrika, vol.39 (1952)
- [Bollegala+ 05] D.Bollegala et. al. "A Machine learning Approach to Sentence Ordering for Multidocument Summarization and its Evaluation", IJCNLP2005

Bibliography

- [Cao+ 07] Z.Cao et. al. "Learning to Rank: From Pairwise Approach to Listwise Approach" ICML2007
- [Cohen+ 99] W.W.Cohen et. al "Learning to Order Things", JAIR, vol. 10 (1999)[Cosley+ 03] D.Cosley et. al. "Is Seeing Believing? How Recommender Interfaces Affect Users' Opnions", SIGCHI 2003
- [Critchlow+ 91] D.E.Critchlow et. al. "Probability Models on Rankings", J. of Math. Psychology, vol.35 (1991)
- [Freund+ 03] Y.Freund et. al. "An Efficient Boosting Algorithm for Combining Preferences", JMLR, vol.4 (2003)
- [Grötschel+ 84] M.Grötschel et. al. "A Cutting Plane Algorithm for the Linear Ordering Problem", Operations Research, vol.32 (1984)
- [Herbrich+ 98] R.Herbrich et. al. "Learning Preference Relations for Information Retrieval", ICML1998 Workshop: Text Categorization and Machine Learning

Bibliography

- [Herlocker+ 99] J.L.Herlocker et. al. "An Algorithmic Framework for Performing Collaborative Filtering", SIGIR1999
- [Joachims 02] T.Joachims, "Optimizing Search Engines Using Clickthrough Data", KDD2002
- [Kamishima 03] T.Kamishima, "Nantonac Collaborative Filtering: Recommendation Based on Order Responses", KDD2003
- [Kamishima+ 05] T.Kamishima et. al. "Supervised Ordering — An Empirical Survey", ICDM2005
- [Kamishima+ 06] T.Kamishima et. al., "Nantonac Collaborative Filtering — Recommendation Based on Multiple Order Responses", DMSS Workshop 2006
- [Kazawa+ 05] H.Kazawa et. al. "Order SVM: a kernel method for order learning based on generalized order statistics", Systems and Computers in Japan, vol.36 (2005)

Bibliography

- [Mallows 57] C.L.Mallows, "Non-Null Ranking Models. I", Biometrika, vol.44 (1957)
- [Marden 95] J.I.Marden "Analyzing and Modeling Rank Data", Chapman & Hall (1995)
- [McCullagh 80] P.McCullagh, "Regression Models for Ordinal Data", JRSS(B), vol.42 (1980)
- [Thurstone 27] L.L.Thurstone "A Law of Comparative Judgment", Psychological Review, vol.34 (1927)
- [Plackett 75] R.L.Plackett, "The Analysis of Permutations", JRSS (C), vol.24 (1975)
- [Radlinski+ 05] F.Radlinski & T.Joachims, "Query Chains: Learning to Rank from Implicit Feedback", KDD2005