

Learning to Rank

Nirajan Bekoju

Problem Statement

Say that each training example in our dataset belongs to a customer. Say that we have some feature vector for this customer which serves as an input to ur model. For our labels we have an ordered list of products which are ordered by relevance to that customer. How can we go about training a model that learns to rank this list of products in the order described by our labels?

-0.21 1.5 -1.52 ... 0.87



Model



1.



2.



3.



Learning to Rank

- Learning to rank refers to machine learning techniques for training the model in a ranking task.

Scenario

- Let's use a scenario most of us are familiar with to understand what this is: searching for an article on wikipedia
 - We have a website, Wikipedia with a search function
 - User submit search requests('queries') to the search function
 - Users are then presented with ranked lists of articles('documents').
- In learning to rank, the list ranking is performed by a ranking model $f(q, d)$, where:
 - f is some ranking function that is learnt through supervised learning
 - q is our query, and
 - d is our document

What does our training data look like?

- The curator asks users to assign each article one of the relevance grades number(say 2 for relevant, 1 for somewhat relevant, and 0 for irrelevant)



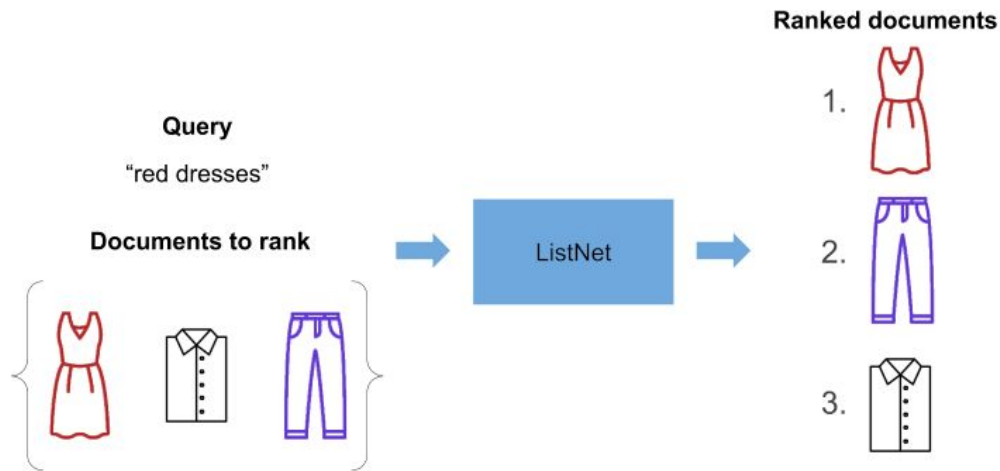
What features will we be using?

- We'll be using neural nets so we could be using any arbitrary feature that we think might help in our ranking task.
- In the example, we'll be focussing on using the words in our **queries and documents**.

Part 2: let's implement ListNet

Problem Scenario

We are going to give ListNet a query, and a bunch of documents to rank. Then, as if through some sorcery, we get a ranked list of documents



Output of Listnet

ListNet outputs a bunch of real numbers. Each real number is a score assigned to the document we want to rank and then simply sort them to obtain ranked documents.



How does the paper describe ListNet?

We employ a new learning method for optimizing the listwise loss function based on the top one probability, with Neural network as model and Gradient Descent as optimization algorithm. We refer to the method as ListNet.

- What do they mean by **Listwise**?
- What is the **top one probability** they speak of?
- What is the **listwise loss function**?
- What is the **neural network architecture**?

Approach to “Learning to Rank”

- Pointwise
- Pairwise
- Listwise

What is a “listwise” approach to learning to rank?

- The listwise approach addresses the ranking problem in a more straightforward way. Specifically, it takes ranking lists as instances in both learning and prediction. The group structure of ranking is maintained and ranking evaluation measures can be more directly incorporated into the loss function in learning.
- Pointwise and pairwise approaches ignore the group structure of rankings.
- Learning to rank often involves optimizing a surrogate loss function.
 - The loss function may be difficult to minimize because it isn't continuous and uses sorting.
 - Listnet allows us to construct our ranking task in such a way that decreasing its loss value more directly impacts our true objective i.e increasing NDCG or MAP

Where do probabilities fit into ListNet?

- The author use a probability-based approach to map their lists of scores to probability distributions.
- Once this is done, they calculate their loss between the predicted probability distribution and a target probability distribution.
- *We assume that there is uncertainty in the prediction of ranking lists(permutation) using the ranking function. In other words, any permutation is assumed to be possible, but different permutations may have different likelihood calculated based on the ranking function. We define the permutation probability, so that it has desirable properties for representing the likelihood of a permutation (ranking list), given the ranking function.*

Probability Models

- Permutation Probability
- Top One Probability

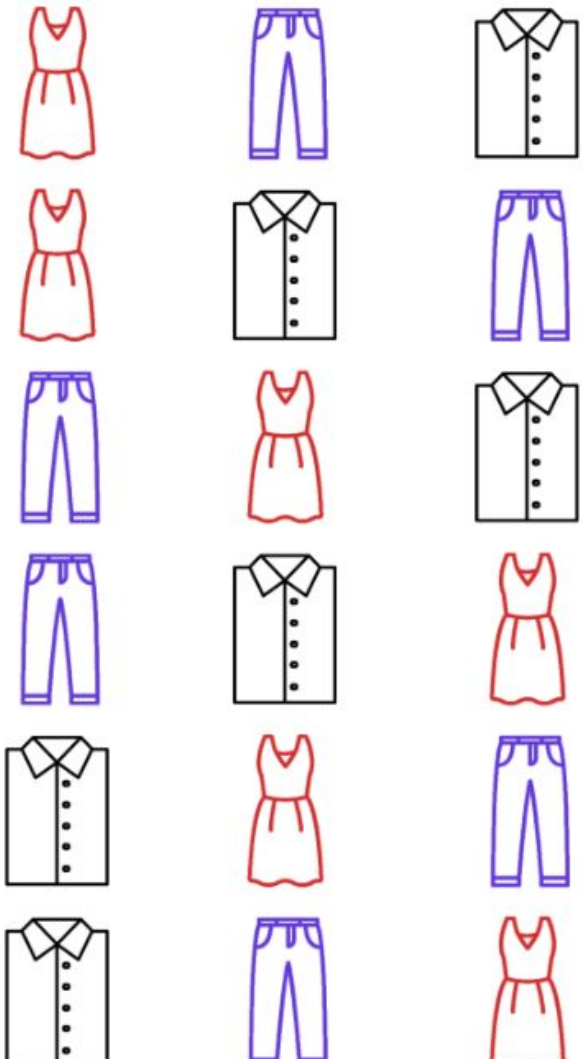
Permutation Probability

Say, we have three objects to rank:



Continued: Permutation Probability

All the possible permutations for the given objects are shown.



Continued: Permutation Probability

Let the following be the score given by our model to each object.



1.62



-0.61



-0.53

Continued: Permutation Probability

$$P_s(\pi) = \prod_{j=1}^n \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^n \phi(s_{\pi(k)})}$$

- $\phi(x) = \exp(x)$
- The probability of one of the permutation is given as below
- Firstly, We are calculating the probability of some permutation π given some list of scores s . This is depicted by the LHS of the above by $P_s(\pi)$
- Next, we notice the big \prod . This symbol says that we will be calculating the product of n terms.
- Next, we have some ϕ 's. Here, it's simply some transformation applied to our scores. The only requirement is that it is “an increasing and strictly positive function”
- The denominator contains a big \sum . It tells us that we will be summing $n-k+1$ terms. Each one of these terms is a score transformed by the same function ϕ .

Continued: Permutation Probability Formula Explored

- For the permutation (dress, pants, shirt)



















$$\text{first term} = \frac{e^{s_{\text{dress}}}}{e^{s_{\text{dress}}} + e^{s_{\text{pants}}} + e^{s_{\text{shirt}}}}$$

$$\text{second term} = \frac{e^{s_{\text{pants}}}}{e^{s_{\text{pants}}} + e^{s_{\text{shirt}}}}$$

$$\text{third term} = \frac{e^{s_{\text{shirt}}}}{e^{s_{\text{shirt}}}} = 1$$

Continued: For all permutations

- The scores sorted in descending order have the highest permutation probability.
- The scores sorted in ascending order have the lowest permutation probability.

| Rank 1 | Rank 2 | Rank 3 | |
|---|---|---|--------|
|  |  |  | 42.59% |
|  |  |  | 39.17% |
|  |  |  | 8.58% |
|  |  |  | 0.92% |
|  |  |  | 7.83% |
|  |  |  | 0.01% |

Issue: Permutation Probability

- To calculate the difference between our distributions using a listwise loss function, we could first calculate the permutation probability distributions for each training example.
- But the issue is that there are $n!$ permutations and number of calculations quickly get out of hand.
- Hence, Top one probability is proposed.

Top one probability

Give some object, j that we want to rank, the top one probability for that object is the sum of the permutation probabilities of the permutations where j is ranked first.

$$P_s(j) = \sum_{\pi(1)=j, \pi \in \Omega_n} P_s(\pi)$$

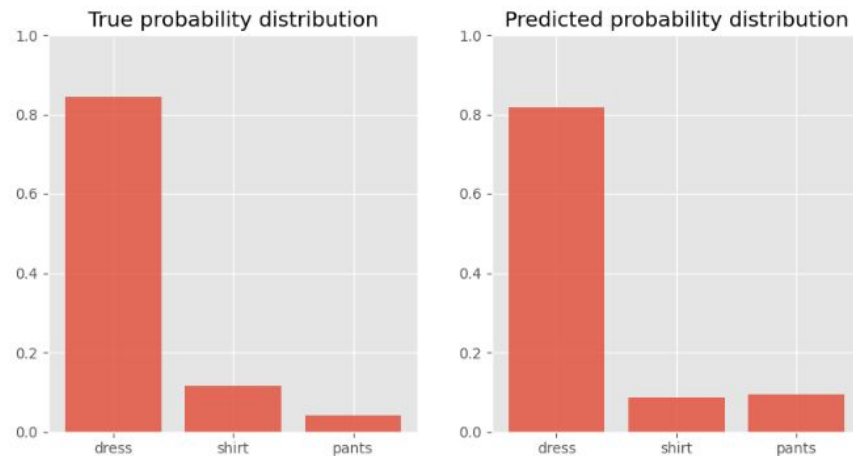
Continued: Top one probability

- The authors then observe that to calculate the top one probability of a given object, one doesn't need to calculate all permutation probabilities of n objects to rank! The top one probability of our object is equivalent to this:
- Here, S_j is the score of the j -th object.

$$P_s(j) = \frac{\exp(s_j)}{\sum_{k=1}^n \exp(s_k)}$$

Converting scores and relevance labels into probability distributions

- Calculate the softmax of the true relevant score and the predicted score.
- Note: The top one probability formula is similar to softmax calculation.
- We can observe, that the probabilities for shirt and pants are ranked in incorrect order.

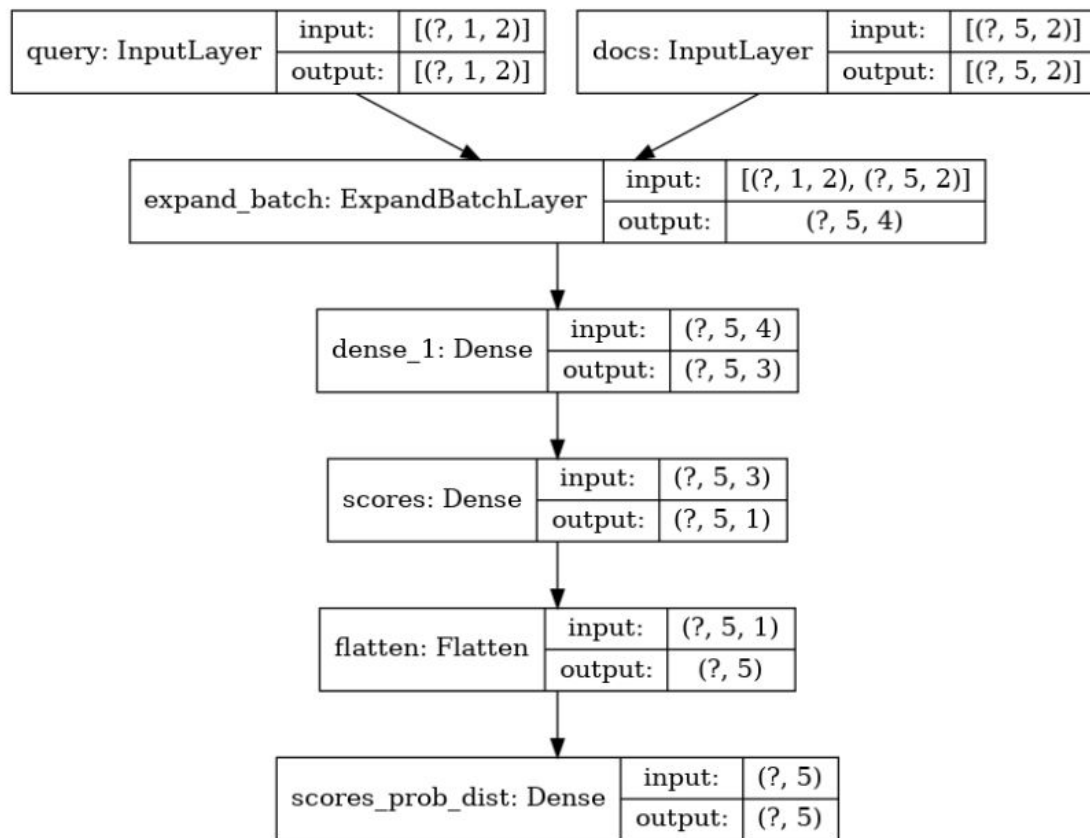


Loss Function - KL Divergence

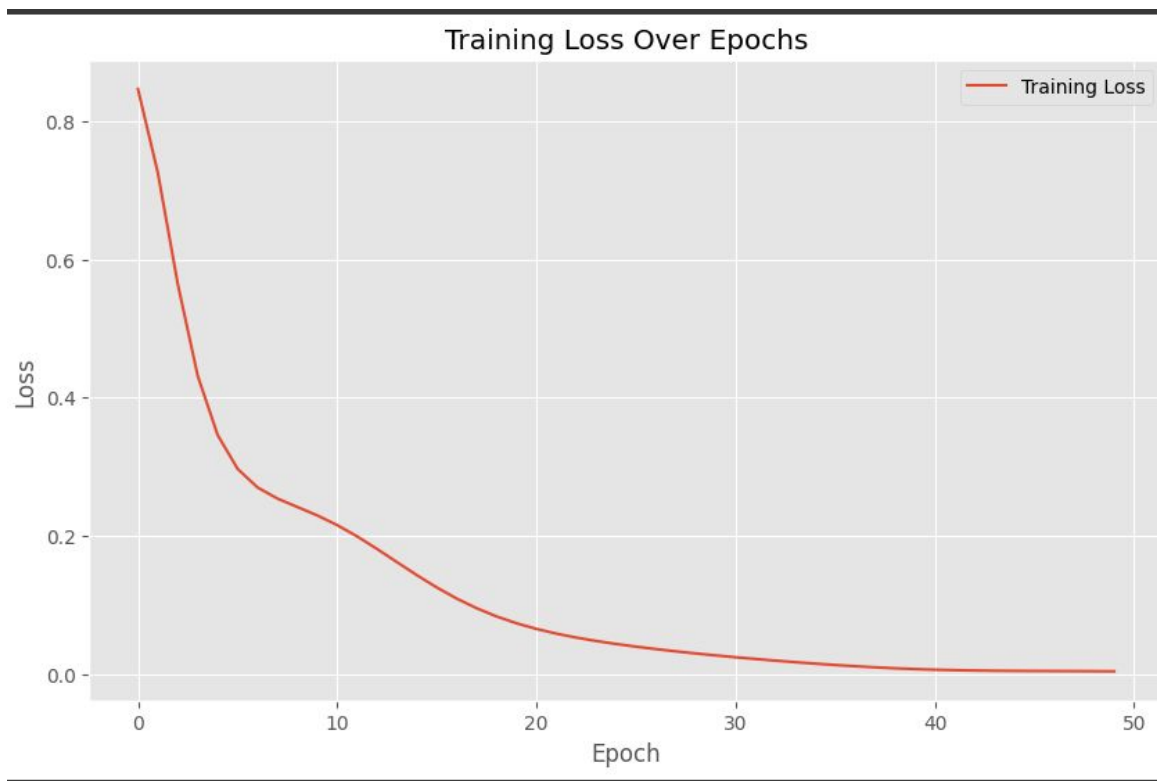
- If we have two separate probability distribution $P(X)$ and $Q(X)$ over the same random variable X , we can measure how different these two distributions are using the KL divergence.

$$D_{KL} = \text{true distribution} \cdot \log\left(\frac{\text{true distribution}}{\text{predicted distribution}}\right)$$

Neural Network



Training Curve



Result

