McFadden’s Discrete Choice and Softmax under Interval (and Other) Uncertainty: Revisited

Bartłomiej Jacek Kubica, Olga Kosheleva, and Vladik Kreinovich

Abstract Studies of how people actually make decisions have led to an empirical formula that, based on the utilities of possible outcomes, predicts the probability of different decisions. This formula is known as McFadden’s formula, after a Nobel prize winning economist who discovered it. A similar formula – known as softmax – describes the probability that the classification predicted by a deep neural network is correct, based on the neural network’s degrees of confidence in the object belonging to each class. In practice, we usually do not know the exact values of the utilities – or of the degrees of confidence. At best, we know the intervals of possible values of these quantities. For different values from these intervals, we get, in general, different probabilities. It is desirable to find the range of all possible values of these probabilities. In this paper, we provide a feasible algorithm for computing these ranges.

1 Formulation of the Problem

What is McFadden’s discrete choice: a brief reminder. According to the decision theory (see, e.g., [2, 3, 7, 11, 17, 18, 23]) preferences of a rational decision maker are described by a special function – called utility $u$ – so that the decision makes always selects the alternative $i$ with the largest possible value of utility $u_i$. In particular, this
means that decisions of a rational decision maker should be deterministic – in the sense that if we offer, to the decision maker, the same choice some time in the future, he/she will make the exact same decision.

In practice, however, people’s decisions are not deterministic: in many cases, we select the alternative with the largest utility, but sometimes, we select an alternative with somewhat smaller utility. In general, alternatives with higher utility are selected more frequently, while alternatives with lower utility are selected less frequently. A study of this phenomena led Daniel L. McFadden to the following empirical formula that uses the utilities $u_1, \ldots, u_n$ of different alternatives to predict the probability $p_i$ that this alternative will be selected:

$$p_i = \frac{\exp(k \cdot u_i)}{\sum_{j=1}^{n} \exp(k \cdot u_j)},$$

for some constant $k > 0$; see, e.g., [13, 14, 24]. For this discovery, Professor McFadden was awarded the Nobel Prize in Economics.

**What is softwax: a brief reminder.** One of the main applications of deep learning (see, e.g., [4]) is the classification problem, when:

- we are given several classes of objects, and
- we need to decide to which of the classes the given object belongs.

For example, in autonomous driving systems, we need to be able to tell whether an object in front of our car is a person, a bicycle, or another car. In the vast majority of cases, a trained neural network provides the correct answer, but sometimes the neural network errs. It is therefore desirable to make sure that the system not only provide an answer, but that it should also provide us with the probability that this answer is correct. This way, if this probability is low, we can perform additional measurements and observations.

In a nutshell, the neural networks classify the objects as follows: for each class $i$ of objects, a sub-network is trained to recognize objects of this class. For each object, each of these sub-networks produces a degree $u_i$ to which, according to this sub-network, the given object belongs to the $i$-th class. If we want a single answer, then, of course, we select the class $i$ for which the corresponding degree is the largest.

In addition to this selection, we also want to estimate the probabilities. In other words, based on the $n$ degrees $u_1, \ldots, u_n$, we need to estimates the probabilities $p_1, \ldots, p_n$ that the given object belongs to the corresponding class. Interestingly, an empirically reasonable way to estimate these probabilities is to use the following formula – which is very similar to the formula (1):

$$p_i = \frac{\exp(k \cdot u_i)}{\sum_{j=1}^{n} \exp(k \cdot u_j)}.$$
This formula is known as softmax – because, instead of the “hard” maximum, when we simply select the class $i$ with the largest degree $u_i$, we have “soft” maximum, when we select the most probable class with higher probability, but we also, with some non-zero probability, select other classes.

**Need to consider interval uncertainty.** The formula (1) assumes that we know the exact utility values $u_i$. In practice, we only get these values with some uncertainty. For example, we may only know the range $[u_i, \overline{u}_i]$ of possible values of $u_i$. For different values $u_i$ from the corresponding intervals, we get, in general, different values of the probabilities $p_i$. A natural question is: what is the resulting range $[p_i, \overline{p}_i]$ of possible values of each probability $p_i$?

A similar problem emerges in the softmax situation. The values $u_i$ are computed based on the results of measuring the corresponding object. Measurement are never absolutely accurate: the result $\bar{x}$ of measuring a quantity $x$ is, in general, somewhat different from the actual (unknown) value of the corresponding quantity. In many practical situations, the only information that we have about the measurement error $\Delta x \equiv \bar{x} - x$ is the upper bound $\Delta$ on the error’s absolute value: $|\Delta x| \leq \Delta$; see, e.g., [22]. In this case, after the measurement, the only information that we get about the actual value $x$ is that this value belongs to the interval $[\bar{x} - \Delta, \bar{x} + \Delta]$. For different values $x$ from the corresponding intervals, we get, in general, different values $u_i$. As a result, for each $i$, we get the interval $[u_i, \overline{u}_i]$ of possible values $u_i$ corresponding to different values of $x$; computing this interval is an important particular case of interval computations; see, e.g., [5, 9, 12, 16].

For different values $u_i$ from the corresponding intervals, we get, in general, different values of the probability $p_i$. It is therefore desirable to find the range $[p_i, \overline{p}_i]$ of possible values of each probability $p_i$.

This is useful in practice: for example:

- it is one thing to say that an estimate of the probability that the classification is correct is 80%, and
- it is a different thing to say that this probability is somewhere 70% and 90%.

**Resulting problem.** From the computational viewpoint, in both cases, we face the same problem:

- we know the value $k$, and we know intervals $[u_i, \overline{u}_i]$ of possible values of $u_i$;
- we want to find the range $[p_i, \overline{p}_i]$ of possible values of the expression (1).

At first glance, this problem sounds very complicated but, as we show, it is not. In general, problems of interval computations are NP-hard; see, e.g., [8, 21]. This means, crudely speaking, that unless $P = NP$ (which most scientists believe not to be the case), no feasible algorithm can solve all particular cases of this problem.

Interval computation problems are even NP-hard if we want to compute the range of a quadratic function. The expression (1) has exponential functions and division – operations much more complex than addition and multiplication needed to compute a quadratic expression. So, it seems reasonable to expect that computing (1) is also computationally complicated.
In this paper, we show, however, that the above problem is quite feasible; moreover, it can be solved in linear time. We also show that this feasibility holds for reasonable generalizations of the formula (1) and of interval uncertainty.

Why we say “Revisited”. In our title, we use the word Revisited, since we dealt with softmax and McFadden’s discrete choice under interval uncertainty in our previous paper [10]. The difference is that:

• in that paper, we dealt with a different problem: how to select the most reasonable single value of each probability \( p_i \) under interval uncertainty, while
• in this paper, we are interested in finding the whole range of possible probability values.

2 Our Algorithm

Preliminary analysis of the problem. To simplify our analysis, let us divide both the numerator and the denominator of the formula (1) by its numerator. As a result, we get the following expression:

\[
p_i = \frac{1}{1 + \sum_{j \neq i} \frac{\exp(k \cdot u_j)}{\exp(k \cdot u_i)}}.
\] (3)

Here:

• Each fraction \( \frac{\exp(k \cdot u_j)}{\exp(k \cdot u_i)} \) increases with \( u_j (j \neq i) \) and decreases with \( u_i \).
• Thus, the denominator – which is the sum of these terms – also increases with each \( u_j (j \neq i) \) and decreases with \( u_i \).
• Since the function \( 1/x \) is decreasing, the probability \( p_i \) – which is equal to 1 over denominator – decreases with \( u_j (j \neq i) \) and increases with \( u_i \).

So:

• the probability \( p_i \) is the largest when \( u_i \) is the largest possible and other values \( u_j \) are the smallest possible, i.e., \( u_i = \overline{u}_i \) and \( u_j = \underline{u}_j \) for all \( j \neq i \); and
• the probability \( p_i \) is the smallest when \( u_i \) is the smallest possible and other values \( u_j \) are the largest possible, i.e., \( u_i = \underline{u}_i \) and \( u_j = \overline{u}_j \) for all \( j \neq i \).

Thus, we arrive at the following formulas:

\[
p_i = \frac{\exp(k \cdot u_i)}{\exp(k \cdot \underline{u}_i) + \sum_{j \neq i} \exp(k \cdot \overline{u}_j)};
\] (4)
\[ p_i = \frac{\exp(k \cdot \pi_i)}{\exp(k \cdot \pi_i) + \sum_{j \neq i} \exp(k \cdot u_j)}. \] (5)

**What if we follow these formulas directly?** According to the formulas (4) and (5), to compute each of the \(2n\) bounds \(p_i\) and \(\overline{p}_i\), we need a linear number of steps \(C \cdot n\) for some constant \(C\). Thus, overall, we need \(2n \cdot C \cdot n = O(n^2)\), i.e., quadratic time.

Can we compute all the probabilities faster? Yes, it we take into account that, e.g., for the formula (4), if we add and subtract the term \(\exp(k \cdot \pi_i)\) to its denominator – thus not changing the value of the denominator – we get the form

\[ \exp(k \cdot u_i) - \exp(k \cdot \pi_i) + \sum_{j=1}^{n} \exp(k \cdot u_j). \] (6)

Similarly, if we add and subtract the term \(\exp(k \cdot u_i)\) to the denominator of the formula (5), we get the following expression:

\[ \exp(k \cdot \pi_i) - \exp(k \cdot u_i) + \sum_{j=1}^{n} \exp(k \cdot u_j). \] (7)

The \(n\)-term sums in the expressions (6) and (7) are the same for all \(i\), so they can be computed only once. Thus, we arrive at the following linear-time algorithm.

**Linear-time algorithm for computing the ranges \([p_i, \overline{p}_i]\).** We are given the values \(u_i\) and \(\pi_i\). Based on these values:

- first, we compute the values \(\exp(k \cdot u_i)\), \(\exp(k \cdot \pi_i)\), and the differences
  \[ m_i \overset{\text{def}}{=} \exp(k \cdot \pi_i) - \exp(k \cdot u_i); \]
- then, we compute the sums \(s = \sum_{i=1}^{n} \exp(k \cdot u_i)\) and \(\overline{s} = \sum_{i=1}^{n} \exp(k \cdot \pi_i)\);
- after that, we compute the desired values
  \[ p_i = \frac{\exp(k \cdot u_i)}{s - m_i}; \quad \overline{p}_i = \frac{\exp(k \cdot \pi_i)}{\overline{s} + m_i}. \]

One can easily check that this algorithm requires linear time.

**Comment.** We cannot compute the desired bounds faster than in linear time since we need to process all \(2n\) inputs \(u_i\) and \(\pi_i\), and each elementary operation – arithmetic operation of an application of an elementary function like \(\exp(x)\) – can process at most two values. Thus, to process all \(2n\) inputs, we need at least \((2n)/2 = n\) computational steps. So, from the computational viewpoint, our algorithm is asymptotically optimal.
3 Possible Generalizations

What if we only know the value $k$ with interval uncertainty? Since we are taking into account that many values are known with uncertainty, it is reasonable to also consider the case when the value of the parameter $k$ is also known with interval uncertainty, i.e., when we only know the interval $[\underline{k}, \overline{k}]$ of possible values $k$. In this case, we should look for the range of values $p_i$ corresponding to all possible combinations of values $u_i$ and $k$ from the corresponding intervals.

In classification problems, we are mostly interested in the probability $p_i$ that the generated answer – that corresponds to the largest value of $u_i$ – is correct. For this value $i$, we can reformulate the expression (3) in the following equivalent form:

$$p_i = \frac{1}{1 + \sum_{j \neq i} \exp(k \cdot (u_j - u_i))}.$$  \hspace{1cm} (11)

Here, $u_i \geq u_j$ for all $j$, so $u_j - u_i \leq 0$.

- Thus, $\exp(k \cdot (u_j - u_i))$ decreases with $k$, so the sum of these terms also decreases with $k$, and so it the denominator of the expression (11).
- So, the fraction (11) increases with $k$.

Therefore:

- to compute the lower endpoint $p_i^\text{lower}$, it is sufficient to consider the smallest possible value of $k$, namely $k = \underline{k}$, and
- to compute the upper endpoint $p_i^\text{upper}$, it is sufficient to consider the largest possible value of $k$, namely $k = \overline{k}$.

So, we get the following formulas:

$$p_i^\text{lower} = \frac{\exp(\underline{k} \cdot u_i)}{\exp(\underline{k} \cdot u_i) + \sum_{j \neq i} \exp(\underline{k} \cdot u_j)};$$  \hspace{1cm} (12)

$$p_i^\text{upper} = \frac{\exp(\overline{k} \cdot u_i)}{\exp(\overline{k} \cdot u_i) + \sum_{j \neq i} \exp(\overline{k} \cdot u_j)}.$$  \hspace{1cm} (13)

Since we are only interested in computing the values $p_i^\text{lower}$ and $p_i^\text{upper}$ for one class $i$, we can simply follow these formulas and get a linear-time algorithm.

Comment. The possibility to have a linear-time algorithm depends on the fact that for the class $i$ with the largest value $u_i$, the probability $p_i$ monotonically depends on $k$ – namely, it increases with $k$. One can similarly show that for the smallest value $u_i$, we also have a monotonic dependence – namely, $p_i$ decreases with $k$. However, for intermediate values $u_i$, the dependence on $k$ is not necessarily monotonic, as the following simple example shows.

Let us take $u_1 = \ln(1) = 0 > u_2 = \ln(0.6) > u_3 = \ln(0.1)$. Then:
• For $k = 0$, we get $\exp(k \cdot u_i) = 1$ for all $i$, so

$$p_2(0) = \frac{1}{1+1+1} = \frac{1}{3} = 0.33\ldots$$

• For $k = 1$, we get $\exp(k \cdot u_1) = 1$, $\exp(k \cdot u_2) = \exp(\ln(0.6)) = 0.6$, and $\exp(k \cdot u_3) = \exp(\ln(0.1)) = 0.1$, so

$$p_2(1) = \frac{0.6}{1+0.1+0.6} = \frac{0.6}{1.7} = 0.35\ldots$$

• For $k \to \infty$, we get $\exp(k \cdot u_1) = 1$ while $\exp(k \cdot u_2)$ and $\exp(k \cdot u_3)$ tend to 0. So in the limit, we get

$$p_2(\infty) = \frac{0}{1+0+0} = 0.$$
What if we consider fuzzy uncertainty instead of interval uncertainty? In the previous text, we considered situations when the approximate values $\bar{x}$ come from measurements. There is another possibility: that the approximate values come from an expert estimate. Experts usually describe the accuracy of their estimates not in terms of precise bounds, but rather by using imprecise ("fuzzy") words from natural language, e.g., “the value is approximately 1 with accuracy about 0.1.” To describe such information in precise terms, Lotfi Zadeh came up with an idea that he called fuzzy logic; see, e.g., [1, 6, 15, 19, 20, 25]. Specifically, for each imprecise property like “approximately 1 with accuracy about 0.1,” he suggested to assign, to each real number $x$, the degree $\mu(x)$ – from the interval $[0, 1]$ – the degree to which this number $x$ satisfies this property, so that:

- 1 means that the expert is absolutely sure that $x$ satisfies this property,
- 0 means that the expert is absolutely sure that $x$ does not satisfy this property, and
- intermediate values mean that the expert is somewhat sure.

The function that assigns the degree $\mu(x)$ to each number $x$ is known as the membership function, or, alternatively, as the fuzzy set.

In this case, instead of intervals $[u_i, \bar{u}_i]$, we have fuzzy sets $\mu_i(u_i)$.

It is known that in general, application of an algorithm $y = F(u_1, \ldots, u_n)$ to fuzzy inputs $\mu_i(u_i)$ – that should result in a fuzzy set $\mu(y)$ – can be reduced to processing intervals if we use the following alternative representation of fuzzy sets. Namely, for each fuzzy set $\mu(x)$, for each $\alpha \in (0, 1]$, we can form an $\alpha$-cut

$$x(\alpha) \overset{\text{def}}{=} \{ x : \mu(x) \geq \alpha \}.$$ 

For $\alpha = 0$, the $\alpha$-cut is defined as $\{ x : \mu(x) > 0 \}$, where $\overline{S}$ means the closure of the set $S$, i.e., the set $S$ and all its limit points.

Once we know all the $\alpha$-cuts $x(\alpha)$, we can reconstruct the membership function as $\mu(x) = \sup\{ \alpha : x \in x(\alpha) \}$.

Then, it turns out that for each $\alpha$, the $\alpha$-cut $y(\alpha)$ of $y$ can be obtained by applying interval computations to the $\alpha$-cuts $u_i(\alpha)$:

$$y(\alpha) = \{ F(u_1, \ldots, u_n) : u_i \in u_i(\alpha) \text{ for all } i \}.$$ 

So, all we need to do is select, e.g., levels $\alpha = 0.0, 0.1, 0.2, \ldots, 0.9.1$. For each level, we apply the above algorithm to the $\alpha$-cuts $u_i(\alpha)$, and thus get the desired $\alpha$-cuts $p_i(\alpha)$ for the probabilities $p_i$.

4 Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-
2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology.

It was also supported by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

The authors are thankful to the anonymous referees for valuable suggestions.

References


