There are only $m!/(m-k)!$ such permutations. If we set $k=1$, we can evaluate each cross entropy term (and its derivative) in $O(m)$ time.

In the special case where only one document from the presented list is deemed relevant, say $y_i = c$, we can instead use multinomial logistic regression:

$$p(y_i = c|\mathbf{x}) = \frac{\exp(s_c)}{\sum_{c'=1}^{m} \exp(s_{c'})} \tag{9.125}$$

This often performs at least as well as ranking methods, at least in the context of collaborative filtering (Yang et al. 2011).

9.7.4 Loss functions for ranking

There are a variety of ways to measure the performance of a ranking system, which we summarize below.

- **Mean reciprocal rank (MRR)**. For a query q, let the rank position of its first relevant document be denoted by $r(q)$. Then we define the **mean reciprocal rank** to be $1/r(q)$. This is a very simple performance measure.
- **Mean average precision (MAP)**. In the case of binary relevance labels, we can define the **precision at k** of some ordering as follows:

$$\text{P@k}(\boldsymbol{\pi}) \triangleq \frac{\text{num. relevant documents in the top } k \text{ positions of } \boldsymbol{\pi}}{k} \tag{9.126}$$

  We then define the average precision as follows:

$$\text{AP}(\boldsymbol{\pi}) \triangleq \frac{\sum_k \text{P@k}(\boldsymbol{\pi}) \cdot I_k}{\text{num. relevant documents}} \tag{9.127}$$

  where I_k is 1 iff document k is relevant. For example, if we have the relevancy labels $\mathbf{y} = (1,0,1,0,1)$, then the AP is $\frac{1}{3}(\frac{1}{1} + \frac{2}{3} + \frac{3}{5}) \approx 0.76$. Finally, we define the **mean average precision** as the AP averaged over all queries.
- **Normalized discounted cumulative gain (NDCG)**. Suppose the relevance labels have multiple levels. We can define the **discounted cumulative gain** of the first k items in an ordering as follows:

$$\text{DCG@k}(\mathbf{r}) = r_1 + \sum_{i=2}^{k} \frac{r_i}{\log_2 i} \tag{9.128}$$

  where r_i is the relevance of item i and the $\log_2$ term is used to discount items later in the list. Table 9.3 gives a simple numerical example. An alternative definition, that places stronger emphasis on retrieving relevant documents, uses

$$\text{DCG@k}(\mathbf{r}) = \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(1+i)} \tag{9.129}$$

  The trouble with DCG is that it varies in magnitude just because the length of a returned list may vary. It is therefore common to normalize this measure by the ideal DCG, which is

i	1	2	3	4	5	6
r_i	3	2	3	0	1	2
$\log_2 i$	0	1	1.59	2.0	2.32	2.59
$\frac{r_i}{\log_2 i}$	N/A	2	1.887	0	0.431	0.772

Table 9.3 Illustration of how to compute NDCG, from `http://en.wikipedia.org/wiki/Discounted_cumulative_gain`. The value r_i is the relevance score of the item in position i. From this, we see that DCG@6 $= 3 + (2 + 1.887 + 0 + 0.431 + 0.772) = 8.09$. The maximum DCG is obtained using the ordering with scores 3, 3, 2, 2, 1, 0. Hence the ideal DCG is 8.693, and so the normalized DCG is 8.09 / 8.693 = 0.9306.

the DCG obtained by using the optimal ordering: IDCG@k($\mathbf{r}$) $= \text{argmax}_{\boldsymbol{\pi}}$ DCG@k($\mathbf{r}$). This can be easily computed by sorting $r_{1:m}$ and then computing DCG@k. Finally, we define the **normalized discounted cumulative gain** or **NDCG** as DCG/IDCG. Table 9.3 gives a simple numerical example. The NDCG can be averaged over queries to give a measure of performance.

- **Rank correlation**. We can measure the correlation between the ranked list, $\boldsymbol{\pi}$, and the relevance judgement, $\boldsymbol{\pi}^*$, using a variety of methods. One approach, known as the (weighted) **Kendall's** τ statistics, is defined in terms of the weighted pairwise inconsistency between the two lists:

$$\tau(\boldsymbol{\pi}, \boldsymbol{\pi}^*) = \frac{\sum_{u<v} w_{uv} \left[1 + \text{sgn}(\pi_u - \pi_v)\text{sgn}(\pi_u^* - \pi_v^*)\right]}{2\sum_{u<v} w_{uv}} \tag{9.130}$$

  A variety of other measures are commonly used.

These loss functions can be used in different ways. In the Bayesian approach, we first fit the model using posterior inference; this depends on the likelihood and prior, but not the loss. We then choose our actions at test time to minimize the expected future loss. One way to do this is to sample parameters from the posterior, $\boldsymbol{\theta}^s \sim p(\boldsymbol{\theta}|\mathcal{D})$, and then evaluate, say, the precision@k for different thresholds, averaging over $\boldsymbol{\theta}^s$. See (Zhang et al. 2010) for an example of such an approach.

In the frequentist approach, we try to minimize the empirical loss on the training set. The problem is that these loss functions are not differentiable functions of the model parameters. We can either use gradient-free optimization methods, or we can minimize a surrogate loss function instead. Cross entropy loss (i.e., negative log likelihood) is an example of a widely used surrogate loss function.

Another loss, known as **weighted approximate-rank pairwise** or **WARP** loss, proposed in (Usunier et al. 2009) and extended in (Weston et al. 2010), provides a better approximation to the precision@k loss. WARP is defined as follows:

$$\text{WARP}(\mathbf{f}(\mathbf{x}, :), y) \triangleq L(\text{rank}(\mathbf{f}(\mathbf{x}, :), y)) \tag{9.131}$$

$$\text{rank}(\mathbf{f}(\mathbf{x}, :), y) = \sum_{y' \neq y} \mathbb{I}(f(\mathbf{x}, y') \geq f(\mathbf{x}, y)) \tag{9.132}$$

$$L(k) \triangleq \sum_{j=1}^{k} \alpha_j, \quad \text{with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0 \tag{9.133}$$

Here $\mathbf{f}(x,:) = [f(\mathbf{x},1),\ldots,f(\mathbf{x},|y|)]$ is the vector of scores for each possible output label, or, in IR terms, for each possible document corresponding to input query $\mathbf{x}$. The expression $\text{rank}(\mathbf{f}(\mathbf{x},:),y)$ measures the rank of the true label y assigned by this scoring function. Finally, L transforms the integer rank into a real-valued penalty. Using $\alpha_1 = 1$ and $\alpha_{j>1} = 0$ would optimize the proportion of top-ranked correct labels. Setting $\alpha_{1:k}$ to be non-zero values would optimize the top k in the ranked list, which will induce good performance as measured by MAP or precision@k. As it stands, WARP loss is still hard to optimize, but it can be further approximated by Monte Carlo sampling, and then optimized by gradient descent, as described in (Weston et al. 2010).

Exercises

Exercise 9.1 Conjugate prior for univariate Gaussian in exponential family form

Derive the conjugate prior for μ and $\lambda = 1/\sigma^2$ for a univariate Gaussian using the exponential family, by analogy to Section 9.2.5.5. By suitable reparameterization, show that the prior has the form $p(\mu,\lambda) = \mathcal{N}(\mu|\gamma,\lambda(2\alpha-1))\text{Ga}(\lambda|\alpha,\beta)$, and thus only has 3 free parameters.

Exercise 9.2 The MVN is in the exponential family

Show that we can write the MVN in exponential family form. Hint: use the information form defined in Section 4.3.3.