


Knowledge transfer in SVM and neural networks

Vladimir Vapnik^{1,2} · Rauf Izmailov³ 

Published online: 20 February 2017
© Springer International Publishing Switzerland 2017

Abstract The paper considers general machine learning models, where knowledge transfer is positioned as the main method to improve their convergence properties. Previous research was focused on mechanisms of knowledge transfer in the context of SVM framework; the paper shows that this mechanism is applicable to neural network framework as well. The paper describes several general approaches for knowledge transfer in both SVM and ANN frameworks and illustrates algorithmic implementations and performance of one of these approaches for several synthetic examples.

Keywords Intelligent teacher · Privileged information · Similarity control · Knowledge transfer · Knowledge representation · Frames · Support vector machine · Neural network · Classification · Learning theory · Regression

Mathematics Subject Classification (2010) 68Q32 · 68T05 · 68T30 · 83C32

This material is based upon work partially supported by AFRL and DARPA under contract FA8750-14-C-0008 and the work partially supported by AFRL under contract FA9550-15-1-0502. Any opinions, findings and / or conclusions in this material are those of the authors and do not necessarily reflect the views of AFRL and DARPA.

✉ Rauf Izmailov
rizmailov@vencorelabs.com
Vladimir Vapnik
vladimir.vapnik@gmail.com

¹ Columbia University, New York, NY, USA

² AI Research Lab, Facebook, New York, NY, USA

³ Vencore Labs, Basking Ridge, NJ, USA

1 Introduction

The classical machine learning paradigm considers the following simple scheme: given a set of training examples, find, in a given set of functions, the one that approximates the unknown decision rule in the best possible way. In such a paradigm, Teacher does not play an important role (it only supplies classification labels). However, in human learning, the role of Teacher is much more sophisticated: along with labels of examples, Teacher provides students with explanations, comments, comparisons, metaphors, and so on.

This paper considers the model of learning that includes the so-called Intelligent Teacher, who supplies Student with intelligent (privileged) information during training session. This privileged information exists for almost any learning problem and this information can significantly accelerate the learning process. In the learning paradigm called *Learning Using Privileged Information (LUPI)*, Intelligent Teacher provides additional (privileged) information x^* about training example x at the training stage (when Teacher interacts with Student). The important point in this paradigm is that privileged information is *not* available at the test stage (when Student operates without supervision of Teacher). LUPI was initially introduced in [15, 16]; subsequent work targeted various implementation issues of this paradigm [10] and its applications to a wide range of problems [2, 6, 11, 12, 19].

Formally, the classical paradigm of machine learning is described as follows: given a set of iid pairs (training data)

$$(x_1, y_1), \dots, (x_\ell, y_\ell), \quad x_i \in X, \quad y_i \in \{-1, +1\}, \quad (1)$$

generated according to a fixed but unknown probability measure $P(x, y) = P(y | x)P(x)$, find, in a given set of indicator functions $f(x, \alpha)$, $\alpha \in \Lambda$, the function $y = f(x, \alpha_*)$ that minimizes the probability of incorrect classifications (incorrect values of $y \in \{-1, +1\}$). In this model, each vector $x_i \in X$ is a description of an example generated according to an unknown generator $P(x)$ of random vectors x_i , and $y_i \in \{-1, +1\}$ is its classification defined by Teacher according to an unknown conditional probability $P(y | x)$. The goal is to find the function $y = f(x, \alpha_*)$ that guarantees the smallest probability of incorrect classifications. That is, the goal is to find the function which minimizes the risk functional

$$R(\alpha) = \frac{1}{2} \int |y - f(x, \alpha)| dP(x, y) \quad (2)$$

in the given set of indicator functions $f(x, \alpha)$, $\alpha \in \Lambda$ when the probability measure $P(x, y) = P(y | x)P(x)$ is unknown but training data (1) are given.

The LUPI paradigm describes a more complex model: given a set of iid triplets

$$(x_1, x_1^*, y_1), \dots, (x_\ell, x_\ell^*, y_\ell), \quad x_i \in X, \quad x_i^* \in X^*, \quad y_i \in \{-1, +1\}, \quad (3)$$

generated according to a fixed but unknown probability measure $P(x, x^*, y) = P(x^*, y | x)P(x)$, find, in a given set of indicator functions $f(x, \alpha)$, $\alpha \in \Lambda$, the function $y = f(x, \alpha_*)$ that guarantees the smallest probability of incorrect classifications (2). In this model, each vector $x_i \in X$ is a description of an example generated according to an unknown generator $P(x)$ of random vectors x_i , and Intelligent Teacher generates both its label $y_i \in \{-1, +1\}$ and the privileged information x_i^* using some unknown conditional probability function $P(x_i^*, y_i | x_i)$.

In the LUPI paradigm, we have exactly the same goal of minimizing (2) as in the classical paradigm, i.e., to find the best classification function in the admissible set. However, during

the training stage, we have more information, i.e., we have triplets (x, x^*, y) instead of pairs (x, y) as in the classical paradigm. The additional information $x^* \in X^*$ belongs to space X^* which is, generally speaking, different from X .

The paper is organized in the following way. In Section 2, we outline general models of information theory and their relation to models of learning. In Section 3, we explain how privileged information can significantly accelerate the rate of learning (i.e., accelerate the convergence) when the notion of classical learning model is expanded appropriately to incorporate privileged information. In Section 4, we argue that structures in the space of privileged information reflect more fundamental properties of learning and thus can potentially improve the performance of learning methods even further; we also outline a general *knowledge transfer* approach for realization of that improvement. In Section 5, we present specific algorithms implementing that framework for SVM and neural networks [4, 8] and illustrate their properties and performance on synthetic examples. We conclude with Section 6, in which we summarize our results and outline potential next steps in this research.

2 Brute force and intelligent models

In this section, we show how the general setting of machine learning problems justifies the introduction of the concept of privileged information.

According to [7, p. 53], there exist three categories of integer numbers.

1. **Ordinary numbers:** those numbers n that we use in our everyday life. For simplicity, let these numbers be between 1 and one million.
2. **Large numbers:** those numbers N that are between one million and 2^n (where n belongs to the category of ordinary numbers).
3. **Huge numbers:** those numbers H that are greater than $2^N = 2^{2^n}$ (where N belongs to the category of large numbers).

Kolmogorov argued that the ordinary integers n correspond to the number of items we can handle realistically, say the number of examples in a learning problem. We cannot realistically handle large numbers (say large number of examples in a learning problem), but we can still treat them efficiently in our theoretical reasoning using mathematics (say, select one element from a sample using mathematical methods); however, huge numbers are beyond our reach. In this paper, we describe methods that potentially might operate in huge sets of functions. In contrast to methods based on mathematical models and suitable for large numbers (which we call “brute force” methods), these methods include intelligent agents and thus can be viewed as “intelligent methods”.

Basic Shannon model Suppose that our goal is to find one function among large number N of different functions by making ordinary number of queries that return the reply “yes” or “no” (thus providing one bit of information). Theoretically, we can find the desired function among N functions by making n queries, where $n = \log_2 N$ (for simplicity, we assume that N is an integer power of 2). Indeed, we can split the set of N functions into two subsets and make query to which subset the desired function belongs: to the first one (reply +1) or to the second one (reply -1). After obtaining the reply from the query, we can remove the subset which does not contain the desired function, split the remaining part into two subsets, and continue in the same fashion, removing half of the remaining functions after each reply. So after $n = \log_2 N$ queries we will find the desired function. It is easy to see

that one cannot guarantee that it is possible to find the desired function by making less than

$$n = \log_2 N = \frac{\ln N}{\ln 2} \quad (4)$$

queries. This also means that one cannot find one function from the set of huge number $H = 2^N$ of functions: this would require to make too many (namely N) queries, which is unrealistic.

Basic model using language of learning theory Let us repeat this reasoning for pattern recognition model. Suppose that our set $y = f(x, \alpha_t)$, $t = 1, \dots, N$ is a finite set of binary functions in $x \in R^n$. That is, $f(x, \alpha_t) \in \{-1, +1\}$. Suppose that we can construct such vector $x_1 \in R^n$ that half of the functions take value $f(x_1, \alpha_{t_1}^*) = +1$ and another half take value $f(x_1, \alpha_{t_1}) = -1$. Then the query for the label of vector x_1 provides the first element of training data (x_1, y_1) . As before, we remove half of the functions for which the query reply was $-y_1$ and continue this process. After collecting at most $n = \ln N / \ln 2$ elements of training examples, we obtain the desired function.

2.1 First modification of the learning model

To find the function in the framework of basic model requires solving a difficult problem: on any step of the procedure, to find a vector x_i that splits the remaining set of functions into two equal parts (suppose that such a vector exists). To simplify our model, consider the situation where vectors x are results of random iid trial with a fixed (but unknown) probability measure $p(x)$, and for any x we can query for its label y . After each query, we remove those functions for which the query reply was $-y$ on x (they could comprise less than half of the current set of functions). The main problem for this model is to determine how many queries about labels one has to make¹ to find the function that is ε -close to the desired one with probability $1 - \eta$ (recall that the desired function is any function among those that do not make errors, and ε -closeness is defined with respect to measure $p(x)$, as in (2)). The answer to this problem constitutes a special case of the VC theory [13, 14]: the number of the required queries is at most

$$\ell = \frac{\ln N - \ln \eta}{\varepsilon}. \quad (5)$$

This expression differs from bound (4) by a constant: $(\varepsilon)^{-1}$ instead of $(\ln 2)^{-1}$. After this number of queries, any function in the remaining set is ε -close to the desired one. This bound cannot be improved.

2.2 Second modification of the basic model

So far, we considered the situation when the set of N functions includes the one that does not makes errors. Now we relax this assumption: any function in our set of N functions can make errors. Our problem is to find the function than provides the smallest probability of error with respect to probability measure $p(x)$.

Now we cannot use the method for choosing the desired function defined in the first model: removing from the consideration the functions from the set that disagree with classification of query. We will use another (a more general) algorithm which selects such

¹In other words, how large should be the number ℓ of training examples $(x_1, y_1), \dots, (x_\ell, y_\ell)$.

function among N of them that make the smallest number of disagreements with the query reply (i.e., minimizes the empirical loss) on the training set

$$(x_1, y_1), \dots, (x_\ell, y_\ell).$$

In order to guarantee that we will select an ε -close function to the best in the set of N elements with probability $1 - \eta$, one has to make at most

$$\ell = \frac{\ln N - \ln \eta}{\varepsilon^2}$$

queries. Again, in this modification, the main term $\ln N$ remains the same but constant $(\varepsilon)^{-2}$ is different from the constant in (5). This bound cannot be improved.

2.3 Third modification (VC model)

Consider now the set of functions $f(x, \alpha)$, $\alpha \in \Lambda$ with infinite number of elements. Generally speaking, in this situation one cannot guarantee that it is possible to obtain a good approximation even if we have a large number of training examples. Recall that in the more simple situation with a set that contains finite but huge number of functions $H = 2^{2^n}$, one needs 2^n examples, which is far beyond our reach. Nevertheless, if our infinite set of functions has finite VC dimension $VCdim$, then ε -close solution can be found with probability $1 - \eta$ using at most ℓ queries, where ℓ satisfies the equation

$$\ell = \frac{(VCdim) \ln \ell - \ln \eta}{\varepsilon}$$

if the desired function does not make errors; otherwise, if errors are allowed, ℓ satisfies the equation

$$\ell = \frac{(VCdim) \ln \ell - \ln \eta}{\varepsilon^2}.$$

Note that this bound matches the form of bound (5), where the value of VC dimension multiplied by $\ln \ell$ replaces the logarithm of the number of functions in the set. This bound cannot be improved. Note that these formulas correspond to those in Chapter 3 of [13] for sufficiently large ℓ .

The finiteness of the VC dimension of the set of functions defines the necessary and sufficient conditions of learnability (consistency) of empirical risk minimization method. This means that VC dimension characterizes not just the *quantity* of elements of the set: it also characterizes something else, namely, the *measure of diversity* of the set of functions: the set of functions must not be too diverse.

The structural risk minimization principle that uses structure on the nested subsets of functions with finite VC dimension (defined on the sets of functions the closure of which can have infinite VC dimension) guarantees convergence of risk to the best possible risk for this structure [13, 14].

To summarize, we have outlined the best bounds for general machine learning models and stated that they cannot be improved. To put it differently, in order to improve these bounds, the models themselves will have to be changed. The specific model change that we are concerned with in this paper is provided by the notion of privileged information, which is described and explored in the subsequent sections.

3 Privileged information as learning acceleration

The learning models described in the previous section can be solved by different methods. In particular, SVM algorithms with universal kernels (i.e., capable of approximating any continuous function with arbitrary precision) realize structural risk minimization method and thus are universally consistent. This means that the VC theory completely solves the problem of learning from examples providing not only the necessary and sufficient conditions of learnability but also an effective practical algorithm for machine learning. The rate described by this theory cannot be improved essentially (without additional information).

The intriguing question in VC theory was why the number of examples one needs to construct ε -close hyperplane in separable case (when training data can be separated without errors) and unseparable case (when training data cannot be separated without errors) vary so much in their corresponding constants (ε^{-1} and ε^{-2}).

This effect can be explained and leveraged within the LUPI framework [15, 16]. In that framework, Intelligent Teacher supplies Student with triplets

$$(x_1, x^*, y_1), \dots, (x_\ell, x_\ell^*, y_\ell)$$

where $x_i \in X^*$, whereas, in the classical setting of the problem, Student uses training pairs

$$(x_1, y_1), \dots, (x_\ell, y_\ell)$$

where vector $x_i \in X$ is generated by the generator of random events $p(x)$ and Teacher supplies Student with the label $y_i \in \{-1, +1\}$. In contrast to classical setting, in the LUPI paradigm, Intelligent Teacher supplies Student with triplets (x_i, x_i^*, y_i) where vector $x_i^* \in X^*$ and label y_i are generated by conditional probability $p(x^*, y | x)$. Formally, by providing both vector $x^* \in X^*$ and label y_i for any example x_i , Intelligent Teacher can supply Student with *more than one bit of information*, so the rate of convergence can be greater.

Indeed, as was shown in [15, 16], an SVM-based approach in LUPI can improve the constant from ε^{-2} to ε^{-1} . The recent papers [17, 18] introduced more important approaches in LUPI that could be potentially used for further improvement of convergence. Although those approaches were set within the SVM context, they are quite general, and we will show further that they are applicable for neural networks as well.

In order to use such mechanisms effectively, Intelligent Teacher has to possess some knowledge that can describe physical model of events better than x . In the subsequent sections, we describe these ideas in greater detail.

4 Privileged information and knowledge transfer

Let us suppose that Intelligent Teacher has some knowledge about the solution of a specific pattern recognition problem and would like to transfer this knowledge to Student. For example, Teacher can reliably recognize cancer in biopsy images (in a pixel space X) and would like to transfer this skill to Student.

Formally, this means that Teacher has some function $y = f_0(x)$ that distinguishes cancer ($f_0(x) = +1$ for cancer and $f_0(x) = -1$ for non-cancer) in the pixel space X . Unfortunately, Teacher does not know this function explicitly (it only exists as a neural net in Teacher's brain), so how can Teacher transfer this construction to Student? Below, we describe a possible mechanism for solving this problem; we call this mechanism *knowledge transfer*.

Suppose that Teacher believes in some theoretical model on which the knowledge of Teacher is based. For cancer model, he or she believes that it is a result of uncontrolled multiplication of the cancer cells (cells of type \mathcal{B}) which replace normal cells (cells of type \mathcal{A}). Looking at a biopsy image, Teacher tries to generate privileged information that reflects his or her belief in development of such a process; Teacher can describe the image as:

Aggressive proliferation of cells of type \mathcal{B} into cells of type \mathcal{A} .

If there are no signs of cancer activity, Teacher may use the description

Absence of any dynamics in the standard picture.

In uncertain cases, Teacher may write

There exist small clusters of abnormal cells of unclear origin.

In other words, Teacher uses a specialized language that is appropriate for description x_i^* of cancer development employing the model he believes in. Using this language, Teacher supplies Student with privileged information x_i^* for the image x_i by generating training triplets

$$(x_1, x_1^*, y_1), \dots, (x_\ell, x_\ell^*, y_\ell). \tag{6}$$

The first two elements of these triplets are descriptions of an image in two languages: in language X (vectors x_i in pixel space), and in language X^* (vectors x_i^* in the space of privileged information), developed for Teacher’s understanding of cancer model.

Note that the language of pixel space is universal (it can be used for description of many different visual objects; for example, in the pixel space, one can distinguish between male and female faces), while the language used for describing privileged information is very specialized: it reflects just a model of cancer development. This has an important consequence: the set of admissible functions in the general space X has to be rich (has large VC dimension), while the set of admissible functions in the specialized space X^* may be not rich (has small VC dimension).

One can consider two related pattern recognition problems using triplets (6):

1. The problem of constructing a rule $y = f(x)$ for classification of biopsy in the pixel space X using data

$$(x_1, y_1), \dots, (x_\ell, y_\ell). \tag{7}$$

2. The problem of constructing a rule $y = f^*(x^*)$ for classification of biopsy in the space X^* using data

$$(x_1^*, y_1), \dots, (x_\ell^*, y_\ell). \tag{8}$$

Suppose that language X^* is so good that it allows to create a rule $y = f_\ell^*(x^*)$ that classifies vectors x^* corresponding to vectors x with higher accuracy.

Since the VC dimension of the admissible rules in the specialized space X^* is much smaller than the VC dimension of the admissible rules in the universal space X and since the number of examples ℓ is the same in both cases, the bounds on error rate for the rule $y = f_\ell^*(x^*)$ in X^* will be better² than those for the rule $y = f_\ell(x)$ in X . That is, generally speaking, the classification rule $y = f_\ell^*(x^*)$ will be more accurate than classification rule $y = f_\ell(x)$.

²According to VC theory, the guaranteed bound on accuracy of the chosen rule depends only on two factors: frequency of errors on the training set and VC dimension of the admissible set of functions.

As a result, the following question of “knowledge transfer” arises: how one can use the knowledge of the rule $y = f_\ell^*(x^*)$ in space X^* to improve the accuracy of the desired rule $y = f_\ell(x)$ in space X ? We now address this question when both problems (7)–(8) are solved with neural networks.

Consider three elements of knowledge representation used in Artificial Intelligence [1]:

1. Fundamental elements of knowledge.
2. Frames (fragments) of the knowledge.
3. Structural connections of the frames (fragments) in the knowledge.

We call the *fundamental elements of the knowledge* a limited number of elements (functions) in X^* that can approximate well the classification rule $y = f_\ell^*(x^*)$; then knowledge transfer is about approximation of those fundamental elements. We now illustrate this concept for SVMs and neural networks.

4.1 Knowledge transfer for SVM

In order to describe methods of knowledge transfer for SVM, consider the following three-level structure:

1. Level I^X : the input vectors $x = (x^1, \dots, x^n) \in X$.
2. Level I^Z : the result of transformation of the vectors x into vectors $z = (K(x_1, x), \dots, K(x_\ell, x)) \in Z$, where K is the kernel function [14] for SVM.
3. Level I^Y : the linear threshold indicator function ³ $y = \theta(a^T z(x) - b)$ in space Z .

Thus the structures of SVM rules in spaces X and X^* can be described as

$$X : (I^X \longrightarrow I^Z \longrightarrow I^Y) \quad \text{and} \quad X^* : (I^{X^*} \longrightarrow I^{Z^*} \longrightarrow I^{Y^*}).$$

To transfer the knowledge about the rule

$$y = f(x^*, \alpha_\ell^*) - b^* = \sum_{i=1}^{\ell} \alpha_i^* K^*(x_i^*, x^*) - b^*$$

in space X^* to the rule

$$y = f(x, \alpha_\ell) - b = \sum_{i=1}^{\ell} \alpha_i K(x_i, x) - b$$

obtained in space X , one can use several strategies. Below we consider three of them.

1. **A-mapping of privileged information:** $X^* \longrightarrow X$. In this scheme, the goal is to transfer information that exists in level I^{X^*} of SVM in space X^* to level I^X of SVM in space X . In order to do this, one maps vectors $x \in X$ into vectors $x^* \in X^*$ by transforming space X obtaining vectors $\bar{x} = Ax$ and then constructs SVM in the transform space \bar{X} . Scheme (A) of information transfer can be thus described as

$$(A): (I^{X^*} \longrightarrow I^X) \longrightarrow I^Z \longrightarrow I^Y$$

³Neural Network with one hidden layer has the same structure; as SVM, it is a *universal* learning machine.

In this scheme, in order to find the transformation Ax of vectors $x = (x^1, \dots, x^n)^T \in X$ into vectors $Ax = (\phi_1(x), \dots, \phi_m(x))^T$ that minimizes the functional

$$R(A) = \min_A \int |x^* - Ax|^2 p(x^*, x) dx^* dx,$$

we look for the minimum

$$R(\phi) = \sum_{k=1}^n \min_{\phi_k} \int (x^{*k} - \phi_k(x))^2 p(x^{*k}, x) dx^{*k} dx,$$

where $p(x^{*k} | x)$ is the marginal conditional probability of coordinate x^{*k} given vector x , and m functions $\phi_k(x)$ are defined by m regressions

$$\phi_k(x) = \int x^{*k} p(x^{*k} | x) dx^{*k}, \quad k = 1, \dots, m.$$

We construct approximations to functions $\phi_k(x)$, $k = 1, \dots, m$ by solving m regression estimation problems based on data

$$(x_1^{*k}, x_1), \dots, (x_\ell^{*k}, x_\ell), \quad k = 1, \dots, m.$$

In order to find these approximations, we Structural Risk Minimization principle [14] in the set of functions that belong to the Reproducing Kernel Hilbert Space (RKHS) associated with some kernel, that is, by minimizing the regularized functional

$$R(\phi_k) = \min_{\phi_k} \sum_{i=1}^{\ell} (x_i^{*k} - \phi_k(x_i))^2 + \gamma \langle \phi_k(x), \phi_k(x) \rangle, \quad k = 1, \dots, m.$$

The obtained approximations to the regressions $\phi_1(x), \dots, \phi_m(x)$ define our transformation. In this scheme, we first transform the input space $\bar{X} = AX$ and then train SVM in the transformed space.

- 2. **B-mapping of privileged information:** $Z^* \rightarrow X$. In this scheme, the goal is to transfer information that exists in level I^{Z^*} of SVM in space X^* to level I^X of SVM in space X . In order do this, one maps vectors $x \in X$ to vectors $z^* \in Z^*$ by transforming space X and obtaining vectors $\bar{x} = Bx \in \bar{X}$ and then constructs SVM in the transformed input space. Scheme (B) of information transfer can be thus described as

$$(B): (I^{X^*} \rightarrow I^{Z^*}), \rightarrow I^X \rightarrow I^Z \rightarrow I^Y \tag{9}$$

or, in its simplified form, as

$$(B^*): (I^{X^*} \rightarrow I^{Z^*}) \rightarrow I^X \rightarrow I^Y. \tag{10}$$

The transformation of input space in this scheme is based on solving the following t regression estimation problems (t is the dimension of vector $z^* = (K(x_1^*, x^*), \dots, K(x_t^*, x^*))^T$, i.e., the number of support vectors in SVM solution for space X^*): given data

$$(K(x_k^*, x_1^*), x_1), \dots, (K(x_k^*, x_\ell^*), x_\ell), \quad k = 1, \dots, t,$$

find the regression functions

$$\phi_k(x) = \int K(x_k^*, x^*) p(x^* | x) dx^*, \quad k = 1, \dots, t.$$

As already described above for A-mapping, one can find such approximation in the RKHS associated with some kernel function. The obtained approximations $\phi_1(x)$,

..., $\phi_m(x)$ define our transformation: in general scheme (9), we construct SVM rule in the transformed space; in simplified scheme (10), we construct linear SVM rule in the transformed space \bar{X} .

3. **C-mapping of privileged information:** $I^{Z^*} \rightarrow I^Z$. In this scheme, the goal is to transfer information that exists in the level I^{Z^*} of SVM in space X^* to the level I^Z of SVM in space X . In order to do this, one maps t -dimensional vectors $z \in Z$ (t is the number of support vectors of the SVM rule obtained in space X) into t^* -dimensional vectors $z^* \in Z^*$ (t^* is the number of support vectors of the SVM rule obtained in space X^*) constructing vectors of the form $\bar{z} = Cz \in \bar{Z}$. Every coordinate k in Z space defines similarity $K(x_k, x)$ between support vector x_k and vector $x \in X$, while every coordinate k^* in Z^* space defines similarity $K^*(x_k^*, x^*)$ between support vector x_k^* and vector $x^* \in X^*$, where x and x^* are connected through $p(x^* | x)$. Scheme (C) of information transfer can be described as

$$(C): (I^{X^*} \rightarrow I^{Z^*}) \rightarrow (I^X \rightarrow I^Z) \rightarrow I^Y.$$

Our goal is to approximate the similarity function $K^*(x_k^*, x^*)$, $k = 1, \dots, t^*$ between support vector x_k^* of SVM solution in space X^* and vector $x^* \in X^*$ using t similarity functions $K(x_1, x), \dots, K(x_t, x)$ defined by SVM solution in space X for the pairs (x, x^*) generated by $p(x^* | x)$.

Let x_1, \dots, x_t be the support vectors of SVM solution in space X and let $x_1^*, \dots, x_{t^*}^*$ be the support vectors of SVM solution in space X^* , where t and t^* are the numbers of support vectors in SVM solutions obtained in spaces X and X^* , respectively. The SVM rule of the space X has the form

$$f(x, \alpha) = \sum_{i=1}^t \alpha_i K(x_i, x) + b,$$

and SVM rule in space X^* has the form

$$f^*(x^*, \alpha^*) = \sum_{i=1}^{t^*} \alpha_i^* K^*(x_i^*, x^*) + b^*.$$

In order to achieve our goal, we approximate the functions $K^*(x_k^*, x^*)$, $k = 1, \dots, t^*$ with the regression functions

$$\phi_k(x) = \int K(x_k^*, x^*) p(x^* | x) dx^*, \quad k = 1, \dots, t^*.$$

For each $k = 1, \dots, t^*$, we construct the approximation to $\phi_k(x)$ by using the data

$$(K^*(x_k^*, x_1^*), z_1), \dots, (K^*(x_k^*, x_{\ell}^*), z_{\ell}), \quad k = 1, \dots, t^*,$$

where $z_i = (K(x_1, x_i), \dots, K(x_t, x_i)) \in Z$ is t -dimensional vector. Let space $\bar{Z} = (\phi_1(x), \dots, \phi_t(x))$ be the result of transformation of space Z . After that, we construct linear SVM in space \bar{Z} .

One can construct many different schemes of knowledge transformation from space X^* to space X (as well as schemes of combining knowledge existing in both spaces) based on described approaches.

In particular, in all three described mappings A–C, one may also concatenate the constructed knowledge transferred features with those already available from space X and solve SVM on this augmented set; constructed knowledge transferred features could be subject to feature selection in order to improve the classification performance; for C-mapping, one

could construct regression functions only to those functions $K^*(x_i^*, x^*)$ that correspond to “significant” weights α_i ; if linear regression functions are used for C-mapping, their positive versions could be explored as more relevant, etc. Note that C-mapping requires executing two versions of SVM: one for standard space, and one for privileged one.

4.2 Knowledge transfer in neural networks

Knowledge transfer in Neural Networks is analogous to the one used for knowledge transfer in SVMs. As in the case of SVM described above, one constructs and trains two neural networks: one network in space X and another network in space X^* . To simplify the notations, we assume that both networks have the same architecture containing s layers. Let input vector $x \in X$ define the first layer $I^X(0)$ of neural network in space X ; this vector is transferred into vector $z^1 \in Z(1)$ in the next layer of the trained network, and layers $I^Z(k), k = 2, \dots, s - 1$ provide subsequent transformations $z^k \in Z(k)$. As in SVM, the last layer is the linear indicator function $y = \theta((a^s, z^s) - b)$ (or its sigmoid approximation). The structure of Neural Network in space X is

$$X : I^Z(0) \longrightarrow I^Z(1) \longrightarrow \dots \longrightarrow I^Y. \tag{11}$$

and the structure of Neural Network in space X^* is

$$X^* : I^{Z^*}(0) \longrightarrow I^{Z^*}(1) \longrightarrow \dots \longrightarrow I^Y. \tag{12}$$

The *simple scheme* of knowledge transfer from network (12) in space X^* to network (11) in space X can be described as follows: information accumulated into first k layers of network (12) trained in space X^* is transferred into m -th layer of network (11) in space X :

$$\left(I^{Z^*}(0) \rightarrow \dots \rightarrow I^{Z^*}(k) \right) \rightarrow \left(I^Z(0) \rightarrow \dots \rightarrow I^Z(m) \right) \rightarrow I(m + 1) \rightarrow \dots \rightarrow I^Y,$$

which forms the operator $\overline{z}(k) = \mathcal{A}z(m)$ that transforms vectors $z(m)$ from neural network in space X into vectors $z^*(k)$ of neural network in space X^* .

The new neural network contains three parts:

1. The first part of the network contains first m layers of trained network in space X ; we denote it $\mathcal{N}(0, m)$. This network performs transformation $z(m) = \mathcal{N}(0, m)x(0)$.
2. The second part of the network contains operator \mathcal{A} that transforms vectors $z(m)$ in vectors $\overline{z}(k) = \mathcal{A}z(m)$.
3. The third part of the networks is the part of the network in space X^* starting from level $(k + 1)$, free parameters of which have to be learned; we denote it $\mathcal{N}^*(k, s)$. Vectors $\overline{z}(k)$ are the input of this part of network, and classifiers are the output.

The scheme of such combined networks is

$$\mathcal{A}\{\mathcal{N}(0, m)\} \longrightarrow \mathcal{N}^*(k, s),$$

where $\mathcal{N}(0, m)$ is fixed (does not have free parameters), while $\mathcal{N}^*(k, s)$ contains free parameters. Therefore operator \mathcal{A} transforms knowledge about neural network in X^* .

In order to find this operator based on two trained networks, one uses the same techniques of regression estimation as in the case of SVM. Let $z^*(k) = (z^{*1}(k), \dots, z^{*s}(k))$ be vectors produced on the level $I^{Z^*}(k)$ by the network trained in space X^* , and let $z(m) = (z^1(m), \dots, z^s(m))$ be vectors produced on the level $I^Z(k)$ by the network trained in space X .

Consider pairs

$$(x_1, x_1^*), \dots, (x_\ell, x_\ell^*)$$

from the training triplets (3). Let

$$z_1(m), \dots, z_\ell(m)$$

be vectors produced by m -th layer of neural networks (11) corresponding to vectors x and let

$$z_1^*(k), \dots, z_\ell^*(k)$$

be vectors

$$z_i^*(k) = (z_i^{*1}(k), \dots, z_i^{*s^*}(k))$$

produced by k -th layer of neural networks (12) corresponding to vectors x^* . In order to construct mapping operator \mathcal{A} as in SVM case, we estimate s^* regression functions $x^{*t}(k) = \phi_t(x(m))$ using data

$$(x_1^{*t}, x_1(m)), \dots, (x_\ell^{*t}, x_\ell(m)), \quad t = 1, \dots, s^*.$$

Therefore operator \mathcal{A} transforms vectors $x(m)$ into vectors

$$\mathcal{A}x(m) = (\phi_1(x(m)), \dots, \phi_{s^*}(x(m))).$$

For neural network that contains more than one hidden layer, one can transfer knowledge from network in X^* using more than one operator \mathcal{A}_j , $j = 1, \dots, p$ by sequentially constructing several transformations between different layers of network in X^* .

5 Knowledge transfer with A-mapping and C-mapping of privileged information

In the previous section, we described three key approaches for mapping of privileged information for knowledge transfer. In this section, we present scalable algorithms for two of them, namely A-mapping and C-mapping, based on multivariate regressions of privileged features as functions of decision variables; we also illustrate the algorithms' performance and their properties on several examples.

We start with A-mapping and assume again that we are given a set of iid triplets

$$(x_1, x_1^*, y_1), \dots, (x_\ell, x_\ell^*, y_\ell), \quad x_i \in X = R^n, x_i^* \in X^* = R^m, y_i \in \{-1, +1\}, \quad (13)$$

generated according to a fixed but unknown probability measure $P(x, x^*, y)$. Our training dataset consists of ℓ decision vectors x_1, \dots, x_ℓ from n -dimensional decision space $X = R^n$ and corresponding ℓ privileged vectors x_1^*, \dots, x_ℓ^* from m -dimensional privileged space $X^* = R^m$.

Specifically, for each $j = 1, 2, \dots, m$, do the following: using n -dimensional vectors x_1, x_2, \dots, x_ℓ as explanatory variables and corresponding scalar values $(x_1^*)^j, (x_2^*)^j, \dots, (x_\ell^*)^j$ as response variables, construct a (linear or nonlinear) regression function φ_j so that

$$\varphi_j(x_1^1, x_1^2, \dots, x_1^n) = z_1^j \approx (x_1^*)^j$$

$$\varphi_j(x_2^1, x_2^2, \dots, x_2^n) = z_2^j \approx (x_2^*)^j$$

.....

$$\varphi_j(x_\ell^1, x_\ell^2, \dots, x_\ell^n) = z_\ell^j \approx (x_\ell^*)^j$$

for pairs $(x_1, x_1^*), \dots, (x_\ell, x_\ell^*)$ from (13).

Various types of regression could be used for that purpose; in this paper, we use two of them: (1) linear ridge regression, and (2) nonlinear kernel regression, which is constructed as an approximation of the regressed values with linear combination of radial basis functions (where parameters are selected using 2-fold cross-validation).

In the next step, we create, following the previously described framework of knowledge transfer, the modified training dataset, consisting of the set with m -dimensional regression-based replacements of privileged vectors. As a result, our modified training data will form the matrix

$$\begin{pmatrix} \varphi_1(x_1^1, \dots, x_1^n) \cdots \varphi_m(x_1^1, \dots, x_1^n) & y_1 \\ \varphi_1(x_2^1, \dots, x_2^n) \cdots \varphi_m(x_2^1, \dots, x_2^n) & y_2 \\ \dots & \dots \\ \varphi_1(x_\ell^1, \dots, x_\ell^n) \cdots \varphi_m(x_\ell^1, \dots, x_\ell^n) & y_\ell \end{pmatrix}.$$

However, in real applications, it is also possible not to discard the decision training data from X , but, instead, concatenate m -dimensional regression-based replacements of privileged vectors with the decision data. As a result, our modified training data will form the matrix

$$\begin{pmatrix} x_1^1 \cdots x_1^n & \varphi_1(x_1^1, \dots, x_1^n) \cdots \varphi_m(x_1^1, \dots, x_1^n) & y_1 \\ x_2^1 \cdots x_2^n & \varphi_1(x_2^1, \dots, x_2^n) \cdots \varphi_m(x_2^1, \dots, x_2^n) & y_2 \\ \dots & \dots & \dots \\ x_\ell^1 \cdots x_\ell^n & \varphi_1(x_\ell^1, \dots, x_\ell^n) \cdots \varphi_m(x_\ell^1, \dots, x_\ell^n) & y_\ell \end{pmatrix}.$$

After that, we train an SVM or a neural network on the modified set of set of $(n + m)$ -dimensional vectors and construct the corresponding classification decision function F , which, when applied to any $(n + m)$ -dimensional vector Z , produces the classification output $Y = F(Z)$, where $Y \in \{-1, +1\}$.

The designed classification decision algorithm F can now be applied to any standard vector x from n -dimensional space R^n in following manner. First, we construct m scalar values

$$z^1 = \varphi_1(x), z^2 = \varphi_2(x), \dots, z^m = \varphi_m(x).$$

using already constructed (during training) m regressions $\varphi_1, \varphi_2, \dots, \varphi_m$. Then, we construct $(n + m)$ -dimensional vector Z by concatenating these m scalar values with n -dimensional vector X :

$$Z = (x^1 \ x^2 \ \dots \ x^n \ z^1 \ z^2 \ \dots \ z^m)$$

Finally, we apply the classification decision algorithm F (either SVM or ANN) to the constructed $(n + m)$ -dimensional vector Z and obtain the classification label $Y = F(Z)$, where $Y \in \{-1, +1\}$; this label Y is the desired classification of standard n -dimensional vector x .

In order to illustrate this version of knowledge transfer LUPU, we explored the synthetic dataset derived from dataset ‘‘Parkinsons’’ in [9]. Since none of 22 features of ‘‘Parkinsons’’ dataset is privileged, we created several artificial scenarios emulating the presence of privileged information in that dataset. Specifically, we ordered ‘‘Parkinsons’’ features according to the values of their mutual information (with first features having the lowest mutual information, while the last features having the largest one). Then, for several values of parameter k , we treated the last k features as privileged ones, while first $22 - k$ features being treated as decision ones. Since our ordering was based on mutual information, these experiments

corresponded to privileged spaces of various dimensions and various relevance levels for classification. For each considered value of k , we generated 20 pairs of training and test subsets, containing, respectively 75 % and 25 % of elements of the “Parkinsons” dataset. For each of these pairs, we considered the following four types of classification scenarios for both SVM (with RBF kernel) and ANN algorithms:

1. SVM and ANN on $22 - k$ decision features;
2. Knowledge transfer LUPI (linear) based on constructing k multiple linear regressions from $22 - k$ decision features to each of k privileged ones, replacing the corresponding values in privileged vectors with their regressed approximations, and then training SVM and ANN on the augmented dataset consisting of 22 features;
3. Knowledge transfer LUPI (non-linear) based on constructing k non-linear (in the class of RBF functions) regressions from $22 - k$ decision features to each of k privileged ones, replacing the corresponding values in privileged vectors with their regressed approximations, and then training SVM and ANN on the augmented dataset consisting of 22 features;
4. SVM and ANN on all 22 features.

For each scenario, the algorithms were trained in the following way:

SVM. Two parameters for RBF kernels, namely SVM penalty parameter C and RBF kernel parameter γ , were selected using 6-fold cross-validation error rate over the two-dimensional grid of both parameters C and γ . In that grid, $\log_2(C)$ ranged of from -5 to $+5$ with step 0.5, and $\log_2(\gamma)$ ranged $+6$ to -6 with step 0.5 (thus the whole grid consisted of $21 \times 25 = 525$ pairs of tested parameters C and γ).

ANN. Neural networks were trained using Mathworks™ Matlab Neural Network Toolbox™ with the same default parameters [3] such as using hyperbolic tangent sigmoid as activation function, applying Levenberg-Marquardt backpropagation training algorithm and selecting the ratio for training:validation:test as 70:15:15 for early stopping on cross-entropy, etc. For each N -dimensional input, the architecture of ANN was selected [5] with several hidden layers (from one to five) with the number of neurons in it ranging from 5 to 100 (a separate ANN was trained for each of these architecture choices final architecture was then selected based on the best performance). Note that we do not claim that these particular architecture choices for SVM and ANN are optimal; our point is to demonstrate the significant potential of LUPI improvement with different classification methods, whether these methods are optimal or not.

The averaged (over 20 realizations) error rates for these scenarios are shown in Table 1 (for SVM) and in Table 2 (for ANN). The collected results show that performance of SVM (and its LUPI modifications) is better than that of ANN (and its LUPI modifications). They also show that both linear and nonlinear versions of Knowledge Transfer LUPI improve the performance of SVM and ANN on decision inputs (often significantly, in relative terms) in all of the considered scenarios. Note that both versions are just examples of knowledge transfer and other mappings (especially if relevant domain knowledge is available) could be leveraged.

Numerically, the error rates of LUPI are between the corresponding SVM or ANN constructed on decision features and on all features. In other words, if the error rate of the algorithm on decision features is B , while the error rate of the algorithm on all features is C , the error rate A of LUPI satisfies the bounds $C < A < B$. So one can evaluate the efficiency of LUPI approach by computing the metric $(B - A)/(B - C)$, which describes how much of the performance gap $B - C$ can be recovered by LUPI. For SVM, this metric varies between 12 % and 78 %; for ANN, this metric varies between 16 % and 67 %. Generally,

Table 1 Performance of SVM and LUPI on modified “Parkinsons” example (A-mapping)

k	SVM on decision features	LUPI (linear)	LUPI (nonlinear)	SVM on all features	LUPI gain (linear)	LUPI gain (nonlinear)
12	13.26 %	9.18 %	11.32 %	6.63 %	61.55 %	29.25 %
11	13.52 %	10.66 %	12.70 %	6.63 %	41.49 %	11.85 %
10	13.16 %	10.00 %	12.19 %	6.63 %	48.45 %	14.85 %
9	12.70 %	8.67 %	10.76 %	6.63 %	66.41 %	31.95 %
8	12.81 %	8.52 %	10.76 %	6.63 %	69.44 %	33.07 %
7	14.49 %	11.07 %	13.16 %	6.63 %	43.51 %	16.88 %
6	13.78 %	11.17 %	12.35 %	6.63 %	36.43 %	20.01 %
5	10.56 %	8.98 %	9.49 %	6.63 %	40.27 %	27.28 %
4	11.22 %	10.36 %	10.10 %	6.63 %	18.88 %	24.42 %
3	12.04 %	9.59 %	9.44 %	6.63 %	45.28 %	48.13 %
2	8.47 %	7.55 %	7.04 %	6.63 %	49.99 %	77.76 %

in realistic examples, the typical value for this LUPI efficiency metric is in the ballpark of 35 %. Also note that if the gap $B - C$ is small compared to C , it means that the privileged information is not particularly relevant; in that case, it is likely hopeless to apply LUPI anyway: there is little space for improvement for that. It is probably safe to start looking for LUPI solution if the gap $B - C$ is at least 1.5 – 2 times larger than C .

We have also implemented C-mapping for SVM for the already described datasets using the same setting as for A-mapping, with the following modifications instead of constructing regressions to privileged features, we constructed (positive linear or nonlinear kernel) regressions to functions $K^*(x_i^*, x^*)$ with subsequent selection of top 40 or them, in terms of their relevance to the label, as was determined by RandomForest method.

Table 2 Performance of ANN and LUPI on modified “Parkinsons” example (A-mapping)

k	ANN on decision features	LUPI (linear)	LUPI (nonlinear)	ANn on all features	LUPI gain (linear)	LUPI gain (nonlinear)
12	19.49 %	16.43 %	15.46 %	8.01 %	26.66 %	35.11 %
11	19.44 %	15.20 %	15.56 %	8.01 %	37.05 %	33.93 %
10	21.33 %	14.64 %	15.66 %	8.01 %	50.18 %	42.52 %
9	20.66 %	12.70 %	13.72 %	8.01 %	62.90 %	54.83 %
8	20.26 %	12.04 %	13.98 %	8.01 %	67.08 %	51.25 %
7	18.57 %	13.01 %	15.05 %	8.01 %	52.65 %	33.33 %
6	20.20 %	13.83 %	13.93 %	8.01 %	52.29 %	51.45 %
5	16.84 %	11.63 %	11.27 %	8.01 %	58.96 %	63.00 %
4	17.35 %	12.45 %	12.50 %	8.01 %	52.46 %	51.91 %
3	12.14 %	11.48 %	11.48 %	8.01 %	16.05 %	16.05 %
2	10.97 %	10.25 %	10.25 %	8.01 %	24.13 %	24.15 %

Table 3 Performance of SVM and LUPI on modified “Parkinsons” example (C-mapping)

k	SVM on decision features	LUPI (linear)	LUPI (nonlinear)	SVM on all features	LUPI gain (linear)	LUPI gain (nonlinear)
12	13.26 %	9.59 %	9.59 %	6.63 %	53.35 %	55.35 %
11	13.52 %	9.79 %	10.10 %	6.63 %	54.14 %	49.64 %
10	13.16 %	11.22 %	9.79 %	6.63 %	29.71 %	51.61 %
9	12.70 %	10.30 %	10.51 %	6.63 %	39.54 %	36.08 %
8	12.81 %	10.20 %	10.20 %	6.63 %	42.23 %	42.23 %
7	14.49 %	9.49 %	11.53 %	6.63 %	63.61 %	37.66 %
6	13.78 %	10.71 %	11.94 %	6.63 %	42.94 %	25.73 %
5	10.56 %	9.18 %	10.20 %	6.63 %	35.11 %	9.16 %
4	11.22 %	8.26 %	10.30 %	6.63 %	64.49 %	20.04 %
3	12.04 %	9.08 %	10.41 %	6.63 %	54.71 %	30.13 %
2	8.47 %	7.57 %	8.18 %	6.63 %	48.91 %	15.76 %

The averaged (over 20 realizations) error rates for these scenarios are shown in Table 3. The collected results show that both linear and nonlinear versions of Knowledge Transfer LUPI with C-mapping improve the performance of SVM on decision inputs (often significantly, in relative terms) in all of the considered scenarios.

6 Conclusions

In this paper, we described several properties of privileged information including its role in machine learning, its structure, and its applications. We extended the existing knowledge transfer research in the area of privileged information (initially considered for SVM) to neural networks and presented a scalable algorithmic framework, which has the same scalability properties as current implementations. The described framework is the first step in the proposed direction, and its further improvements (especially concerning alternative methods of knowledge transfer) will be the subject of future work.

References

1. Brachman, R., Levesque, H.: Knowledge Representation and Reasoning. Morgan Kaufmann, San Francisco (2004)
2. Fouad, S., Tino, P., Raychaudhury, S., Schneider, P.: Incorporating privileged information through metric learning. *IEEE Transactions on Neural Networks and Learning Systems* **24**, 1086–1098 (2013)
3. Demuth, H., Beale, M.: Neural Network Toolbox for Use with MATLAB. Mathworks, Inc., Natick, MA (2000)
4. Hagan, M., Demuth, H., Beale, M.: Neural Network Design. PWS Publishing Co, Boston, MA (1996)
5. Heaton, J.: Introduction to Neural Networks for Java. Heaton Research Inc (2008)
6. Ilin, R., Streltsov, S., Izmailov, R.: Learning with privileged information for improved target classification. *International Journal of Monitoring and Surveillance Technologies Research* **2**(3), 5–66 (2014)
7. Kolmogorov, A.: Mathematics as a Profession. Moscow, Nauka (1988). (in Russian)
8. LeCun, Y., Bottou, L., Orr, G., Müller, K.: Efficient BackProp. In: Orr, G., Müller, K. (eds.) *Neural Networks: Tricks of the Trade*, pp. 5–50. Springer, Heidelberg (1998)

9. Lichman, M.: UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California School of Information and Computer Science (2013)
10. Pechyony, D., Izmailov, R., Vashist, A., Vapnik, V.: SMO-style algorithms for learning using privileged information. In: 2010 International Conference on Data Mining, pp. 235–241 (2010)
11. Ribeiro, B., Silva, C., Chen, N., Vieirac, A., Carvalho das Nevesd, J.: Enhanced default risk models with SVM+. *Expert Syst. Appl.* **39**, 10140–10152 (2012)
12. Sharmanska, V., Lampert, C.: Learning to rank using privileged information. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 825–832 (2013)
13. Vapnik, V.: *The Nature of Statistical Learning Theory*. Springer, New York (1995)
14. Vapnik, V.: *Statistical Learning Theory*. Wiley, New York (1998)
15. Vapnik, V.: *Estimation of Dependencies Based on Empirical Data*, 2nd edn. Springer, New York (2006)
16. Vapnik, V., Vashist, A.: A new learning paradigm: learning using privileged information. *Neural Netw.* **22**, 546–557 (2009)
17. Vapnik, V., Izmailov, R.: Learning with intelligent teacher: similarity control and knowledge transfer. In: *Statistical Learning and Data Sciences*. LNAI 9047, pp. 3–32. Springer, London (2015)
18. Vapnik, V., Izmailov, R.: Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.* **16**, 2023–2049 (2015)
19. Yang, H., Patras, I.: Privileged information-based conditional regression forest for facial feature detection. In: 2013 IEEE International Conference on Automatic Face and Gesture Recognition, pp. 1–6 (2013)