Artificial IntelliDance: Teaching Machine Learning through a Choreography

Apoorv Agarwal
Department of Computer Science
Columbia University, New York, USA
apoorv@cs.columbia.edu

Caitlin Trainor
Department of Dance
Barnard College, Columbia University
caitlinmarytrainor@gmail.com

Abstract

In this paper we present a choreography that explains the process of supervised machine learning. We present how a perceptron (in its dual form) uses convolution kernels to learn to differentiate between two categories of *objects*. Convolution kernels such as string kernels and tree kernels are widely used in Natural Language Processing (NLP) applications. However, the baggage associated with learning the theory behind convolution kernels, which extends beyond graduate linear algebra, makes the adoption of this technology intrinsically difficult. The main challenge in creating this choreography was that we were required to represent these mathematical equations at their *meaning* level before we could translate them into the language of movement. By orchestrating such a choreography, we believe, we have obviated the need for people to posses advanced math background in order to appreciate the core ideas of using convolution kernels in a supervised learning setting.

1 Introduction

Natural Language Processing (NLP) and Machine Learning (ML) are making a significant impact in our day to day lives. Advancement in these areas of research is changing the way humans interact with each other and with objects around them. For example, speech to speech translation is making it possible for people speaking different languages to communicate seamlessly.[1] In this eco-system, where machines and objects around us are becoming smarter, there is a need to make this complex technology available to a general audience.

The Dance Your PhD competition[2] is a recent effort that encourages doctoral students pursuing research in Physics, Chemistry, Biology and Social Sciences to explain the scientific ideas in their theses through movement. The main advantage of this approach is that the scientific ideas become available to a general audience through a medium that is both visual and entertaining. The main challenge, of course, is to abstract away from the technical vocabulary and physicalize these scientific ideas.

In this paper, we present a choreography that explains the process of learning from data in a supervised setting. Through this choreography, we bring out some of the main ideas of supervised machine learning, including representing data as structured objects and formulating similarity functions that a machine uses to calculate distances between data points. While these are general ideas, more relevant to an audience that is not familiar with machine learning, the choreography may also be used for explaining convolution kernels to researchers familiar with machine learning but not necessarily familiar with how a perceptron uses a convolution kernel in its dual form.

The main challenge in creating this choreography was that we were required to represent these mathematical equations at the *meaning* level before translating them into the language of movement. In doing so, our primary concerns were accuracy, aesthetics, and legibility. The scientific ideas at hand could not be compromised, and yet a literal representation of the symbols would negate the intent of the project.

[1] http://www.bbn.com/technology/speech/speech_to_speech_translation

[2] http://gonzolabs.org/dance/

Proceedings of the Fourth Workshop on Teaching Natural Language Processing, pages 51–55,
Sofia, Bulgaria, August 4-9 2013.

Equally vital to the success of the piece is the quality of the choreography on its own formal and aesthetic terms. The challenge of the translation was both critical to the process and also enriching, because it deepened our understanding of convolution kernels.

As Jason Eisner correctly notes in his paper on interactive spreadsheets for teaching the forward-backward algorithm (Eisner, 2002) – *They are concrete, visual, playful, sometimes interactive, and remain available to the students after the lecture ends* – we believe this choreography shares the same spirit. Artificial IntelliDance functions to explain a relatively sophisticated machine learning paradigm in an accessible and entertaining format that can be viewed repeatedly.[3]

The rest of the paper is structured as follows: In section 2, we review the perceptron algorithm, its dual form and convolution kernels. In section 3 we present details of the choreography, focusing on the aspects that explain the process of supervised machine learning and bring out the strengths and weaknesses of kernel learning. We conclude in Section 4.

2 The Perceptron algorithm and Convolution Kernels

The perceptron algorithm is an online learning algorithm invented by Frank Rosenblatt in 1958 (Rosenblatt, 1958). Given a set of training data points, $D = \{(\mathbf{x_i}, y_i)\}$, where $y_i \in \{1, -1\}$, the algorithm works as follows:[4]

1. Start with the all-zeroes weight vector $\mathbf{w_1} = 0$, and initialize t to 1.
2. Given $\mathbf{x_i}$, predict positive if $\mathbf{w_t} \cdot \mathbf{x_i} > 0$
3. On a mistake, update as follows:

   $\mathbf{w_{t+1}} \leftarrow \mathbf{w_t} + y_i \mathbf{x_i}$
4. $t \leftarrow t + 1$

In natural language, a perceptron maintains a weight vector $\mathbf{w_t}$ at time instance t. The weight vector is initialized to zero at the start of the algorithm. The perceptron receives one data point after the other. For each data point, it predicts the category of the data point by calculating its dot product with the weight vector. If the dot product is greater than zero, it predicts the category of the data point as 1, and -1 otherwise. On a mistake, the perceptron updates the weight vector by adding the product of the data point ($\mathbf{x_i}$) and its category (1 or -1).

The key idea here is that the weight vector is a linear combination of the training data points whose categories the perceptron predicted incorrectly at the time of training. The algorithm *remembers* these incorrectly classified data points by *marking* them with their true category (1 or -1). Abusing terminology, we refer to these incorrectly classified training data points as support vectors. Notationally, the final weight vector then is $\mathbf{w} = \sum_{k=1}^{N_s} y_k \mathbf{x_k}$, where N_s is the number of support vectors.

This simple fact that the weight vector is a linear combination of the data points has a deeper consequence – to predict the category of an unseen example, call it $\mathbf{x}$, all we need is a dot product of $\mathbf{x}$ with all the support vectors: $\mathbf{w} \cdot \mathbf{x} = \sum_{k=1}^{N_s} y_k (\mathbf{x_k} \cdot \mathbf{x})$. This is usually referred to as the dual form of the perceptron. The dual form allows for the use of kernels because the dot product between two examples can be replaced by a kernel as follows: $\mathbf{w} \cdot \mathbf{x} = \sum_{k=1}^{N_s} y_k K(\mathbf{x_k}, \mathbf{x})$. This is exactly where convolution kernels come into the picture. We review those next.

Convolution kernels, first introduced by David Haussler (1999), can be viewed as functions that calculate similarities between abstract objects, $K : X \times X \rightarrow \mathbb{R}$, where X is the set of abstract objects. Since their introduction, convolution kernels have been widely used in many NLP applications (Collins and Duffy, 2002; Lodhi et al., 2002; Zelenko et al., 2003; Culotta and Sorensen, 2004; Moschitti, 2004; Zhou et al., 2007; Moschitti et al., 2008; Agarwal and Rambow, 2010; Agarwal et al., 2011). The reason for their popular use in NLP applications is that text has *natural* representations such as strings, trees, and graphs. Representing text in its natural representation alleviates the need for fine-grained feature engineering and is therefore a convenient way of data representation. Using this natural data representation, convolution kernels calculate

[3] The video is available at the following URL: http://tinyurl.com/mte8wda

[4] From lecture notes of Avrim Blum: http://www.cs.cmu.edu/∼avrim/ML09/lect0126.pdf. Modified for our purposes.

the similarity between two objects by recursively dividing the objects into "parts", calculating the similarity between smaller parts, and aggregating these similarities to report a similarity between objects. For example, the way a string kernel will calculate the similarity between two strings (say "abc" and "aec") is by mapping each string into an *implicit* feature space and then calculating the similarity between the two strings by taking a dot product of the mappings (see Table 1). The feature space is called *implicit* because the kernel never explicitly writes out these features (or sub-structures). It calculates the similarity by using a dynamic program that recurses over these structures to find similar sub-structures.

	a	b	c	e	ab	ac	bc	ae	ec
"abc"	1	1	1	0	1	1	1	0	0
"aec"	1	0	1	1	0	1	0	1	1
$\vec{v}$	1	0	1	0	0	1	0	0	0

Table 1: An example showing how a string kernel will calculate the similarity between two strings. The implicit feature space is {a, b, c, e, ab, ac, bc, ae, ec }. $\vec{v}$ refers to the dot product of the vectors of the two strings. Similarity between these two strings is $\sum_{i=1}^{9} v_i = 3$

Thus, convolution kernels allow the learner to make similarity calculations without compromising the original structure of the objects (unlike feature engineering, where every object is represented as a vector in a finite dimensional space, thus losing the original structure of objects). This was the key observation that lead us to define *objects* as dance forms, and to our choice of using convolution kernels for explaining the machine learning process through a choreography. We discuss this in detail in the next section.

3 Artificial IntelliDance

In 2011, we created a choreography to present the idea of how a machine goes through the process of learning from data. We presented a perceptron, in its dual form, that uses convolution kernels to learn how to differentiate between two categories of *objects*. The 15 minute choreography is supported by a narrative, which is an interaction between a machine, depicted by a dancer, and a *user*, whose voice is heard but who remains unseen.

One of the main and early challenges we ran into during the ideation of the choreography had to do with the definition of *objects*. Though the central goal of the choreography was to explain a scientific idea, we wanted the choreography to maintain its aesthetic value. As a consequence of this constraint, we decided to stay away from defining objects as *things* that would restrict the dancers from moving freely in a natural way.

As discussed in the previous section, since convolution kernels allow for a natural representation of objects, we define our objects to be two dance forms: Ballet and Modern dance. Much like string kernels, where the implicit feature space is the space of sub-strings (that form a string), in our case, the high dimensional kernel space is the space of sub-movements (that form a movement). Each dancer is a data point, seen as a sequence of movements in an infinite dimensional space.

Figure 1: Above is a scene from one of the performances in which the machine, represented by the dancer in silver, "considers" the data. Prominently featured are data point dancers in red and yellow, both of whom have been marked with category-differentiating shapes (round for Ballet and diamond for Modern).

The choreography is broken into multiple phases. In the first phase, we motivate the need for machine learning, or pattern recognition, by presenting an apparently chaotic scene; all of the dancers are onstage at once, performing unique movement sequences, with only brief moments of synchronized action. The cacophonous dancing conveys the overwhelming difficulty for data scientists to find patterns in data using the naked eye. The dialogue advances the choreography to the next phase, where we sketch out the learning process.

In the learning phase, the machine starts by mak-

ing a prediction on the first data point. Since the machine has no prior knowledge ($\mathbf{w_1} = 0$), it makes a random prediction and gets the category wrong. The machine marks the dancer with a symbol in order to remember the data point ($\mathbf{w_t} \leftarrow \mathbf{w_t} + y_i\mathbf{x_i}$). The machine is then asked to make a prediction on a second data point. The machine compares this new data point with the data point it marked and makes a prediction ($\mathbf{w} \cdot \mathbf{x} = \sum_{k=1}^{N_s} y_k K(\mathbf{x_k}, \mathbf{x})$). Once again, it gets the category wrong and marks the second data point as well. This process continues until the machine has seen all the training instances and has selected data points it *thinks* encode structures important for classification.

Marking of dancers is done explicitly where the machine dancer attaches a round or triangular symbol to the data points: round is for Ballet and triangle is for Modern (see Figure 1). This is analogous to how a perceptron attaches positive and negative weights to the data points belonging to positive and negative categories respectively.

The narration points out a big limitation of convolution kernel methods, which is, in the worst case, every data-point is compared with every other data point in the training data, thus making the learning process slow (because the machine needs to go through the training data twice).

We also differentiate between the low dimensional feature space, in which the machine is unable to separate the data, and the high dimensional space, which offers distinguishability. The set of inactive training data points, i.e. the data points in a low dimensional feature space, is depicted by a clump of dancers in a corner who are hardly moving. The set of data points that are actively moving lie in a high dimensional feature space in which the machine learns a linear separator.

The next phase is testing, in which the machine compares the test dancers with the dancers it marked in the learning phase. After comparing each test point with all the support vectors, the machine makes a prediction. This phase concludes by showing that the machine has in fact learned to differentiate between the two categories.

The user is impressed and asks the machine to reveal the secret sauce. In this part of the choreography we visually describe how convolution kernels go about calculating similarities between two abstract objects, by breaking the object into parts, and recursing over the parts to calculate similarity. This action is illustrated by a comparison of similar movements and sub-movements as executed by a ballet and modern dancer. Situated side by side, the two dancers fragment the movements into increasingly smaller bits so as to make differences in the two forms of dance (objects) more visibly comparable. We also highlight the reason for the machine to look at a pair of data points instead of individual data points. The reason is that the machine does not remember the sub-structures important for classification (because the implicit feature space is enormous). By marking the data points, it only remembers the data points that encode these sub-structures. To this, the user voice points out another limitation of using convolution kernels; interpretability of models is hard. We can learn predictive models, but the fine grained structures important for classification remain hidden.

The piece ends with all the dancers linearly separated into categories in a high dimensional implicit feature space. Through the narration we point out the main differences and similarities between the two forms of dance, which are aesthetically visible but are sometimes hard to articulate.

4 Conclusion

In this paper, we presented a choreography that illustrates the process of supervised machine learning using a perceptron and convolution kernels. The choreography is structured around a scene in which the machine (represented by a dancer) learns to differentiate between two categories of objects, ballet and modern dance. The choreography not only explains the process of machine learning and how convolution kernels work, it also brings out two major limitations of using convolution kernels visually – having to go through the data twice, which makes the learning process slow, and that the interpretability of the models is hard, because the important substructures are not stored explicitly. While the general ideas about supervised machine learning may be more relevant to an audience that is not familiar with machine learning, the choreography may also be used to explain convolution kernels (in a visual and entertaining way) to researchers familiar with

machine learning but not necessarily familiar with how a perceptron uses a convolution kernel in its dual form.

Artificial IntelliDance premiered at Barnard College in April 2012, and has since been invited to perform at the World Science Festival 2012 and TEDx ColumbiaEngineering 2012. The audience was comprised of a combination of scientists and non-scientists, including dance artists, undergraduate and graduate students, and the general public. The primary concepts of the presentation were understood clearly by a number of viewers lacking familiarity with any machine learning paradigm as evidenced by the post-presentation discussions. Unfortunately, we do not have a more precise evaluation as to how many people actually understood the scientific ideas. The only "evaluation" we have is that the video continues to be showcased; we were recently invited to showcase it at Australia's National Science Week 2013. However, we have not yet heard of the video being used in a Machine Learning lecture.

In addition to functioning as an educational tool, a noteworthy outcome of the project is that it fosters dialogue between the general public and the arts and computer science communities.

Acknowledgments

We would like to thank Caronae Howell, Kapil Thadani, and anonymous reviewers for useful comments. We thank the dancers involved in the production and performance of the piece (in no particular order): Aditi Dhruv, Jenna Simon, Morgan Caglianone, Maddie James, Claire Salant, Anna Brown Massey, Emily Craver, Mindy Upin, Temple Kemezis, Taylor Gordon, Chelsea Cusack, Lane Halperin, Lisa Fitzgerald and Eleanor Barisser. We would like to thank Robert Boston for music and Marlon Cherry for voice of the user.

References

Apoorv Agarwal and Owen Rambow. 2010. Automatic detection and classification of social events. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1024–1034, Cambridge, MA, October. Association for Computational Linguistics.

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, Portland, Oregon, June. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 423–429, Barcelona, Spain, July.

Jason Eisner. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 10–18, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Christianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Special Issue on Semantic Role Labeling, Computational Linguistics Journal,*.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Conference on Association for Computational Linguistic*.

Frank Rosenblatt. 1958. The perceptron. *Psych. Rev*, 65(6):386–408.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLP-CoNLL*.