Repairing Learned Controllers with Convex Optimization: A Case Study

Dario Guidotti[1(✉)], Francesco Leofante[1], Claudio Castellini[2], and Armando Tacchella[1]

[1] DIBRIS, Università degli Studi di Genova, Via all'Opera Pia, 13, 16145 Genoa, Italy
{dario.guidotti,francesco.leofante}@edu.unige.it, armando.tacchella@unige.it

[2] Institute of Robotics and Mechatronics, German Aerospace Center, Münchener Straße, 20, Oberpfaffenhofen, 82234 Weßling, Germany
claudio.castellini@dlr.de

Abstract. Despite the increasing popularity of Machine Learning methods, their usage in safety-critical applications is sometimes limited by the impossibility of providing formal guarantees on their behaviour. In this work we focus on one such application, where Kernel Ridge Regression with Random Fourier Features is used to learn controllers for a prosthetic hand. Due to the non-linearity of the activation function used, these controllers sometimes fail in correctly identifying users' intention. Under specific circumstances muscular activation levels may be misinterpreted by the method, resulting in the prosthetic hand not behaving as intended. To alleviate this problem, we propose a novel method to verify the presence of this kind of intent detection mismatch and to repair controllers leveraging off-the-shelf LP technology without using additional data. We demonstrate the feasibility of our approach using datasets gathered from human participants.

1 Introduction

In the last few years Machine Learning techniques proved to be successful in many domains of application such as image classification [1] or speech recognition [2], with some architectures even claiming to match the cognitive abilities of humans [3]. Despite this popularity, the usage of Machine Learning (ML) methods in safety-critical applications is still sometimes limited by the absence of effective methods to provide formal guarantees on their behavior. In this paper we focus on one such safety-critical application, where Kernel Ridge Regression with Random Fourier Features [4,5] is used to learn controllers for prosthetic hands. In this framework, multi-fingered, self-powered prosthetic hands [6] are controlled using signals generated from a certain number of surface electromyography [7] sensors. Ensuring the correct behavior of the controller is critical to the safety of the amputee wearing the robotic artifact.

L.-M. Rousseau and K. Stergiou (Eds.): CPAIOR 2019, LNCS 11494, pp. 364–373, 2019.
https://doi.org/10.1007/978-3-030-19212-9_24

Although several approaches have been proposed to build ML models enforcing reliable myocontrol [8,9], reliability is still an issue. In particular we tackle a problem of *intent mismatch*: whenever a subject increases her muscle activation, the prosthesis should in turn increase the applied force. However, if the ML model is a non-linear one, there is no guarantee that this will happen. To reduce the chance of intent mismatch, we propose an approach that couples a standard ML-based myocontrol system with a Linear Programming (LP) solver to *automatically repair* and improve the learned model *without requiring additional data*. This last point is crucial as gathering more data from the subject to amend the ML model is not desirable: to gather relevant data, the subject would need to apply a large amount of force leading to muscle strain, fatigue and frustration.

By leveraging LP technology, we can represent the ML model, as well as the property of interest, as a set of arithmetic constraints and establish algorithmically whether a controller satisfies the given property. If the property is not satisfied, a LP solver is used to iteratively repair the controller until the resulting model is mathematically guaranteed to be safe.

To demonstrate the feasibility of our approach, we compare results obtained with a standard ML-based myocontroller against a myocontroller that was repaired using our methodology on datasets gathered from human subjects at the German Aerospace Center. Remarkably, we show not only that LP-based repair is effective, but it also comes at a reasonably low computational price.

2 Preliminaries

Kernel Ridge Regression with Random Fourier Features. KRR-RFF with Tikhonov regularization [4] has been demonstrated multiple times in literature to match most of the requirements of myocontrol [8,9]. Ridge Regression (RR) builds a linear model of the input space data $\mathbf{x}$ of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is the vector of weights computed as a result of the training phase. KRR-RFF modifies standard RR by introducing a feature map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ that maps a sample $\mathbf{x} \in \mathbb{R}^d$ onto a D-dimensional space as shown in Eq. 1

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T\Phi(\mathbf{x}) \\ \Phi(\mathbf{x}) &= \sqrt{2}\cos(\boldsymbol{\Omega}\mathbf{x} + \boldsymbol{\beta}) \end{aligned} \tag{1}$$

where $\boldsymbol{\Omega}$ is a $D \times d$ matrix and $\boldsymbol{\beta}$ is a D-dimensional vector, whose values are drawn from a normal distribution and a uniform distribution from 0 to 2π, respectively – we refer the reader to, e.g., [4] for more details.

This method can be seen as a finite-(D-)dimensional approximation to a RBF-kernel Least-Squares SVM [10], therefore it can be made arbitrarily accurate by tuning D; nevertheless, since its kernel is finite-dimensional and can be explicitly written, it enjoys most useful properties of Ridge Regression such as, e.g., space-boundedness (making it ideal for online learning) and extremely fast training and testing. On top of this, using a rank-1 update method such as, e.g., the Sherman-Morrison formula, it can be made incremental, paving the way to interactive myocontrol.

Myocontrol and Intent Detection. Natural, simultaneous and proportional myocontrol is an instance of (multi-variate) regression. Let $S = (X, Y)$ be a dataset composed by a set of observation $X = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, N\}$ and a set of corresponding target values $Y = \{\mathbf{y}_i \in \mathbb{R}^m \mid \forall \mathbf{x}_i \in X\}$. Each observation consists of d features evaluated from a set of sensors and denotes the muscular activation corresponding to an action (e.g., wrist flexion, power grasp, etc.); each associated target value, in turn, is a vector of m motor activation values (currents, torques, ...) for a prosthetic device and corresponds to the desired action as enacted by the device itself. In practice S is built by gathering, for each desired action, an adequate number of observations recorded while the subject is stimulated to perform it; each such observation is then coupled with the target value enforcing the action by the prosthetic device. For instance, the subject is asked to power grasp ("make a fist"); once the experimenter verifies that the signals have reached a stable pattern which is also sufficiently distinct from the baseline, a representative amount of observations is recorded and associated to (synthetic) target values denoting maximal activation of all fingers. At the end of the data gathering phase, S consists of one or more observation clusters for each action considered, coupled with adequate target values – see, e.g., [11] for more details.

KRR-RFF builds an approximant function $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^m$ which best fits S and offers the best generalization power on so-far unseen data. The approximant f can be seen as an *intent detector*: whenever the wearer's muscles are activated to enforce a specific action, the prosthesis should behave accordingly. If properly built out of S, f will smoothly and timely activate every motor of the prosthesis whenever required; minimal and maximal muscular activations, as gathered from the subject, will correspond to minimal and maximal motor activations; under plausible assumptions, intermediate activation values too will be correctly predicted in a monotonically-increasing fashion [4,8,11]. Finally, increasing the muscle activation beyond the maximal values obtained during data gathering should correspond to coordinated increased activation of the motors of the prosthesis, but this cannot be guaranteed by learning techniques only.

Linear Programming. Linear Programming solves linear problems over a set of *decision variables*, a set of *linear constraints* over these variables and a *convex objective function* that is linear in the decision variables [12].

Let $x_1, \ldots, x_n$ be a set of decision variables, a general linear program can be written as

$$\begin{aligned} \text{minimize} \sum_{i=1} c_i x_i \\ \text{subject to} \sum_{i=1} a_{ij} x_i \geq b_j, j \in [1, m] \\ x_i \in \mathbb{R}, i \in [1, n] \end{aligned}$$

Values c_i, a_{ij} and b_j are constants that are specified during problem formulation. General approaches to solving LPs include methods such as *simplex* or the *ellipsoid* – for further details see, e.g., [12].

Verification and Repair of Learned Models. Several approaches have been proposed to verify different classes of ML models automatically. Even though such approaches might differ in several aspects, most of them consider trained models, i.e., they do not intermingle with the learning process, and they seek a transformation from the ML model to a decision or optimization problem in some constraint system – see, e.g., [13] for a recent account on the subject. For instance, in [14] Boolean satisfiability solvers are proposed to verify robustness of Binarized Neural Networks. *Satisfiability Modulo Theories* engines are leveraged to verify neural networks in, e.g., [15,16] and Support Vector Machines in [17]. MILP-based approaches have been proposed to verify neural networks by, e.g., [18,19], while a combination of SAT and LP techniques is used by [20] for the same purpose. A different approach is taken in [21], where abstract interpretation is used to certify safety and robustness of Deep Neural Networks with ReLU and max pooling layers. Repair has received less attention compared to verification, with [15] and [17] being the only contributions in this direction. However, the repair they propose involves retraining the ML model with data generated by solvers.

3 Verification and Repair

In order to enhance the reliability of the controllers we consider in this work, we propose an automated procedure that iteratively brings the controller to a condition where intent detection mismatch does not occur anymore. In the following, we describe the assumptions on which our approach rests, show how our ideas can be formalized into an executable algorithm and provide experimental evidence to support the approach we propose.

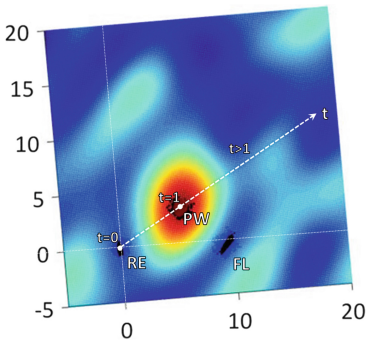


Fig. 1. The heat map representing the value of the approximant for power grasping f_{PW}, learned using readings from two sensors.

3.1 A Linear Model of Intent Detection

The data gathering procedure produces a cluster of samples for each action of interest. An example is shown in Fig. 1, where observations are gathered for three actions: rest (RE), power grasp (PW) and wrist flexion (FL). For a specific action, we assume that values of the input space lying on the line joining the resting cluster and the cluster corresponding to the action of interest denote an execution of the action with increasing strength (see Fig. 1). This assumption is justified by physiological reasons: muscle activation remains roughly coordinated as the action is performed with more force (see, e.g., [22–26]).

More precisely, for a generic action a, let $\mathbf{x}_a$ denote the coordinates corresponding to the average of points belonging to the cluster corresponding to a.

Performing an action a with increasing strength can be seen as moving along the line $\mathbf{x}_{RE} + (\mathbf{x}_a - \mathbf{x}_{RE})t_a$, where $t_a \in \mathbb{R}_{\geq 0}$[1] is a parameter proportional to the muscle activation value. By construction, $t_a = 1$ corresponds to the maximum activation value registered during the data gathering phase. Figure 1 shows an example for $a = PW$: for $t_{PW} = 0$, the subject is at rest; as t_{PW} increases the subject starts power-grasping moving along $\mathbf{x}_{RE} + (\mathbf{x}_{PW} - \mathbf{x}_{RE})t_{PW}$. Under this assumption, absence of intent mismatch for action a can be encoded as follows:

$$\forall t_a.(\mathbf{x} = \mathbf{x}_{RE} + (\mathbf{x}_a - \mathbf{x}_{RE})t_a \wedge t_a > 1) \implies f_a(\mathbf{x}) \geq a_{max} \tag{2}$$

where $f_a(\mathbf{x})$ is the value of the approximant function for a applied to the input $\mathbf{x}$ and a_{max} is the full activation value (e.g., the value corresponding to the full closure of the hand when the action is PW). Therefore, a mismatch happens whenever the value of muscular activation t_a is greater than 1 (i.e., the maximum value observed during data gathering) and the corresponding motor activation value predicted by the approximant f is less than a_{max}.

3.2 The Algorithm

Our repair procedure LP-Repair leverages training data and our prior knowledge about the problem in order to modify the parameters of the machine learning model used in the controller. In particular, LP-Repair analyses the initial model in order to determine if there exists an unsafe point, i.e., an instance of intent detection mismatch. Such point is considered to create additional constraints for an optimization problem which, once solved, gives as solution a new set of parameters for the machine learning model. This constraint generation approach is justified by the fact that, in the feature space, the model is a linear function of the weights as per Eq. (1). The resulting model is guaranteed to fix activation insofar the unsafe point is concerned. The procedure proceeds iteratively verifying safety of the modified controller and, possibly, adding a new constraint (corresponding to a new unsafe point) to the problem; when the controller is deemed safe, the procedure ends.

Algorithm 1 shows the pseudocode of LP-Repair: X and Y are, respectively, the inputs and outputs training data of our controller, w is the vector of the weights of the machine learning model and A is a set of actions of interest. **safetyCheck**(...) uses a set of points, in the original input space, with uniform distances along the straight lines of interest in order to find the most unsafe point for each action, i.e., the point whose output has the highest difference from the expected value. This function returns a boolean variable $safe_a$ (which is false if an unsafe point has been found) and an unsafe point $unsafePoint_a$.

The procedure **Opt**(...) defines and solves the optimization problem presented in Fig. 2, where w_i is the i-th component of the vector of the weights, and ϵ_i is the variation on the above mentioned component. D is the dimensionality of the feature space, N is the number of samples in the dataset, and η corresponds to

[1] In practice, the value of t_a is upper bounded by operating range of EMG sensors.

Algorithm 1. Repair procedure

```
1: procedure LP-Repair(X, Y, w, A)
2:     safe ← False
3:     ŵ ← w
4:     unsafeSet ← ∅
5:     while not safe do
6:         safe ← True
7:         for each action a ∈ A do
8:             (safe_a, unsafePoint_a) ← safetyCheck(X, Y, ŵ, a)
9:             if not safe_a then
10:                safe ← False
11:                unsafeSet ← (unsafeSet ∪ unsafePoint_a)
12:                ŵ ← Opt(X, Y, ŵ, unsafeSet)
13:    return ŵ
```

the sampling rate which determines the subset of the training set we consider in the cost function. x_i^{rest} is the i-th component of the data-point corresponding to the center of the rest cluster, and $x_i^{act_a}$ is the i-th component of the data-point corresponding to the center of the cluster associated to the activation of a certain action a. Finally, $x_i^{unsafe_j}$ denotes the i-th component of the j-th unsafe point.

In more detail, Eq. 3a defines the cost function to be minimized, where: *(i)* the first member corresponds to the modification of the parameters of the controllers (w_i), *(ii)* the second member tries to minimize the error on a subset of the training set, and we do this in order to guarantee that the prediction performances are not degraded and *(iii)* the third member is a tolerance on the error δ. Equations 3b and 3c encode the constraints to guarantee the correct output value respectively for the center of the rest cluster and the centers of the activation clusters. Finally, Eq. 3d presents the constraints we use to force the output values of the unsafe points to be correct, where k is the total number of unsafe points found. It is important

$$\min \sum_{i=1}^{D} |\epsilon_i| + \sum_{l=1}^{floor(N/\eta)} \left| Y^{l\cdot\eta} - \sum_{i=1}^{D} (w_i + \epsilon_i) X_i^{l\cdot\eta} \right| + \delta \tag{3a}$$

$$\sum_{i=1}^{D} (w_i + \epsilon_i) x_i^{rest} = 0 \tag{3b}$$

$$a_{max} - \delta \leq \sum_{i=1}^{D} (w_i + \epsilon_i) x_i^{act_a} \leq a_{max} + \delta \quad \forall a \in A \tag{3c}$$

$$\sum_{i=1}^{D} (w_i + \epsilon_i) x_i^{unsafe_j} \geq a_{max}, \quad \forall j = 1, ..., k \tag{3d}$$

Fig. 2. LP model used for repair.

to highlight that, since all the points we consider in the problem are given, i.e., data-points are not variables, we can apply the feature mapping beforehand and formulate the problem of repairing a non-linear classifier without introducing non-linear constraints. For this reason, all the data-points in Eq. 3 are data-points in the feature space, not in the input space.

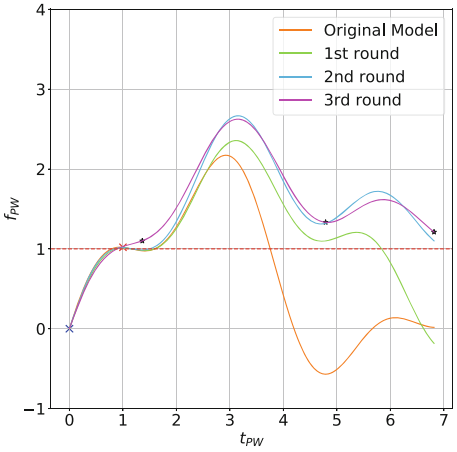
3.3 Experimental Results

A prototypical implementation of our approach was tested[2] on 18 datasets, each consisting of 320 samples, gathered from human subjects. The actions of interests are rest, power grasp, wrist flexion and wrist extension. In all our tests our procedure managed to terminate successfully, e.g. it managed to bring the controller to a safe configuration. In Fig. 3a we show an example of how the output of the controller for power-grasping is modified by LP-Repair. As it can be seen, at the last iteration, the output is greater than one[3] for all admissible values of t above a_{max}.

In Fig. 3b we show the quantities of interest in our analysis: the sampling rate (S.R.) is the rate at which we pick samples from the training data to use in the cost function. D is the parameter which determines the dimension of the feature space. MSE-o (resp. MSE-r) is the mean square error computed on the training set before (resp. after) the repair process. Time corresponds to the CPU time needed for the repair process (in seconds). MSE and time values are both computed as means on 18 datasets. We do not display the last three rows of the Table when the sampling rate is 100 because the repair procedure did not complete successfully, i.e., none of the 18 datasets yields a successful repair due to conditioning problems reported by the LP engine. We have chosen to compare MSEs before and after repair in order to verify that our repair process does not degrade the performances of the controller substantially. As it can be seen from Fig. 3b, the time complexity of the problem grows monotonically with the sampling rate and the dimension of the feature space, but on average, the runtime of the repair procedure is always less than 3 CPU minutes. In particular, for values of D which are relevant from an engineering point of view, i.e., those that yield MSE errors of less than 0.1%, we notice that the repair procedure is feasible for all the sampling rates considered, and that the final accuracy, albeit decreased, is still viable for practical applications. This is more evident as D increases because as the representativity increases the repair process can modify the controller without decreasing too much the accuracy of the prediction.

[2] We implemented our procedure using Python version 2.7. and the libraries *sklearn* and *cvxopt* for learning and optimization respectively. The default solver of *cvxopt*, i.e., *conelp*, was used – see [27] for more details. All the experiments are capped at 10 min of CPU time and 4 GBs of memory; experiments ran on a Ubuntu 18.04 machine equipped with a quad-core i5 Intel CPU running at 2.60 GHz.

[3] Notice that for power-grasping a_{max} is equal to one.



(a)

S. R.	D	MSE-o	MSE-r	Time
10	10	0.0140	0.4370	6.967s
10	50	0.0020	0.0040	18.456s
10	100	0.0012	0.0025	39.752s
10	150	0.0008	0.0019	68.984s
10	200	0.0007	0.0019	103.650s
10	250	0.0006	0.0015	138.009s
10	300	0.0005	0.0012	179.034s
50	10	0.0146	0.5177	4.880s
50	50	0.0021	0.0282	14.988s
50	100	0.0011	0.0151	26.941s
50	150	0.0009	0.0216	41.134s
50	200	0.0007	0.0195	61.431s
50	250	0.0006	0.0134	83.657s
50	300	0.0005	0.0157	110.448s
100	10	0.0157	0.9239	4.041s
100	50	0.0019	0.0648	13.461s
100	100	0.0011	0.0517	25.440s
100	150	0.0009	0.0349	36.183s

(b)

Fig. 3. Results for power-grasping.

4 Conclusion

Our paper provides empirical evidence that convex optimization techniques can be used to *repair* machine learned controllers for prosthetic hands insofar as detection mismatch is concerned and controllers are learned using KRR-RFF. The key factors of our successful evaluation are (*i*) a physiology-rooted modeling of intent detection mismatch along regions that conjoin data clusters corresponding to actions to those corresponding to rest conditions and (*ii*) a formalization of the repair problem based on feature space rather than input space. The former allows us to mathematically define intent detection mismatch, while the latter allows us to solve the problem using a linear program, rather than a non-linear one. The main feature of our method is that it repairs the controller *without requiring additional data* to be gathered. This is very important in applications involving human subjects, where additional data acquisition can be time-consuming, expensive or just plain impossible. While our method is effective on a specific case study, we expect that our findings can help approach the problem in different contexts where KRR-RFF is viable, as well as provide guidance to the repair of controllers learned with different methods: our future work includes furthering research along these directions.

References

1. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: closing the gap to human-level performance in face verification. In: CVPR, pp. 1701–1708 (2014)
2. Yu, D., Hinton, G.E., Morgan, N., Chien, J.-T., Sagayama, S.: Introduction to the special section on deep learning for speech and language processing. IEEE Trans. Audio Speech Lang. Process. **20**(1), 4–6 (2012)

3. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015)
4. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS, pp. 1177–1184 (2008)
5. Gijsberts, A., Metta, G.: Incremental learning of robot dynamics using random features. In: ICRA, pp. 951–956 (2011)
6. Fougner, A., Stavdahl, Ø., Kyberd, P.J., Losier, Y.G., Parker, P.A.: Control of upper limb prostheses: terminology and proportional myoelectric control - a review. IEEE Trans. Neural Syst. Rehabil. Eng. **20**(5), 663–677 (2012)
7. Merletti, R., Botter, A., Cescon, C., Minetto, M.A., Vieira, T.M.M.: Advances in surface EMG: recent progress in clinical research applications. Crit. Rev. Biomed. Eng. **38**(4), 347–379 (2011)
8. Gijsberts, A., et al.: Stable myoelectric control of a hand prosthesis using non-linear incremental learning. Front. Neurorobot. **8** (2014)
9. Strazzulla, I., Nowak, M., Controzzi, M., Cipriani, C., Castellini, C.: Online bimanual manipulation using surface electromyography and incremental learning. IEEE Trans. Neural Syst. Rehabil. Eng. **25**(3), 227–234 (2017)
10. Gestel, T.V., et al.: Benchmarking least squares support vector machine classifiers. Mach. Learn. **54**(1), 5–32 (2004)
11. Patel, G., Nowak, M., Castellini, C.: Exploiting knowledge composition to improve real-life hand prosthetic control. IEEE Trans. Neural Syst. Rehabil. Eng. **25**(7), 967–975 (2017)
12. Schrijver, A.: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, Hoboken (1999)
13. Leofante, F., Narodytska, N., Pulina, L., Tacchella, A.: Automated verification of neural networks: advances, challenges and perspectives. arXiv preprint arXiv:1805.09938 (2018)
14. Narodytska, N., Kasiviswanathan, S.P., Ryzhyk, L., Sagiv, M., Walsh, T.: Verifying properties of binarized deep neural networks. In: AAAI, pp. 6615–6624 (2018)
15. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 243–257. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_24
16. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
17. Leofante, F., Tacchella, A.: Learning in physical domains: mating safety requirements and costly sampling. In: Adorni, G., Cagnoni, S., Gori, M., Maratea, M. (eds.) AI*IA 2016. LNCS (LNAI), vol. 10037, pp. 539–552. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49130-1_39
18. Cheng, C.-H., Nührenberg, G., Ruess, H.: Maximum resilience of artificial neural networks. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 251–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_18
19. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3), 296–309 (2018)
20. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 269–286. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_19

21. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: AI2: safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP) (2018)
22. Valero-Cuevas, F.J.: Predictive modulation of muscle coordination pattern magnitude scales fingertip force magnitude over the voluntary range. J. Neurophysiol. **83**(3), 1469–1479 (2000)
23. Poston, B., Danna-Dos Santos, A., Jesunathadas, M., Hamm, T.M., Santello, M.: Force-independent distribution of correlated neural inputs to hand muscles during three-digit grasping. J. Neurophysiol. **104**(2), 1141–1154 (2010)
24. de Rugy, A., Loeb, G.E., Carroll, T.J.: Muscle coordination is habitual rather than optimal. J. Neurosci. **32**(21), 7384–7391 (2012)
25. He, J., Zhang, D., Sheng, X., Li, S., Zhu, X.: Invariant surface emg feature against varying contraction level for myoelectric control based on muscle coordination. IEEE J. Biomed. Heal. Inform. **19**(3), 874–882 (2015)
26. Al-Timemy, A.H., Khushaba, R.N., Bugmann, G., Escudero, J.: Improving the performance against force variation of emg controlled multifunctional upper-limb prostheses for transradial amputees. IEEE Trans. Neural Syst. Rehabil. Eng. **24**(6), 650–661 (2016)
27. Vandenberghe, L.: The CVXOPT linear and quadratic cone program solvers (2010)