CS 589 Fall 2020

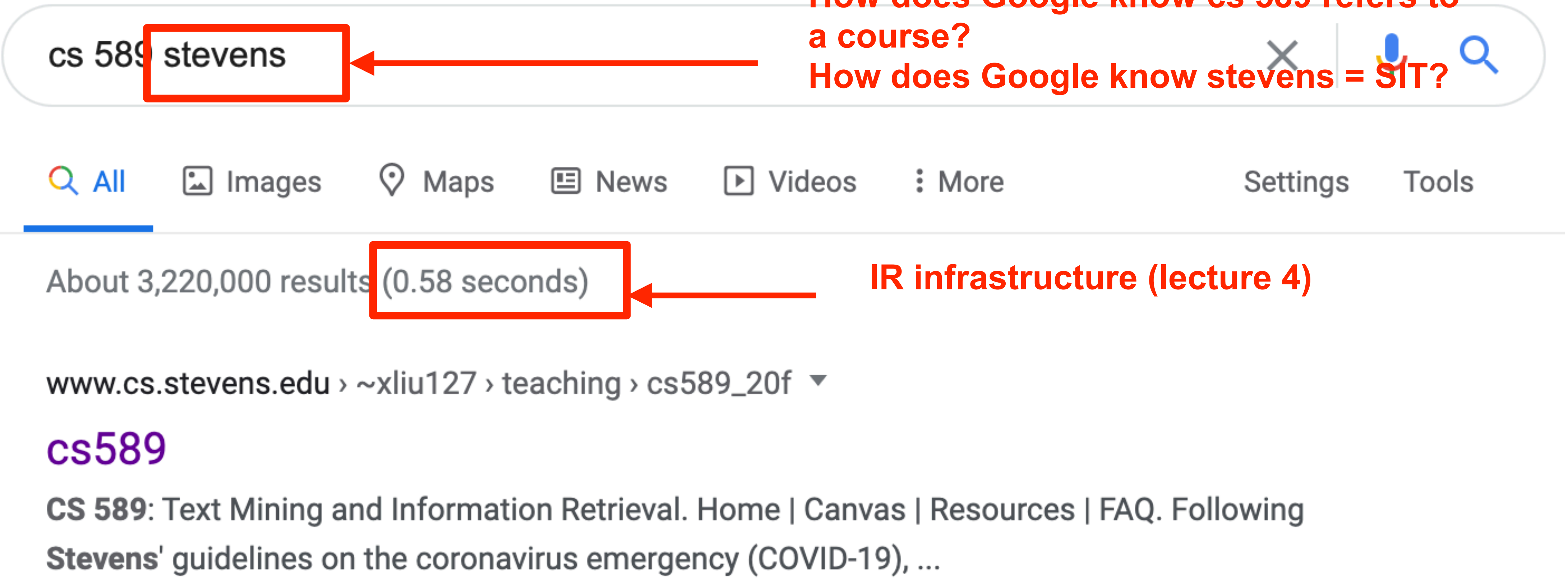
Learning to rank

web search

Instructor: Susan Liu
TA: Huihui Liu

Stevens Institute of Technology

Review of what we've learned



How does Google know cs 589 refers to a course?
How does Google know stevens = SIT?

IR infrastructure (lecture 4)

evaluating accuracy (lecture 3)

Recap of retrieval models

- In Lecture 2, we learned how to measure the similarity between a query and a document

$$score(q,d) = \frac{q \cdot d}{\|q\| \cdot \|d\|}$$

$$score^{BM25}(q,d) = \sum_{w_i \in q} \log \frac{N}{df_i} \times \frac{tf_i\,(k_1+1)}{k_1\left(1-b+b\frac{|dl|}{|avgdl|}\right)+tf_i}$$

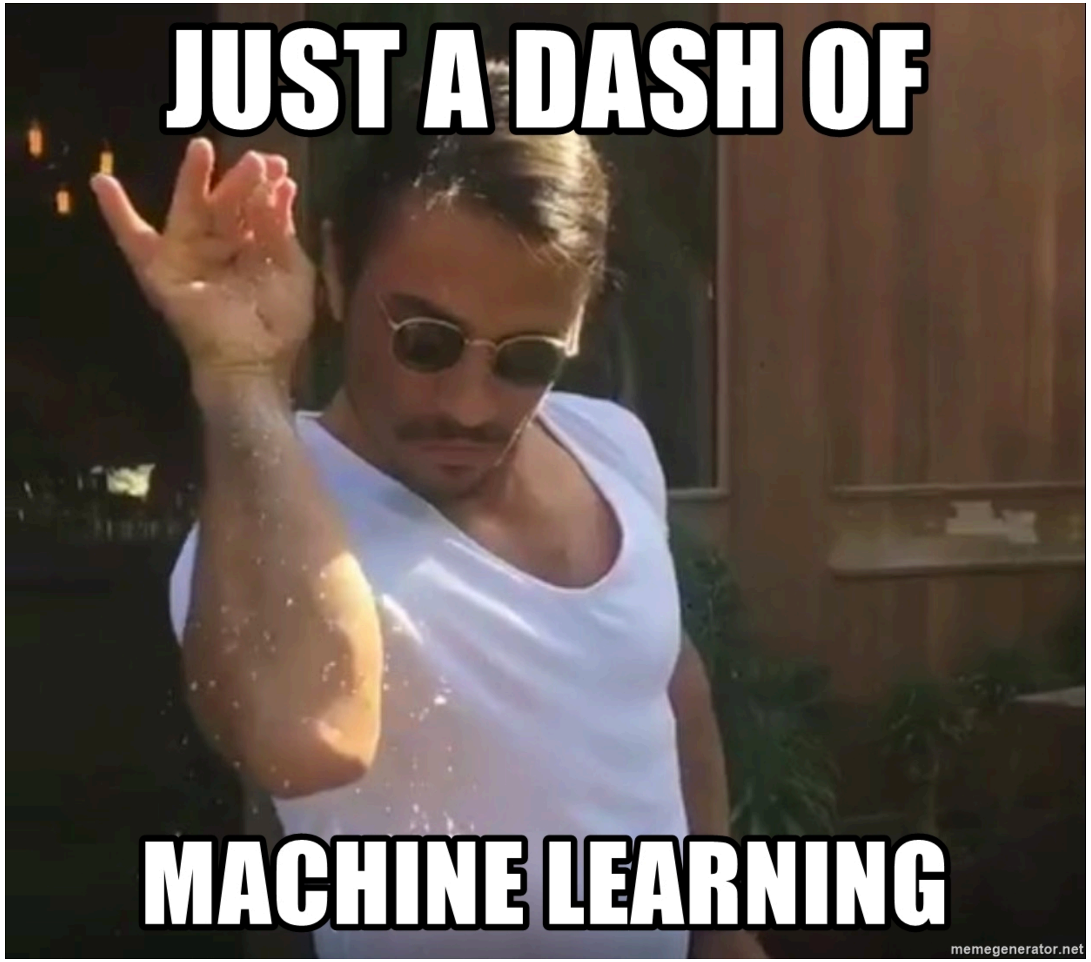
$$score^{LM}(q,d) \overset{rank}{=} \sum_{w_i, w_i \in d} c(w_i,q)\log\frac{p_{seen}(w_i|d)}{\alpha_d p(w_i|C)} + |q|\log\alpha_d$$

The disadvantage of retrieval models

- Cannot adapt to users' fine-grained intents
  - e.g., adapting to certain context (location, demographic information)
  - no personalization
- Cannot naturally leverage the massive amount of user feedback signals
- Formulation is complicated, difficult to tune parameter, e.g., the two-Poisson model
- Difficulty choosing a retrieval model

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

Today's lecture



Today's lecture

- Web search
  - User clicks as implicit feedback
  - Search engine position bias

- Learning to rank
  - Pointwise learning to rank
  - Pairwise learning to rank
  - Listwise learning to rank

- Gradient boosting decision/regression tree (GBDT/GBRT)

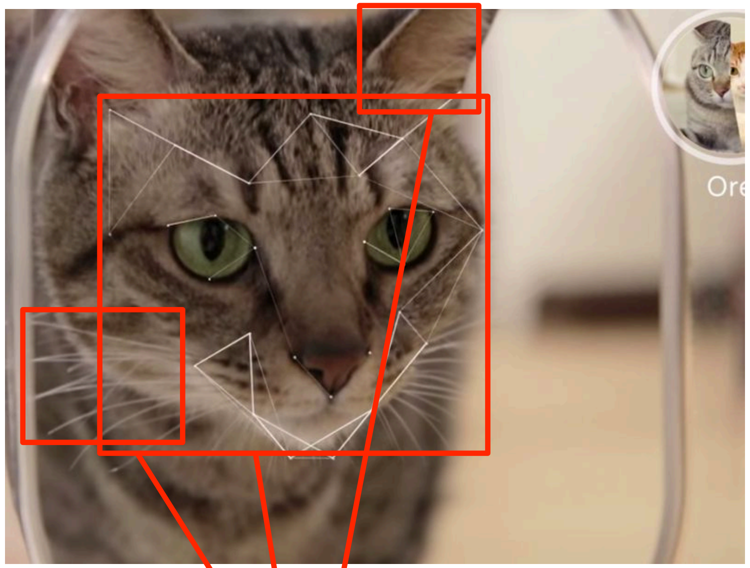
What is machine learning?

	A	B	C	D
1	Country	Age	Salary	Purchased
2	France	44	72000	0
3	Spain	27	48000	1
4	Germany	30	54000	0
5	Spain	38	61000	0
6	Germany	40	1000	1
7	France	35	58000	1
8	Spain	78	52000	0
9	France	48	79000	1
10	Germany	50	83000	0
11	France	37	67000	1

- Machine learning
  - Decision tree
  - Naive bayes
  - logistic regression
  - …

$$y \sim w^T \mathbf{x} + b$$

Learning to rank



Objective: cat vs. dog

Facial feature: [**0.3, -0.2, 0.5**, …]

TF-IDF score = 0.5

BM25 score = 0.35

LM score = 0.3

context, e.g., location

? ← **Objective function: Evaluation metric, e.g., NDCG**

Learning to rank

	• Machine learning	• Learning to *rank*
input:	$(x_1, y_1), \cdots, (x_n, y_n)$	$((q_1, d_1), y_1), \cdots, ((q_n, d_n), y_n)$
learning:	$f = arg \max_{f'} O(f'(x), y)$	$f = arg \max_{f'} O(f'(q, d), y)$
loss function:	accuracy, square loss, hinge loss	P@k, MAP, NDCG

Learning to rank

- An important idea in the past decade of IR community
  - Deployed in industry search engines
  - Yahoo! learning to rank challenge [2011]

- Why does it take so long?
  - Limited data access (search engine, mobile devices was popular only in the last 1-2 decades, data privacy problem)
  - It was possible to tune traditional IR models by hand

Learning to rank

- Feature engineering in modern search engines
  - 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

Learning to rank

- Pointwise
  - Fit the absolute labels individually
  - e.g., A. Shashua and A. Levin, NIPS 2002

- Pairwise
  - Fit the relative order
  - e.g., RankSVM

- Listwise
  - Fit the metric of the entire ranked list
  - e.g., LambdaMART, XGBoost

Pointwise learning to rank = Regression

- Reducing the ranking problem to

Regression:

$$O(f'(q,d),y) = -\sum_i (y_i - f(q_i,d_i))^2$$

Classification:

$$O(f'(q,d),y) = \sum_i \sigma(f(q_i,d_i) = y_i)$$

Shashua et al. *Ranking with large margin principle.* NIPS 2002

Cosssock et al. *Subset ranking using regression.* COLT 2006

Pointwise learning to rank = Regression

- Collect a training corpus of (q,d,r) triples

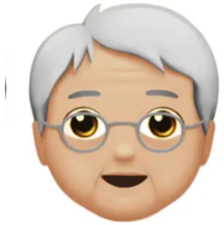
$$score(q,d) = w^T \times [cosine, bm25, \quad \cdots] + b$$

$$\min_{w,b} \sum_{(q,d,r)} (r - score(q,d))^2$$

exampleID	**query ID**	**doc ID**	**cosine**	**bm25**	**span length**	**…**	**relevance**
1	0	0	0.032	0.004	3		0
2	0	1	0.02	0.022	4		1
3	0	2	0.043	0.03	2		0
4	1	0	0.027	0.028	3		1
5	1	3	0.009	0.328	2		1
6	1	4	0.04	0.001	5		0

Ranking is easier than regression

★★★★ ★★★



★★★ ★★

Pointwise -> pairwise learning to rank

Pointwise learning to rank:

$$score(q,d) = w^T \times [cosine, bm25, w, \cdots] + b$$

$$\min_{w,b} \sum_{(q,d,r)} (r - score(q,d))^2$$

Pairwise learning to rank (example):

$$s_i = wx_i + b \qquad P(d_i \succ d_j) = \frac{1}{1+e^{-\sigma(s_i - s_j)}}$$

$$\min_{\Theta} \sum_{i,j} -\mathbb{1}[r_i > r_j] \log P(d_i \succ d_j) - (1 - \mathbb{1}[r_i \succ r_j]) \log (1 - P(d_i \succ d_j))$$

Ranking based on machine learning algorithms

- SVM^{Rank} (Joachims et al. 2002)
    - Ranking algorithm based on support vector machine
- Neural network: RankNet (Burges et al. 2006)
- Tree ensemble
    - Random forests (Breiman and Schapire)
    - Multi additive regression trees (Friedman, 1999)
    - Gradient boosted decision tree (Burges 2010)

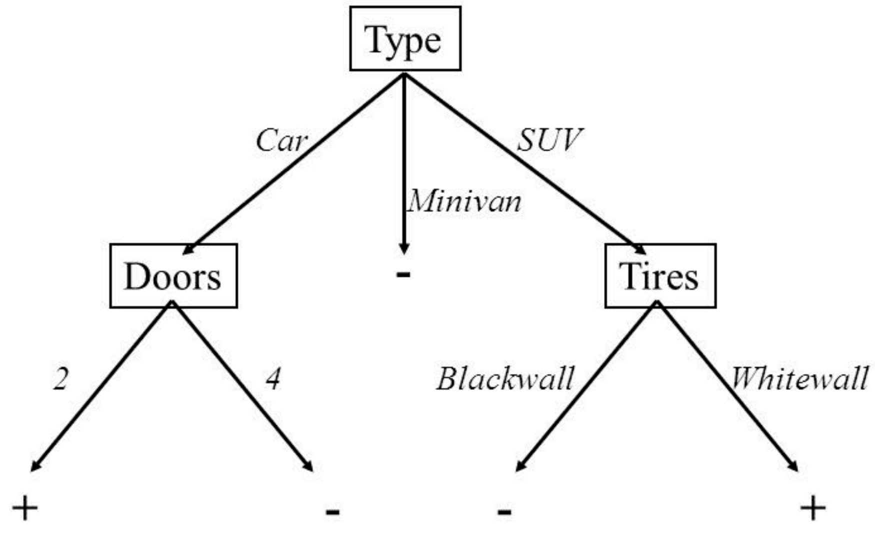
Yahoo! learning to rank challenges

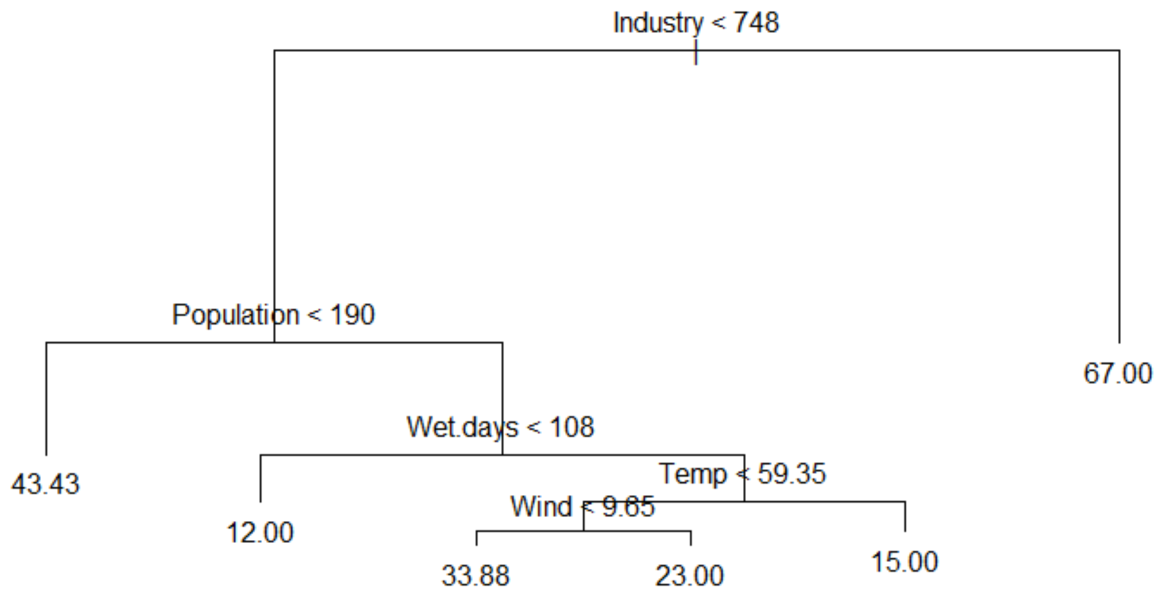
- Yahoo! Webscope dataset : 36,251 queries, 883k documents, 700 features, 5 ranking levels
    - Ratings: Perfect (navigational), Excellent, Good, Fair, Bad
- **LambdaMART** (Burges et al.) was the linear combination of 12 models:
    - 8 Tree Ensembles (LambdaMART)
    - 2 LambdaRank Neural Nets
    - 2 MART models using logistic regression loss

Burges et al. *Learning to Rank Using an Ensemble of Lambda-Gradient Models.*

Regression tree

- Regression tree vs decision tree

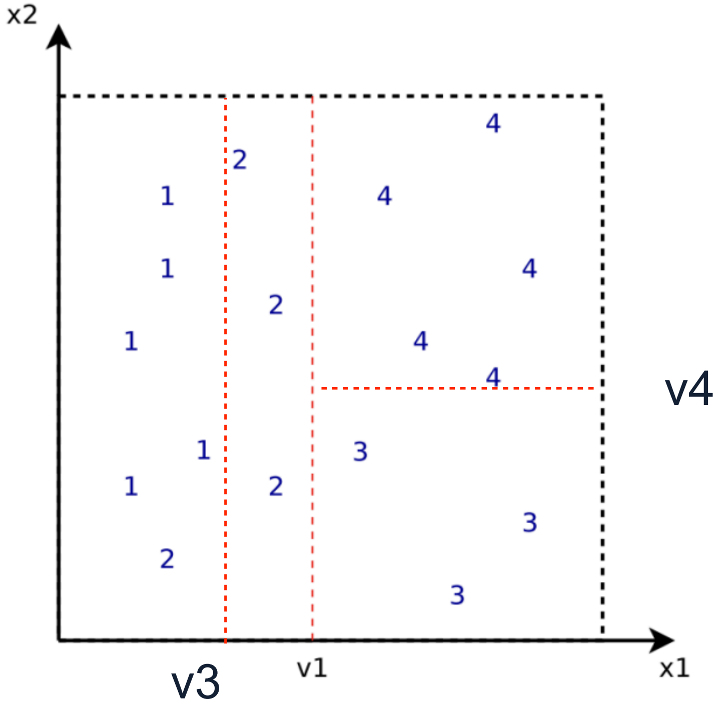




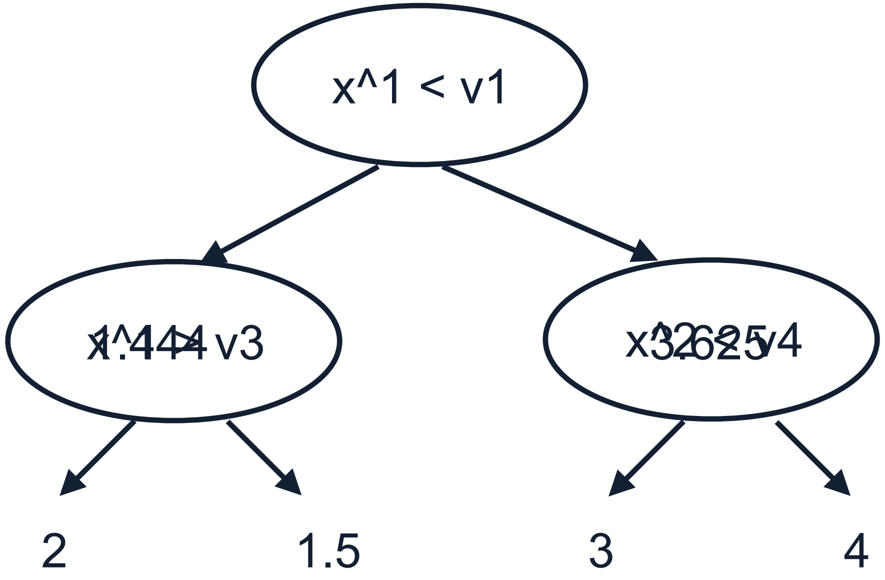
decision tree

regression tree

Regression tree



$$\min_{\Theta} \sum_i (y_i - f(x_i; \Theta))^2$$



slides from Stanford CS276

Boosting in machine learning

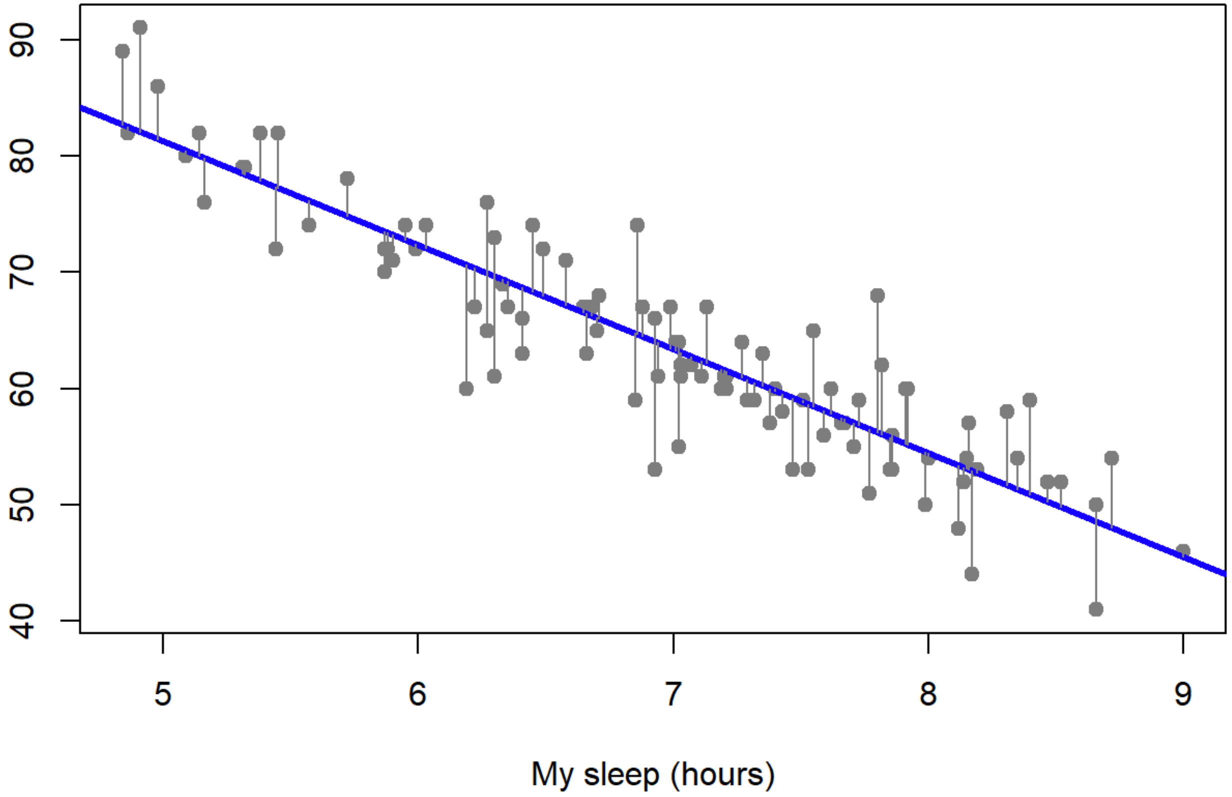
- **AdaBoost**: using the ensemble of multiple weak learners to build a high accuracy classifier
  - Weak learners = small decision trees (1-split decision stumps)
  - Weights for each learner and instance
  - Instances are weighed on probability it's mistaken
  - Learners are weighed on its accuracy

Input: ℓ, α, $\{(\mathbf{x}_i, y_i)\}$, $\mathbb{A}$
$H_0 = 0$
$\forall i : w_i = \frac{1}{n}$
for *t=0:T-1* **do**
  $h = \mathbb{A}((w_1, \mathbf{x}_1, y_1), ..., (w_n, \mathbf{x}_n, y_n))$
  $\epsilon = \sum_{i:h(\mathbf{x}_i) \neq y_i} w_i$
  if $\epsilon < \frac{1}{2}$ **then**
    $\alpha = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$
    $H_{t+1} = H_t + \alpha h$
    $\forall i : \quad w_i \leftarrow \frac{w_i e^{-\alpha h(\mathbf{x}_i) y_i}}{2\sqrt{\epsilon(1-\epsilon)}}$
  else
    return (H_t)
  end
  return (H_T)
end

Gradient boosting regression tree

- Residuals

$$
\begin{aligned}
e &= y - \hat{y} \\
&= y - f(x;\theta) \\
&\dots \\
&= y - f(x;\theta) - f'(x;\theta)
\end{aligned}
$$



Gradient boosting regression tree

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:

   1. Compute so-called *pseudo-residuals*:

$$\boxed{r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

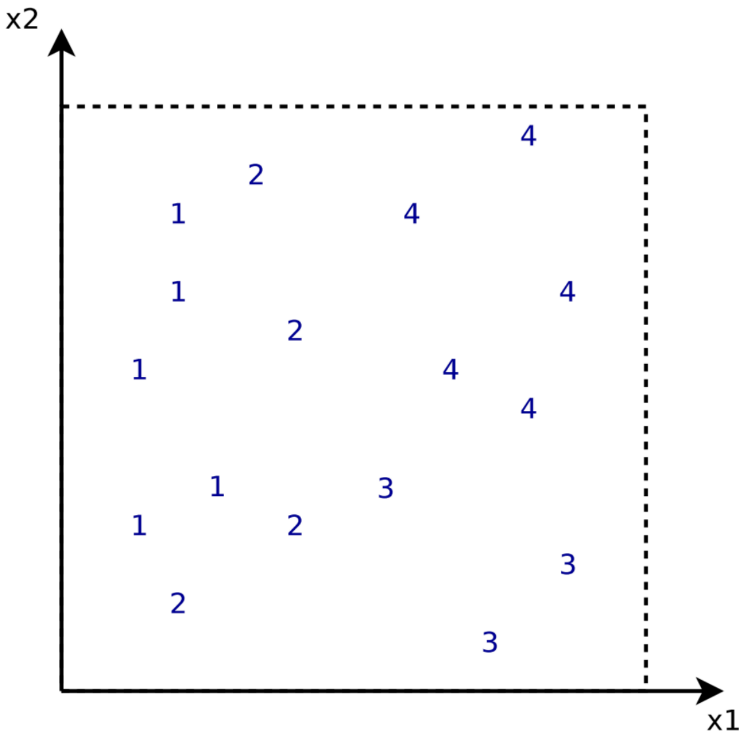
residuals of square loss are just pseudo gradients

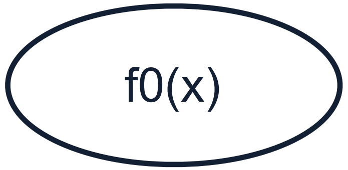
(which is why it's called gradient boosting)

$$\min_{w,b} \sum_{(q,d,r)} (r - score(q,d))^2$$

Gradient boosting regression tree example

- In the first iteration, f0(x) = mean value



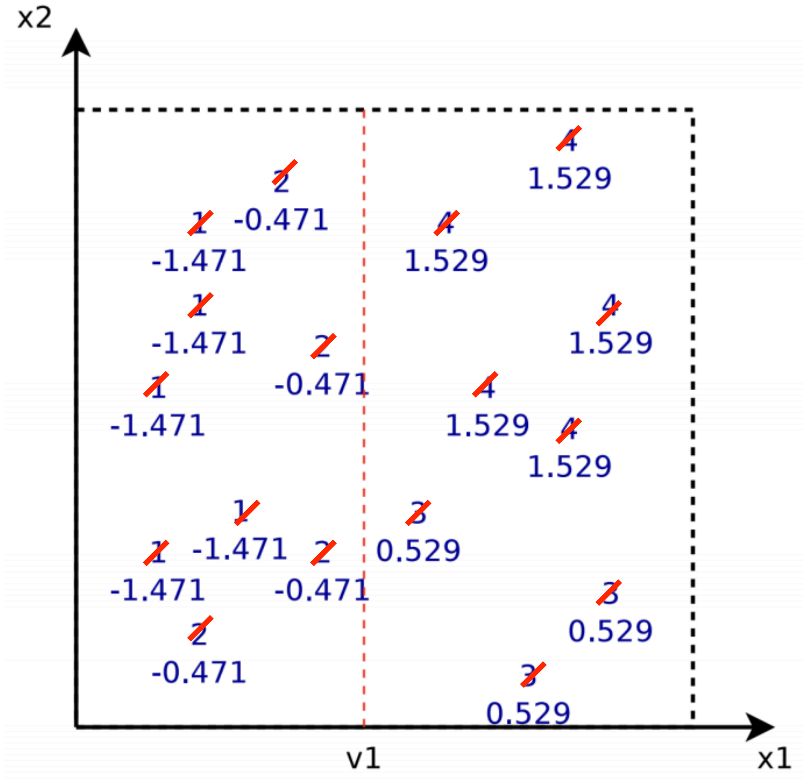


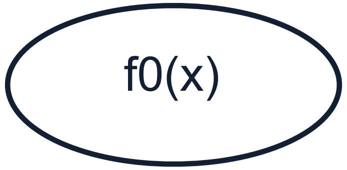
$$min_x 5(1-x)^2 + 4(2-x)^2$$
$$+ 3(3-x)^2 + 5(4-x)^2$$

$x =$ 2.471

Gradient boosting regression tree example

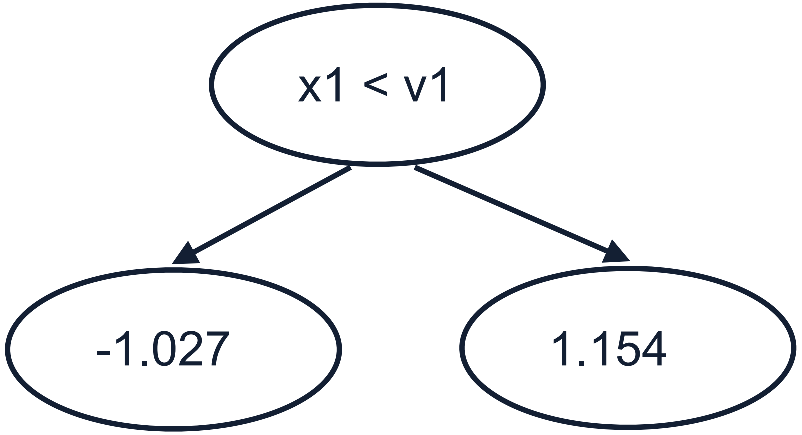
- After the second iteration, F(x) = f0(x) + f1(x)





2.471

f1(x)



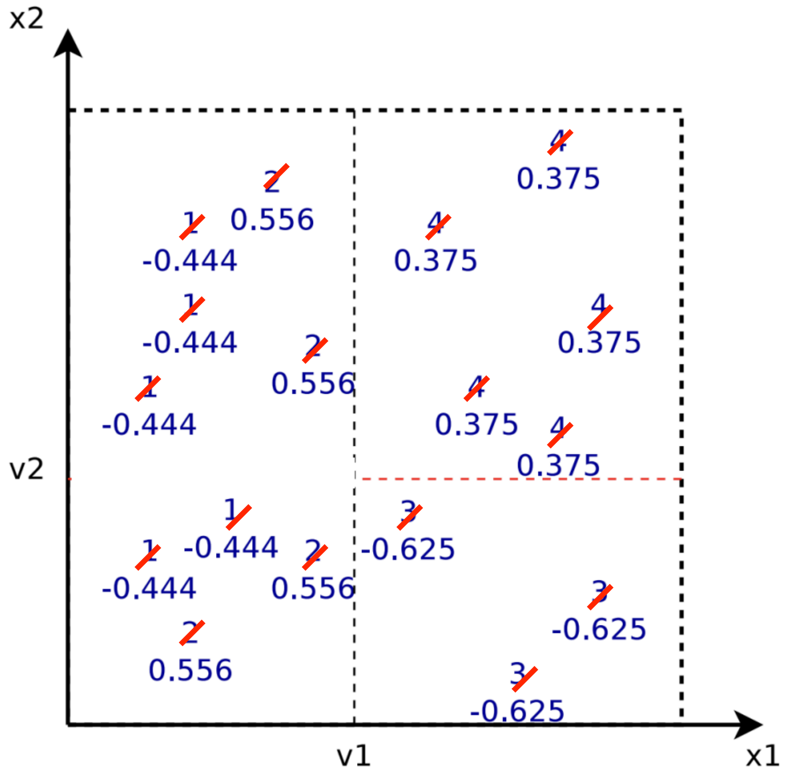
$$min_x 5(1-2.471-x)^2 + 4(2-2.471-x)^2$$

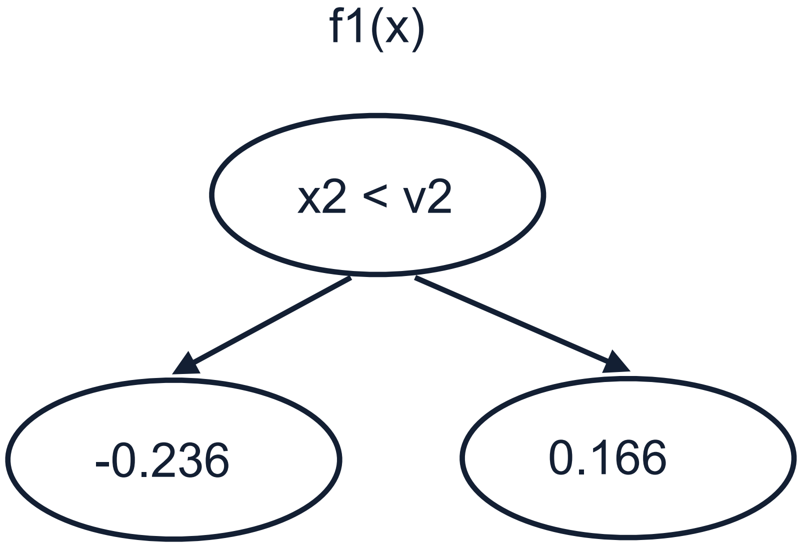
$$min_x 3(3-2.471-x)^2 + 5(4-2.471-x)^2$$

$$2.471 - 1.027 = 1.444$$
$$2.471 + 1.154 = 3.625$$

Gradient boosting regression tree example

- After the third iteration, F(x) = f0(x) + f1(x) + f2(x)





$$min_x 2(1-1.444-x)^2 + 2(2-1.444-x)^2 + 3(3-3.625-x)^2$$

$$min_x 3(1-1.444-x)^2 + 2(2-1.444-x)^2 + 5(4-3.625-x)^2$$

RankNet [Burges et al. 2010]

- Use si to denote the ranking function:

$$s_i = wx_i + b \qquad P(d_i \succ d_j) = \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

$$\min_{\Theta} \sum_{i,j} -\mathbb{1}\,[r_i > r_j] \log P(d_i \succ d_j) - (1 - \mathbb{1}\,[r_i \succ r_j]) \log (1 - P(d_i \succ d_j))$$

- Plugging in the probability gives rise to:

$$\min_{\Theta} \sum_{i,j} C_{i,j}$$

$$C_{i,j} = \frac{1}{2}(1 - S_{i,j})\sigma(s_i - s_j) + \log (1 + e^{-\sigma(s_i - s_j)}), S_{i,j} \in \{0, +1, -1\}$$

Sij = 1 if di is more relevant than dj; –1 if the reverse, and
0 if the they have the same label

RankNet [Burges et al. 2010]

$$C_{i,j} = \frac{1}{2}(1 - S_{i,j})\sigma(s_i - s_j) + \log\left(1 + e^{-\sigma(s_i - s_j)}\right), S_{i,j} \in \{0, +1, -1\}$$

$$\frac{\partial C_{i,j}}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C_{i,j}}{\partial s_j}$$

$$\frac{\partial C_{i,j}}{\partial w_k} = \frac{\partial C_{i,j}}{\partial s_i}\frac{\partial s_i}{\partial w_k} + \frac{\partial C_{i,j}}{\partial s_j}\frac{\partial s_j}{\partial w_k} = \sigma\left(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right)\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right) = \lambda_{i,j}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right)$$

$$\frac{\partial \Sigma}{\partial w_k} = \sum_{i,j}\frac{\partial C_{i,j}}{\partial w_k} = \sum_{i,j}\lambda_{i,j}\left(\frac{\partial s_i}{\partial w_k} - \frac{\partial s_j}{\partial w_k}\right) = \sum_i \underbrace{\left(\sum_{j:\{i,j\}\in I}\lambda_{i,j} - \sum_{l:\{l,i\}\in I}\lambda_{l,i}\right)}_{\lambda_i}\frac{\partial s_i}{\partial w_k}$$

{i, j} in I: for all pairs of i, j in the data (both positive and negative)

LambdaRank

- **RankNet:** minimize the pairwise ranking error

$$\lambda_{i,j} = \sigma(\frac{1}{2}(1 - S_{i,j}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}})$$

- **LambdaRank**: minimize the pairwise ranking error, scale by the change in NDCG

$$\lambda_{i,j} = -\frac{\sigma}{1 + e^{(\sigma(s_i - s_j)}}|\Delta NDCG|$$

LambdaMART [Burges et al. 2010]

- Lambdas are kind of “gradients” in RankNet
- In MART, with the specific lambda as gradients, we get:
  - LambdaMART = LambdaRank + MART (gradient boosting)

set number of trees N, number of training samples m, number of leaves per tree L
learning rate η
for $i = 0$ to m **do**
  $F_0(x_i) = \text{BaseModel}(x_i)$ //If BaseModel is empty, set $F_0(x_i) = 0$
end for
for $k = 1$ to N **do**
  for $i = 0$ to m **do**
    $\boxed{y_i = \lambda_i}$
    $w_i = \frac{\partial y_i}{\partial F_{k-1}(x_i)}$
  end for
  $\{R_{lk}\}_{l=1}^{L}$ // Create L leaf tree on $\{x_i, y_i\}_{i=1}^{m}$ R_{lk} is data items at leaf node l
  $\gamma_{lk} = \frac{\sum_{x_i \in R_{lk}} y_i}{\sum_{x_i \in R_{lk}} w_i}$ // Assign leaf values based on Newton step.
  $F_k(x_i) = F_{k-1}(x_i) + \eta \sum_l \gamma_{lk} I(x_i \in R_{lk})$ // Take step with learning rate η.
end for

XGBoost [Chen and Guestrin]

- State-of-the-art algorithm for gradient boosting

- Ingredients
  - Regularization
  - Gradient boosting
  - Approximate greedy algorithm
  - Weighted quantile sketch
  - Sparsity aware split finding
  - Parallel learning
  - …

LTR: real world use

- Systems that currently used LambdaMART:
  - Bing, search ads
- However:
  - Machine learning was not heavily used in Google! (**why?**)
  - Rule based systems are more interpretable and easy to debug:

From Google's dominance in web search, it's fairly clear that the decision to optimize for explainability and control over search result rankings has been successful at allowing the team to iterate and improve rapidly on search ranking quality

(answer from Quora, 2011)

https://www.quora.com/Why-is-machine-learning-used-heavily-for-Googles-ad-ranking-and-less-for-their-search-ranking-What-led-to-this-difference

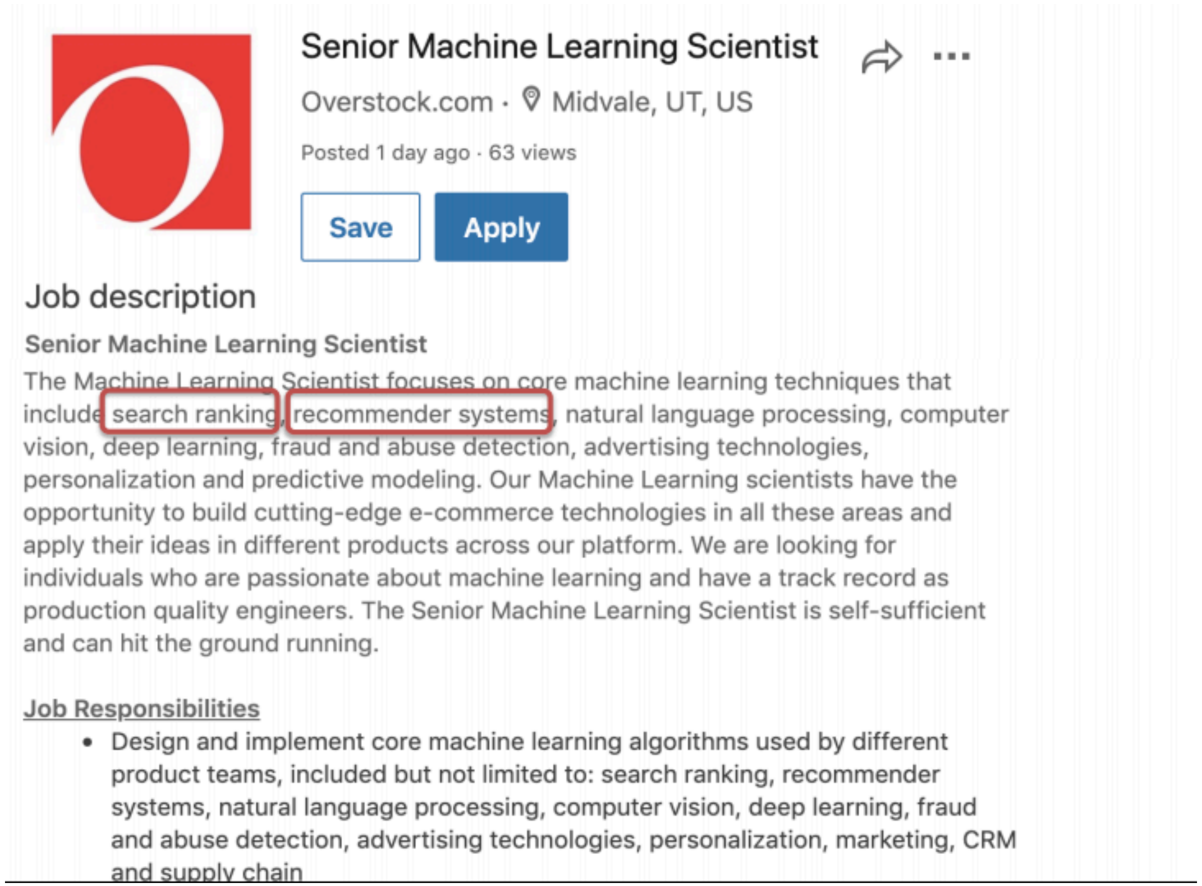
LTR: real world use

- In 2015, Google introduced RankBrain, a query interpretability approach

- The 3rd important feature of Google

- Guessing game of what ambiguous queries mean
  - Human: 70%
  - RankBrain: 80%

Optimizing CTR for industry search engine


Senior Machine Learning Scientist

Overstock.com · Midvale, UT, US

Posted 1 day ago · 63 views

Save Apply

Job description

Senior Machine Learning Scientist

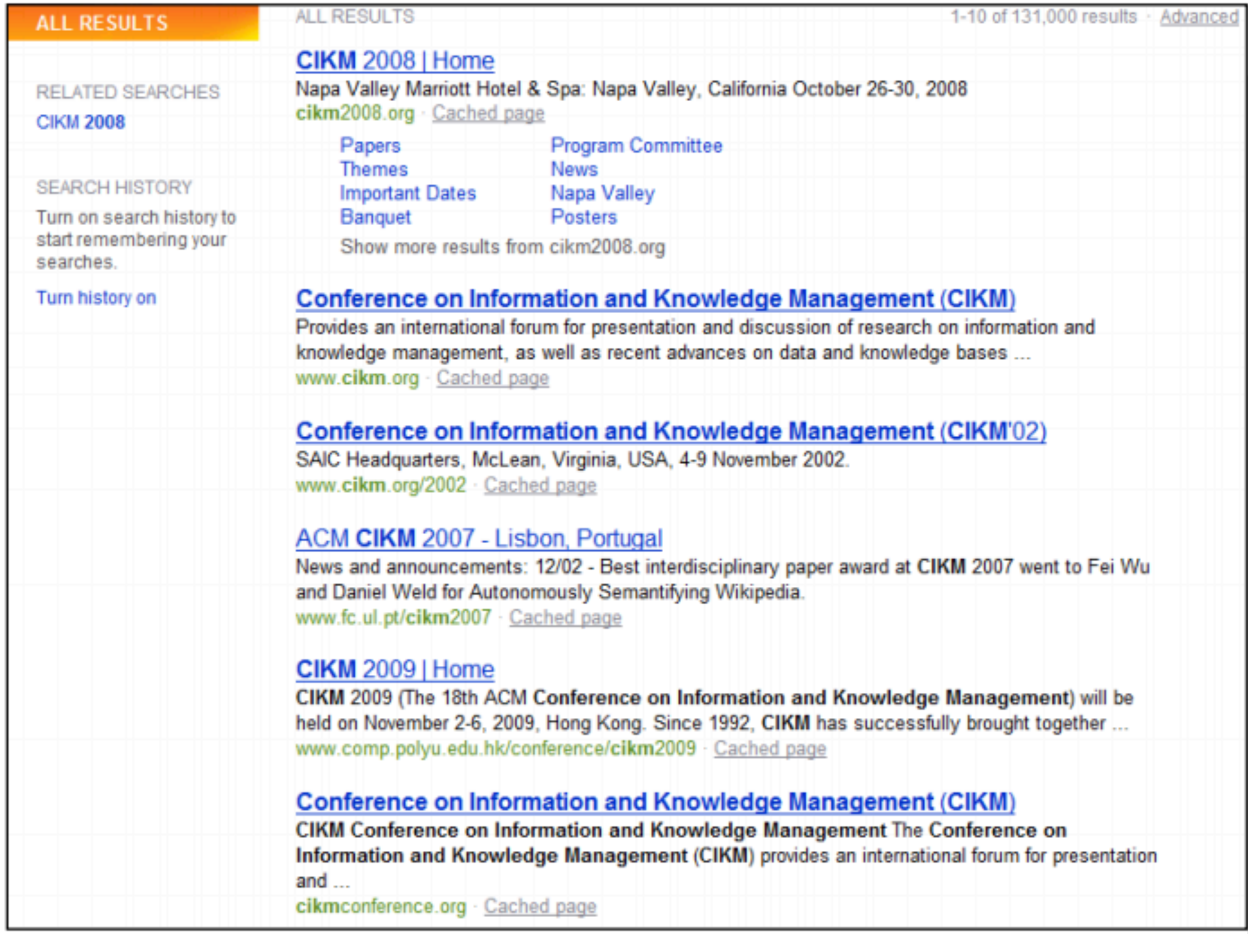
The Machine Learning Scientist focuses on core machine learning techniques that include search ranking, recommender systems, natural language processing, computer vision, deep learning, fraud and abuse detection, advertising technologies, personalization and predictive modeling. Our Machine Learning scientists have the opportunity to build cutting-edge e-commerce technologies in all these areas and apply their ideas in different products across our platform. We are looking for individuals who are passionate about machine learning and have a track record as production quality engineers. The Senior Machine Learning Scientist is self-sufficient and can hit the ground running.

Job Responsibilities

- Design and implement core machine learning algorithms used by different product teams, included but not limited to: search ranking, recommender systems, natural language processing, computer vision, deep learning, fraud and abuse detection, advertising technologies, personalization, marketing, CRM and supply chain

Boss: I have all the **user click logs** (3 million records) for the last year, implement an algorithm for improving the click through rate for the next quarter

Web search: how clicks happen



query = "CIKM" (year = 2009)

Which websites are most clicked?

- Relevance
- Context (location, time)
- Personalization
- Other bias

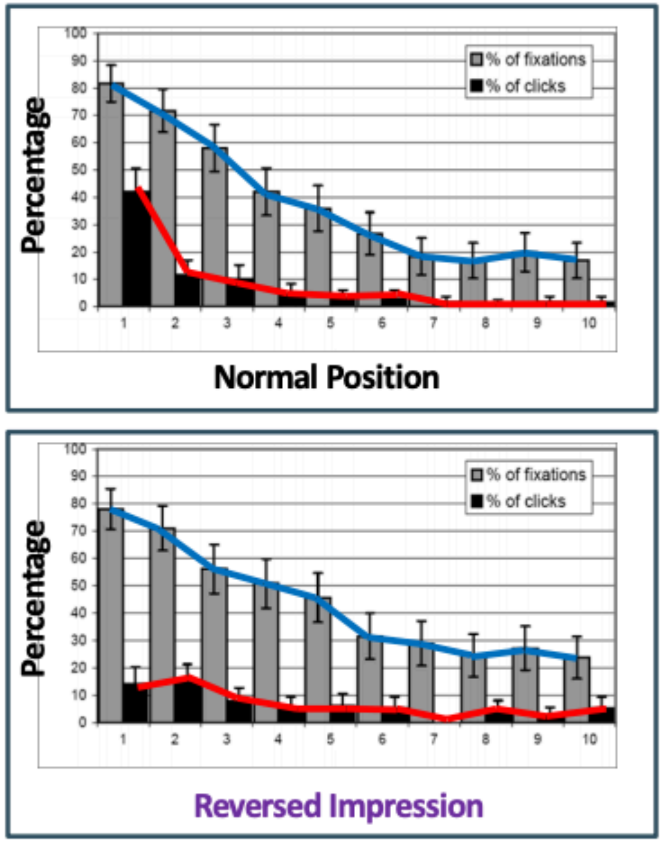
User clicks as implicit feedback

- User clicks != explicit relevance judgment
  - Position bias
  - Exploratory search: clicks on A, not click on B does not always mean A is more relevant than B
  - Clicks are inconsistent

- User clicks ~ noisy relevance feedback
  - Debias the feedback
  - Processing user clicks for better quality
  - Using comparative user feedback

Position bias

- Users always click higher ranked items, regardless of their (relative) relevance
  -



- Higher positions receive more user attention (eye fixation) and clicks than lower positions.
- This is true even in the extreme setting where the order of positions is reversed.
- "Clicks are informative but biased".

[Joachims+07]

Position bias modeling

- Hypothesis testing on user click models:

Hypothesis 1: Click probability is independent of position

$$c_{di} = r_d = c_{dj}$$

Hypothesis 2: Click probability is a mixture model

$$c_{di} = \lambda r_d + (1 - \lambda) b_i$$

Hypothesis 3: Click probability follows a cascade model

$$c_{di} = r_d \prod_{j=1}^{i-1} (1 - r_{docinrank:j})$$

Craswell et al. *An Experimental Comparison of Click Position-Bias Models.* WSDM 2008

Position bias modeling

- Testing hypothesis using a small portion of users in a search engine
  - query, A, B, m
  - query, B, A, m
- There are four types of events:
  - A clicked, B not clicked
  - B clicked, A not clicked
  - both A/B clicked
  - neither A/B clicked
- Based on query,A,B,m's result + hypothesis, estimate query, B,A,m's result

Position bias modeling [Craswell 2009]

- Using cross entropy to examine hypothesis

$$Cross\ Entropy = -\sum_{e} p(e) \log p'(e)$$

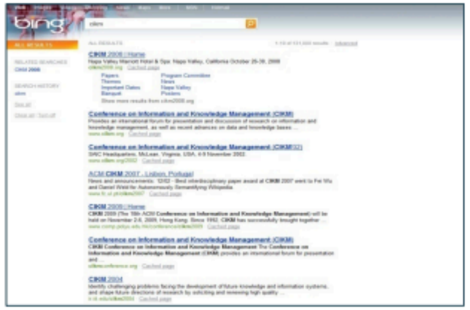
- Cascade model has the lowest CE

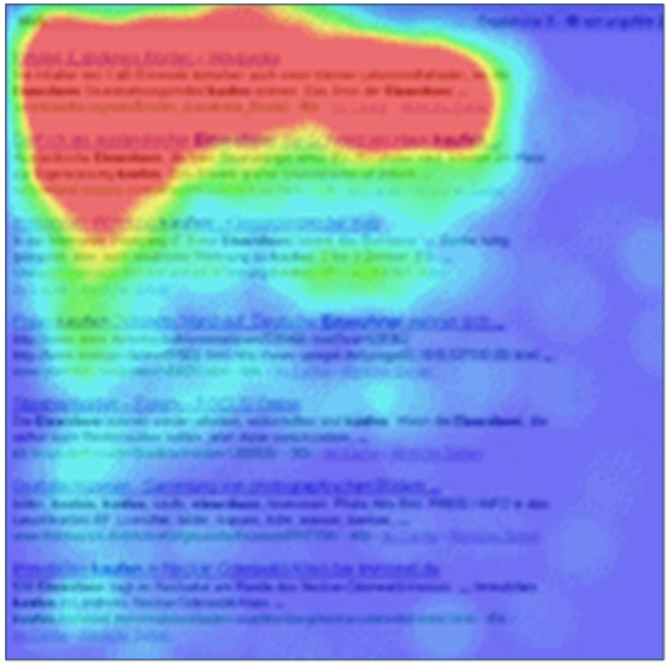
Model	Cross Entropy
Best Possible	0.141 ± 0.0055
Cascade	0.225 ± 0.0052
Logistic	0.236 ± 0.0063
Examination	0.247 ± 0.0072
Baseline	0.250 ± 0.0073

Leveraging other user signals

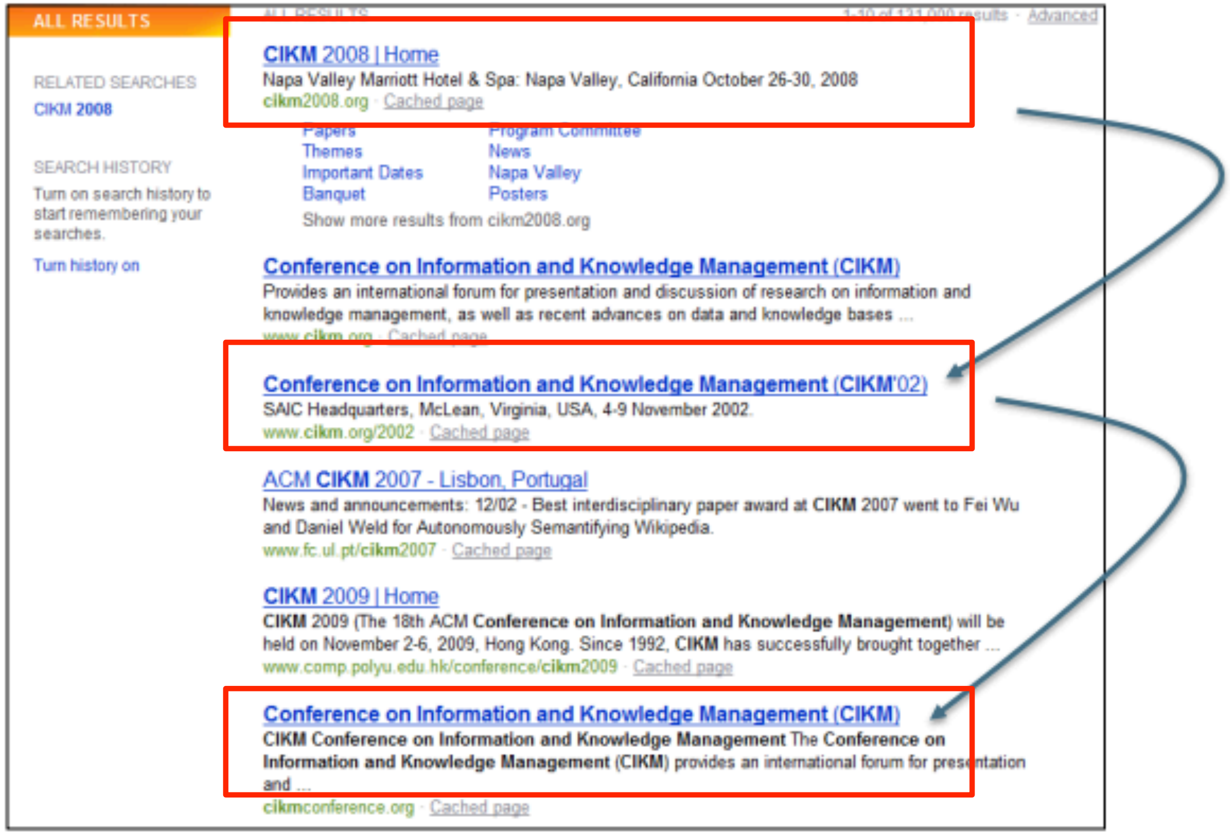
- SAT clicks
  - Clicks that are long enough (> 30 sec)

- Using eye tracking





Using comparative user feedback



User's click sequence

clicked documents are more relevant than unclicked documents (pairwise learning to rank)

Summary

- Web search with user feedback
  - User clicks as implicit feedback
  - Search engine position bias

- Learning to rank
  - Regression tree
  - Gradient boosting
  - RankNet, LambdaRank, LambdaMART

- Real-world use of LambdaMART