Greedy function optimization in learning to rank

Andrey Gulin, Pavel Karpovich

Petrozavodsk
2009

Annotation

Greedy function approximation and boosting algorithms are well suited for solving practical machine learning tasks. We will describe well-known boosting algorithms and their modifications used for solving learning to rank problems.

Content

- Search engine ranking.
  - Evaluation measures.
  - Feature based ranking model.
  - Learning to rank. Optimization problems(listwise, poitnwise, pairwise approaches).
- Pointwise approach. Boosting algorithms and greedy function approximation.
- Modification MatrixNet.
- Listwise approach. Approximations of complex evaluation measures(DCG, nDCG).

Search engine ranking

Main goal: to rank documents according to their quality of conformance to the search query.

How to evaluate ranking?
Prerequisites:

- Set of search queries $Q = \{q_1, .., q_n\}$.
- Set of documents corresponding to each query $q \in Q$.

$$q \rightarrow \{d_1, d_2, ...\}$$

- Relevance judgments for each pair $(query, document)$
(In our model real numbers $rel(q, d) \in [0, 1]$)

Evaluation measures

Evaluation mark for ranking will be an average value of **evaluation measure** over the set of search queries Q:

$$\frac{\sum\limits_{q \in Q} EvMeas(ranking\ for\ query\ q)}{n}$$

Example of evaluation measure $EvMeas$:

- **Precision-10** - *percent of documents with relevance judgments greater than 0 in top-10*

Evaluation measures

Evaluation mark for ranking will be an average value of **evaluation measure** over the set of search queries Q:

$$\frac{\sum_{q \in Q} EvMeas(ranking\ for\ query\ q)}{n}$$

Example of evaluation measure $EvMeas$:

- **Precision-10** - *percent of documents with relevance judgments greater than 0 in top-10*

Evaluation measures

- **MAP** - mean average precision

$$MAP(ranking\ for\ query\ q) = \frac{1}{k}\sum_{i=1}^{k}\frac{i}{n_r(i)}$$

k - number of documents with positive relevance judgments corresponding to query q, $n_r(i)$ - position of the i-th document with relevance judgment greater than 0.

Evaluation measures

- **DCG - discounted cumulative gain**

$$DCG(ranking\ for\ query\ q) = \sum_{j=1}^{N_q} \frac{rel_j}{log_2 j + 1}$$

N_q - total number of documents in ranked list, rel_j - relevance judgment for document on position j.

- **normalized DCG(nDCG)**

$$nDCG(...) = \frac{DCG(ranking\ for\ query\ q)}{DCG(ideal\ ranking\ for\ query\ q)}$$

Feature based ranking model

- Each pair $(query, document)$ is described by the vector of features.

$$(q, d) \rightarrow (f_1(q, d), f_2(q, d), ..)$$

- Search ranking is the sorting by the value of **"relevance function"**. ***Relevance function*** is a combination of features:

$$fr(q, d) = 3.14 \cdot log_7(f_9(q, d)) + e^{f_{66}(q,d)} + \ldots$$

Feature based ranking model

- Each pair $(query, document)$ is described by the vector of features.

$$(q, d) \rightarrow (f_1(q, d), f_2(q, d), ..)$$

- Search ranking is the sorting by the value of **"relevance function"**. ***Relevance function*** is a combination of features:

$$fr(q, d) = 3.14 \cdot log_7(f_9(q, d)) + e^{f_{66}(q,d)} + ...$$

Optimization problems

How to get a good relevance function?

Get **learning set** of examples P_l - set of pairs (q, d) with relevance judgments $rel(q, d)$.

Use learning to rank methods to obtain fr.

Optimization problems (listwise approach)

- Solve direct optimization problem:

$$\arg\max_{fr \in F} = \frac{\sum\limits_{q \in Q_l} EvMeas(ranking\ for\ query\ q\ with\ fr)}{n}$$

F - set of possible ranking functions. Q_l - set of different queries in learning set P_l

Difficulty in solving: most of evaluation measures are non-continuous functions.

Optimization problems (pointwise approach)

- Simplify optimization task to regression problem and minimize sum of loss functions:

$$\arg\min_{fr \in F} L_t(fr) = \frac{\sum_{(q,d)\in P_l} L(fr(q,d), rel(q,d))}{n}$$

$L(fr(q,d), rel(q,d))$ - loss function, F - set of possible ranking functions. Examples of loss functions:

  - $L(fr, rel) = (fr - rel)^2$
  - $L(fr, rel) = |fr - rel|$

Optimization problem (pairwise approach)

- Try to use well-known machine learning algorithms to solve the following classification problem:
  - an ordered pair of documents (d_1, d_2)(corresponding to query q) belongs to first class iff $rel(q, d_1) > rel(q, d_2)$
  - an ordered pair of documents (d_1, d_2)(corresponding to query q) belongs to second class iff $rel(q, d_1) \leq rel(q, d_2)$

Boosting algorithms and greedy function approximation

We will solve regression problem:

$$\arg\min_{fr \in F} \frac{\sum\limits_{(q,d)\in P_l} L(fr(q,d), rel(q,d))}{n}$$

We will search relevance function in the following form:

$$fr(q,d) = \sum_{k=1}^{M} \alpha_k h_k(q,d)$$

Relevance function will be a linear combination of functions $h_k(q,d)$, functions $h_k(q,d)$ belong to simple family H (weak learners family) .

Boosting algorithms and greedy function approximation

We will solve regression problem:

$$\arg\min_{fr \in F} \frac{\sum_{(q,d) \in P_l} L(fr(q,d), rel(q,d))}{n}$$

We will search relevance function in the following form:

$$fr(q,d) = \sum_{k=1}^{M} \alpha_k h_k(q,d)$$

Relevance function will be a linear combination of functions $h_k(q,d)$, *functions* $h_k(q,d)$ *belong to simple family* H *(weak learners family)* .

Boosting algorithms and greedy function approximation

We will construct final function by iterations. On each iteration we will add an additional term $\alpha_k h_k(q,d)$ to our relevance function:

$$fr_k(q,d) = fr_{k-1}(q,d) + \alpha_k h_k(q,d)$$

Values of parameter α_k and weak learner $h_k(q,d)$ can be a solution of natural optimization task:

$$\arg\min_{\alpha, h(q,d)} \frac{\sum_{(q,d)\in P_l} L(fr_{k-1}(q,d) + \alpha h(q,d), rel(q,d))}{n}$$

This problem can be solved directly for quadratic loss function and simple classes H, but it can be very difficult to solve for other loss functions.

Boosting algorithms and greedy function approximation

We will construct final function by iterations. On each iteration we will add an additional term $\alpha_k h_k(q,d)$ to our relevance function:

$$fr_k(q,d) = fr_{k-1}(q,d) + \alpha_k h_k(q,d)$$

Values of parameter α_k and weak learner $h_k(q,d)$ can be a solution of natural optimization task:

$$\arg\min_{\alpha, h(q,d)} \frac{\sum_{(q,d)\in P_l} L(fr_{k-1}(q,d) + \alpha h(q,d), rel(q,d))}{n}$$

This problem can be solved directly for quadratic loss function and simple classes H, but it can be very difficult to solve for other loss functions.

Boosting algorithms and greedy function approximation

We will construct additional term $\alpha_k h_k(q,d)$ in three steps :

- **Gradient approximation.** Consider relevance function fr like vector of values indexed by learning examples. Get gradient vector $g = \{g_{(q,d)}\}_{(q,d)\in P_l}$ for error function :

$$g_{(q,d)} = \left[\frac{\partial L_t(fr)}{\partial fr(q,d)}\right]_{fr=fr_{k-1}}$$

- **Weak learner selection**(up to a constant). Find most highly correlated with g function $h_k(q,d)$ by solving the following optimization task:

$$\arg \min_{\beta, h(q,d)\in H} \sum_{(q,d)\in P_l} (g_{(q,d)} - \beta h(q,d))^2$$

Boosting algorithms and greedy function approximation

We will construct additional term $\alpha_k h_k(q,d)$ in three steps :

- **Gradient approximation.** Consider relevance function fr like vector of values indexed by learning examples. Get gradient vector $g = \{g_{(q,d)}\}_{(q,d)\in P_l}$ for error function :

$$g_{(q,d)} = \left[\frac{\partial L_t(fr)}{\partial fr(q,d)}\right]_{fr=fr_{k-1}}$$

- **Weak learner selection**(up to a constant). Find most highly correlated with g function $h_k(q,d)$ by solving the following optimization task:

$$\arg \min_{\beta, h(q,d)\in H} \sum_{(q,d)\in P_l} (g_{(q,d)} - \beta h(q,d))^2$$

Boosting algorithms and greedy function approximation

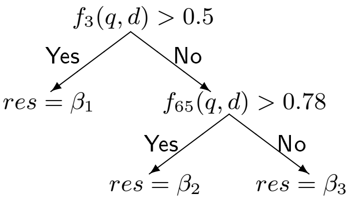
- **Selection of α_k.** Find the value of α_k from one-parameter optimization problem:

$$\arg\min_{\alpha} \frac{\sum_{(q,d)\in P_l} L(fr_{k-1}(q,d) + \alpha h_k(q,d), rel(q,d))}{n}$$

Iterate... Iterate... Iterate...

Weak learner selection

Let our class of weak learners H will be a set of decision-tree functions:



Example of 3-region decision-tree function. The function splits feature space on 3 regions by conditions in the form $f_j(q,d) > \alpha$ (f_j - split feature, α - split bound). It has a constant value for feature vectors in one region.

Weak learner selection (function values)

Our weak learners family will be 6-region(example, const-regions) decision-tree functions. We will try to solve:

$$\arg\min_{h(q,d)\in H} \sum_{(q,d)\in P_l} (g_{(q,d)} - \beta h(q,d))^2$$

Suppose we know tree-structure of weak learner $h(q,d)$ - we know split conditions and regions. We should find "region constant values". Optimization problem reduces to ordinary regression problem:

$$\arg\min_{h(q,d)\in H,\beta} \sum_{(q,d)\in P_l} (g_{(q,d)} - \beta\beta_{ind(q,d)})^2$$

$ind(q,d)$ - number of region, which contains features vector for pair (q,d) ($ind(q,d) \in \{1,..,6\}$).

Weak learner selection (function values)

Our weak learners family will be 6-region(example, const-regions) decision-tree functions. We will try to solve:

$$\arg \min_{h(q,d) \in H} \sum_{(q,d) \in P_l} (g_{(q,d)} - \beta h(q,d))^2$$

Suppose we know tree-structure of weak learner $h(q,d)$ - we know split conditions and regions. We should find "region constant values". Optimization problem reduces to ordinary regression problem:

$$\arg \min_{h(q,d) \in H, \beta} \sum_{(q,d) \in P_l} (g_{(q,d)} - \beta \beta_{ind(q,d)})^2$$

$ind(q,d)$ - number of region, which contains features vector for pair (q,d) ($ind(q,d) \in \{1,..,6\}$).

Weak learner selection (tree structure)

Greedy tree selection:

- $bestTree$ = constant function (1-region tree).
- **Greedy split**. Try to split regions of $bestTree$ and find the best split.

Suppose we have constant set of possible split bounds. Number of possible splits is bounded by the value:

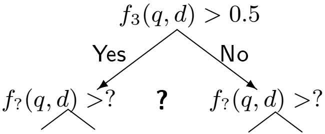
$$\#\{regions\} \cdot \#\{features\} \cdot \#\{split\ bounds\}$$

- Repeat previous step.

Weak learner selection (tree structure)

Greedy tree selection:

- $bestTree$ = constant function (1-region tree).
- **Greedy split**. Try to split regions of $bestTree$ and find the best split.



Suppose we have constant set of possible split bounds. Number of possible splits is bounded by the value:

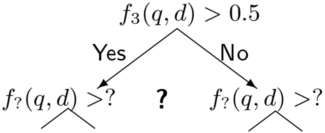
$$\#\{regions\} \cdot \#\{features\} \cdot \#\{split\ bounds\}$$

- Repeat previous step.

Weak learner selection (tree structure)

Greedy tree selection:

- $bestTree$ = constant function (1-region tree).
- **Greedy split**. Try to split regions of $bestTree$ and find the best split.



Suppose we have constant set of possible split bounds. Number of possible splits is bounded by the value:

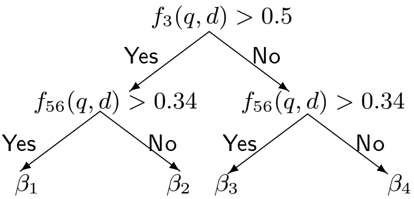
$$\#\{regions\} \cdot \#\{features\} \cdot \#\{split\ bounds\}$$

- Repeat previous step.

MatrixNet

Weak learners set- full decision trees with depth k and 2^k regions.

- Constant number of layers (constant depth).
- The same split conditions for one layer.



We don't need complex structure: depth is the main thing.

MatrixNet

Internet Mathematics 2009

Яндекс

company → internet-mathematics

Search

t 2009

Datasets

on

lution

ard

d Conditions

Leaderboard

The table shows both final contest results (May 15, 2009) and new results. Read more about the contest task and evalu
Datasets section.

MatrixNet

Team	Last upload time	Number of trials	Last result (public evaluation)	Final result
Joker	05.09.2009 (05:07 GMT+03)	2	4.283317	*4.151528*
Euclid	24.08.2009 (09:12 GMT+03)	30	4.280853	*4.149605*
alexeigor	07.05.2009 (17:02 GMT+03)	118	4.280676	*4.141230*
MysteriousGuest	24.08.2009 (12:33 GMT+03)	1	4.279174	*4.143886*
Победа	17.03.2009 (16:25 GMT+03)	3	4.276001	*4.139854*
ACGT	15.05.2009 (14:03 GMT+03)	21	4.274666	*4.128807*
WoodWeb	22.04.2009 (23:09 GMT+03)	12	4.267894	*4.127512*
Nordic	15.05.2009 (23:37 GMT+03)	4	4.266904	*3.857102*
stohastic	15.05.2009 (23:43 GMT+03)	176	4.266712	*4.118830*
Test	15.05.2009 (23:45 GMT+03)	58	4.264024	*3.859052*
ZENIT	15.05.2009 (23:20 GMT+03)	206	4.259964	*4.117877*
Euclid	08.05.2009 (21:46 GMT+03)	40	4.257802	*4.122558*

Approximation of complex evaluation measures (DCG)

Change ranking to "probability ranking". Approximation of DCG for query q, set of documents $\{d_1, .., d_n\}$, and ranking function $fr(q, d)$:

$$apxDCG = \sum_{r \in all\ permutations\ of\ docs} P(fr, r) DCG(r)$$

$P(fr, r)$ - probability to get ranking r in Luce-Plackett model.

$DCG(r)$ - DCG score for permuation r.

Luce-Plackett model

We have set of documents $\{d_1, .., d_n\}$ and set of relevances $\{fr(q, d_1), .., fr(q, d_n)\}$ corresponding them.

Process of ranking selection in Luce-Plackett model:

- Select document for first position. Probability of selection of document d_i is equal to $\frac{fr(q,d_i)}{\sum\limits_{i=1}^{n} fr(q,d_i)}$. Suppose we select document d_x.
- Select document for second position from the rest. Probability of selection of document d_i is equal to $\frac{fr(q,d_i)}{\sum\limits_{i=1}^{n} fr(q,d_i) - fr(q,d_x)}$
- ...

For each selection, if two documents d_i and d_j take part in it, ratio between their selection probabilities should be equeal to the value $\frac{fr(q,d_i)}{fr(q,d_j)}$

Luce-Plackett model

$\{\acute{d}_1, .., \acute{d}_n\}$ - some permutation of $\{d_1, .., d_n\}$

$$P(fr, \{\acute{d}_1, .., \acute{d}_n\}) = \prod_{j=1}^{n} \frac{fr(q, \acute{d}_j)}{\sum\limits_{k=j}^{n} fr(q, \acute{d}_k)}$$

The end. Thank you.

- Tie-Yan Liu. Learning to Rank for Information Retrieval. Tutorial on WWW2008.
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189-1232.
- Friedman, J. H. (1999). Stochastic gradient boosting (Tech. Rep.). Palo. Alto, CA: Stanford University, Statistics Department.
- Plackett, R. L. (1975). The analysis of permutations. Applied Statistics, 24, 193-202