Symmetry Approach Explains Why Some Choices in Computational Intelligence Work Better

Olga Kosheleva and Vladik Kreinovich,
University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
olgak@utep.edu, vladik@utep.edu

1. Introduction

- One of the main objectives of computer science is to make computers help us – or even replace us – in performing intelligent tasks.
- How can we do it?
- If we want to design a machine that flies, a natural idea is:
  - to look at creatures that fly – birds and insects, and
  - to borrow some ideas from how they do it.
- Similarly, if we want to create a machine that thinks:
  - a natural idea is to borrow some idea from the creatures that can think,
  - which is us humans.
- We can use the ideas from the macro-level of this human thinking process.
- This leads us to logic – including fuzzy logic.

2. Introduction (cont-d)

- We can use the ideas from the micro-level of human thinking – i.e., from the level of the corresponding brain cells (neurons).
- This leads to neural networks.
- We can also use the ideas of how living creatures eventually evolved to the level when they are able to think.
- This leads to evolutionary computations.
- In all these areas, there have been many successes.
- However, just like an airplane is not simply a simulation of the bird, these successes are not the result of blind simulation of nature.
- To achieve each success, several different options have been tried, until finally one of the version led to a success.
- A natural question is: why these particular choices turned out to be successful while others were not as good?

3. Introduction (cont-d)

- In this talk, we show that the empirical success of many such choices can be explained in a similar way: by using ideas of symmetry.
- These ideas that have been very successful in modern physics and many other areas of human knowledge.
- Not only we explain why these methods are better.
- We confirm that these methods:
  - are indeed best of all, and
  - not just better than several dozen methods with which they have been compared.
- And in some cases, we come up with alternatives which are also potentially good and thus, worth trying.
- We also relate all this to the current challenge of making modern deep neural networks more intelligent.

4. Introduction (cont-d)

- Crudely speaking, a deep neural network performs gradient descent to come up with a solution.
- This works well if the corresponding function has only one local optimum.
- Such problems are usually feasible to solve.
- However, it is known that many practical problems are NP-hard and thus, not feasible.
- To solve these problems, we cannot start with a single point and then move to a local maximum.
- This is why neural networks do not work so well on these problems.

5. Introduction (cont-d)

- What we need is a multi-start optimization – we need to:
  - select several possible starting points (for which evolutionary computation is the most appropriate) and
  - only then use local search.
- So, we arrive at the idea of memetic algorithms which are so successfully used by Laszlo Koczy and his students and collaborators.
- If we can prove that the choices used in these methods are indeed the best and,
  - better yet, if we can find better methods,
  - this will make more confident that memetic methods will lead to future successes.

6. Methods and Results

- Our main idea is that:
  - while we want to process the values of the physical quantities, what we really process is their numerical values, and
  - these numerical values depend on the choice of the measuring unit and of the starting point.
- If we change the measuring unit and/or the starting point, all numerical values get transformed according to some linear transformation

$$x \mapsto a \cdot x + b.$$

- A classical example is transforming temperature from Celsius of Fahrenheit scale.
- It is natural to require that:
  - the optimality criterion – that enables us to select the best choice,
  - should not change if we simply re-scale all the quantities by applying the corresponding linear transformation.

7. Methods and Results (cont-d)

- The first case to which we apply this idea is the idea of interpolation.
- Interpolation is needed in fuzzy logic.
- Indeed, we can only elicit, from the expert, finitely many values of a membership function or of an "and" or "or" operation.
- To get all other values, we need to interpolate or extrapolate.
- The simplest case is when we need to interpolate based on two values at two points: $A = f(a)$ and $B = f(b)$.
- Of course, it makes sense to consider consistent interpolations, i.e., interpolations from which:
  - if we pick any other two points on the resulting function,
  - interpolation will lead to exactly the same function.
- Our first result is that for any scale-invariant interpolation optimality criteria, the optimal interpolation is linear.

8. Methods and Results (cont-d)

- This result explains why piece-wise linear membership are so successful – especially triangular and trapezoid ones.
- It also explains, indirectly, whey min, max, and $a+b-a\cdot b$ are among the most empirically successful "and"- and "or"-operations.
- This result also explains:
  - why piece-wise linear activation functions – such as Rectified Linear Units (ReLU) $max(0,x)$
  - have been so successful in deep learning.
- Similar ideas help to explain why softmax is so empirically successful in machine learning and in evolutionary computations:

$$p(x)=\frac{\exp(k\cdot f(x))}{\exp(k\cdot f(x))+\exp(k\cdot f(y))+\ldots}.$$

9. Methods and Results (cont-d)

- The explanation is that this method is optimal under any optimality criteria that is shift-invariant.
- In other words, it is invariant under transformation $f(x) \mapsto f(x) + f_0$.
- If instead we postulate invariance under re-scaling $f(x) \mapsto m \cdot f(x)$, we get a different function that has also been successful in many cases.

10. Technical Details: General Lemma

- By an *optimality criterion*, we mane a pre-order (transitive and reflexive) relation $\succeq$ on the set $\mathcal{A}$ of all alternatives.
- The criterion is *final* if there is exactly one optimal alternatives, i.e., $a_{\rm opt}$ for which $a_{\rm opt} \succeq a$ for all a.
- Let G be a group of transformation $G : \mathcal{A} \mapsto \mathcal{A}$.
- The criterion is *G-invariant* if $a \succeq b$ implies $g(a) \succeq g(b)$ for all $g \in G$.
- *Lemma:* if $\succeq$ is G-invariant and final, then $a_{\rm opt}$ is also G-invariant, i.e., $g(a_{\rm opt}) = a_{\rm opt}$ for all $g \in G$.
- *Proof:* since $a_{\rm opt}$ is optimal, we have $a_{\rm opt} \succeq g^{-1}(a)$ for all a.
- Since $\succeq$ is G-invariant, we conclude that $g(a_{\rm opt}) \succeq a$ for all a.
- This means that $g(a_{\rm opt})$ is optimal.
- But since the criterion is final, there is only one optimal alternative, so $g(a_{\rm opt}) = a_{\rm opt}$.

11. Technical Details: Case of Interpolation

- Scaling transformations are $x \mapsto a_0 + a_1 \cdot x$ and $y \mapsto b_0 + b_1 \cdot y$.
- Let $y = f_0(x)$ be an interpolation based on $f(0) = 0$ and $f(1) = 1$.
- A general case $f(x_1) = y_1$ and $f(x_2) = y_2$ can be obtained from this one by $x \mapsto X = x_1 + (x_2 - x_1) \cdot x$ and $y \mapsto Y = y_1 + (y_2 - y_1) \cdot y$.
- So, due to invariance, we get

$$f(x) = y_1 + (y_2 - y_1) \cdot f\left(\frac{x - x_1}{x_2 - x_1}\right).$$

- In particular, if we interpolate based on $x_1 = y_1 = 0$, $x_2 = x_0$ and $y_2 = f_0(x_0)$, we get

$$f(x) = f_0(x_0) \cdot f_0\left(\frac{x}{x_0}\right).$$

- Consistency means that $f(x) = f_0(x)$.
- For $z \stackrel{\text{def}}{=} x/x_0$, we have $x = x_0 \cdot z$, so $f_0(x_0 \cdot z) = f_0(x_0) \cdot f_0(z)$.

12. Technical Details: Case of Interpolation (cont-d)

- It is known that every continuous (even measurable) function $f_0(x)$ with this property is $f_0(x) = x^\alpha$ for some α.
- For $x_1 = x_0$, $y_1 = f_0(x_0)$, $x_2 = y_2 = 1$, we similarly get

$$f_0(x) = f_0(x_0) + (1 - f_0(x_0)) \cdot f_0\left(\frac{x - x_0}{1 - x_0}\right).$$

- Since $f_0(x) = x^\alpha$, this means

$$x^\alpha = x_0^\alpha + (1 - x_0^\alpha) \cdot \frac{(x - x_0)^\alpha}{(1 - x_0)^\alpha}.$$

- Differentiating both sides by x, we get

$$\alpha \cdot x^{\alpha-1} = (1 - x_0^\alpha) \cdot \alpha \cdot \frac{(x - x_0)^{\alpha-1}}{(1 - x_0)^\alpha}.$$

13. Technical Details: Case of Interpolation (cont-d)

- For $x = 0$, when $\alpha > 1$, the left-hand side is 0 but the right-hand side is not 0.
- For $x = 0$, when $\alpha < 1$, the left-hand side is ∞ but the right-hand side is not ∞.
- So, the only possible case is $\alpha = 1$, hence $f_0(x) = x$ and we get linear interpolation.

14. Technical Details: "And"- and "Or"-Operations

- Let us consider "and"-operations $f(a,b)=f(b,a)$; for "or", the proof is similar.
- We want to make sure that $a\mapsto f(a,b)$ is linear for all b.
- Let us assume that all we know is that $f(0,0)=f(0,1)=f(1,0)=0$ and $f(1,1)=1$.
- Then, by linear interpolation, $f(0,b)=0$ and $f(1,b)=b$ for all b.
- Now, linear interpolation for a leads to $f(a,b)=a\cdot b$.
- What if we also know that $f(b,b)=b$ for all b?
- Then for $a<b$, linear interpolation between $f(a,a)=a$ and $f(a,1)=a$ leads to $f(a,b)=a=\min(a,b)$.
- For "or", we similarly get $f(a,b)=a+b-a\cdot b$ and $f(a,b)=\max(a,b)$.

15. Technical Details: Activation Function $s(x)$

- If we only know two values of $s(x)$, the interpolation leads to a linear function.
- But we want a non-linear activation function – otherwise, we cannot represent non-linear functions.
- If we know three values, we get a piecewise-linear function.
- All such 2-part functions are linearly equivalent to ReLU.
- If midpoint is 0, then, in general, we get a *leaky ReLU*.

16. Technical Details: Softmax

- We consider general expressions

$$p(x) = \frac{f(J(x))}{\sum\limits_{y} f(J(y))}.$$

- For two values $A = J(x_1)$ and $B = J(x_2)$, shift-invariance means that

$$\frac{f(A)}{f(A) + f(B)} = \frac{f(A + f_0)}{f(A + f_0) + f(B + f_0)}.$$

- If we take inverse of both sides and subtract 1 from both sides, then:

$$\frac{f(B)}{f(A)} = \frac{f(B + f_0)}{f(A + f_0)}.$$

- Multiplying both sides by $f(A + f_0)/f(B)$, we get

$$\frac{f(A + f_0)}{f(A)} = \frac{f(B + f_0)}{f(B)}.$$

17. Technical Details: Softmax (cont-d)

- The value of this expression E is the same for all A, so it only depends on f_0: $E = E(f_0)$:

$$\frac{f(A+f_0)}{f(A)} = E(f_0), \text{ i.e. }, f(A+f_0) = E(f_0)\cdot f(A).$$

- It is know that the only continuous solution to this functional equation is $f(A) = c\cdot\exp(k\cdot A)$.
- The ration $p(x)$ does not change if we divide both numerator and denominator by c, so we get the usual softmax expression.
- If we use scale-invariance instead of shift-invariance, then we similarly get

$$\frac{f(A)}{f(A)+f(B)} = \frac{f(m\cdot A)}{f(m\cdot A)+f(m\cdot B)} \text{ and } p(x) = \frac{x^\alpha}{\sum\limits_y y^\alpha}.$$

- This expression is also often useful.

18. Acknowledgments

This work was supported in part:

- by the US 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),
- by the AT&T Fellowship in Information Technology, and
- by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Focus Program SPP 100+ 2388, Grant Nr. 501624329,