Understanding LUPI
(Learning using Privileged Information)

Ahmadreza Momeni, Kedar Tatwawadi
Stanford University,
Stanford, US
{amomenis,kedart}@stanford.edu

I. INTRODUCTION

The idea of using privileged information was first suggested by V. Vapnik and A. Vashist in [1], in which they tried to capture the essence of teacher-student based learning which is very effective in case of human beings learning. More specifically, when a human is learning a novel notion, he exploits his teacher's comments, explanations, and examples to facilitate the learning procedure. Vapnik proposed the following framework : assume that we want to build a decision rule for determining some labels y based on some features X, but in the training stage in addition to X, we are also provided with some additional information, denoted as the "privileged information" x^* which is not present in the testing stage.

In such a scenario how can we utilize X^* to improve the learning? In this project report, we try to understand the framework of LUPI using a variety of experiments. We also try to propose a new algorithm based on priviledged information for Neural Networks based on the intuition obtained from the experiments.

A. LUPI Framework

We first briefly describe the mathematical framework of LUPI: In the classical binary classification problems we are given m number of pairs (x_i, y_i), $i = 1, \ldots, m$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$, and each pair is independently generated by some underlying distribution P_{XY}, which is unknown. The goal here is to find a function $f : \mathcal{X} \to \{-1, +1\}$ in the function class $\mathcal{F}$ to assign the labels with the lowest error possible averaged over the unknown distribution P_{XY}.

In the LUPI framework, the model is slightly different, as we are provided with triplets (x_i, x_i^*, y_i), $i = 1, \ldots, m$ where $x_i \in \mathcal{X}$, $x_i^* \in \mathcal{X}^*$, $y_i \in \{-1, +1\}$ with each triplet is independently generated by some underlying distribution P_{XX^*Y}, which is again unknown. However, the goal is the same as before: we still aim to find a function $f : \mathcal{X} \to \{-1, +1\}$ in the function class $\mathcal{F}$ to assign the labels with the lowest error possible.

The important question which Vapnik asks is: can the generalization performance be improved using the privileged information? Vapnik also showed this is true in the case of SVM. We will next briefly describe the SVM and the SVM+ LUPI based framework proposed by Vapnik.

B. SVM and SVM+

We briefly describe the SVM and SVM+ methods that we solve for classification, which in this case is finding some $\omega \in \mathcal{X}$ and $b \in \mathbb{R}$ to build the following predictor:

$$f(x) = \text{sgn}\left[\langle \omega, x \rangle + b\right].$$

1) SVM: The SVM learning method (non-separable SVM) to find ω and b is equivalent to solving the following optimization problem:

$$\begin{aligned} &\min \frac{1}{2}\langle \omega, \omega \rangle + C \sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - \xi_i, \qquad i = 1, \ldots, m. \end{aligned}$$

As a short remark, we should mention that C is a parameter that needs tuning. In addition, if the slacks ξ_i are all equal to zero then we call the set of given examples separable, otherwise they are non-separable.

2) SVM+: In order to take into account the privileged information X^* Vapnik modified the SVM formulation as follows:

$$\begin{aligned} &\min \frac{1}{2}\left[\langle \omega, \omega \rangle + \gamma \langle \omega^*, \omega^* \rangle\right] + C \sum_{i=1}^{m} \left[\langle \omega^*, x^* \rangle + b^*\right] \\ &\text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - [\langle \omega^*, x_i^* \rangle + b^*], \qquad i = 1, \ldots, m, \\ &\quad [\langle \omega^*, x_i^* \rangle + b^*] \geq 0, \qquad i = 1, \ldots, m, \end{aligned}$$

where $\omega^* \in \mathcal{X}^*$ and $b^* \in \mathbb{R}$. In this problem C and γ are hyper parameters to be tuned.

Intuitively, we can think of $[\langle \omega^*, x_i^* \rangle + b^*]$'s as some estimators for the slacks ξ_i's in the previous optimization problem. However, the reduced freedom and better prediction of the slacks using the privileged information improves the learning. Another intuition here is that, in some sense the margins $[\langle \omega^*, x_i^* \rangle + b^*]$ capture the difficulty of the training examples in the privileged space. This difficulty information is then used to relax/tighten the SVM constraints to improve the learning.

We next describe some methodologies which capture this intuition relating to difficulty of examples to construt LUPI based frameworks.

C. Weighted SVM and Margin Transfer SVMs

One way in which privileged information influences learning is by differentiating easy examples from the really difficult ones. This understanding was later formalized in [2], where the authors argue that if the weights are chosen appropriately then Weighted-SVM can always outperform SVM+. In weighted SVMs the exmaple weights themselves tell the difficult/importance of the examples. Although [2] proved that weighted SVMs are better than SVM+, the difficulty arises from the fact that the weights are unknown. In some cases though, there are heuristics to guess the weights which work pretty well, and the subject knowledge can often be utilized for this cause.

We next describe a heuristic proposed in [3] to find weights and solve a WSVM problem

1) Margin Transfer SVM: One way to exploit privileged information is proposed in [3], where they suggest to solve a classification problem using only privileged information x^*, and achieve a classifier f^* (note that there is no requirement for f^* to be of the form $\langle \omega^*, x^* \rangle + b^*$. Now, we store the margins $\rho_i := y_i \cdot f^*(x_i^*)$. For our purpose, we put some threshold ϵ on the margins and define $\hat{\rho}_i := \max\{\rho_i, \epsilon\}$. Now we are equipped to solve the following optimization problem:

$$\min \frac{1}{2}\langle \omega, \omega \rangle + C \sum_{i=1}^{m} \hat{\rho}_i \xi_i$$
$$\text{s.t. } y_i[\langle \omega, x_i \rangle + b] \geq 1 - \xi_i, \qquad i = 1, \ldots, m.$$

Intuitively, the margins ρ_i determine how difficult an example is. In extreme cases, if an example is too difficult ($\rho_i < 0$) then its weight $\hat{\rho}_i$ is equal to zero, which means that we are eliminating that example in the training stage. This is similar to human learning procedure, where if an example is too hard then the teacher does not use it because it makes the student diverge from learning the main subject and waste time on some other useless points.

We next describe various experiments which we conducted to understand LUPI.

II. Experiments

A. SVM+ v.s. SVM

The first experiment that we conducted was to compare the performance of SVM+ and SVM. We used the following datasets:

TABLE I
DATASETS

Data	**Test set size**	d^a	d^{*b}
Ionosphere	201	7	6
Ring	7250	10	10
Wine age	108	4	5

[a] The number of the normal features, [b] The number of the privileged features

In each of the above datasets, we chose some feature as normal ones and some of them as privileged, and then trained the classifiers and computed the error on the corresponding test set. For all of the datasets, we used linear kernel. The resulted graphs are as follows:

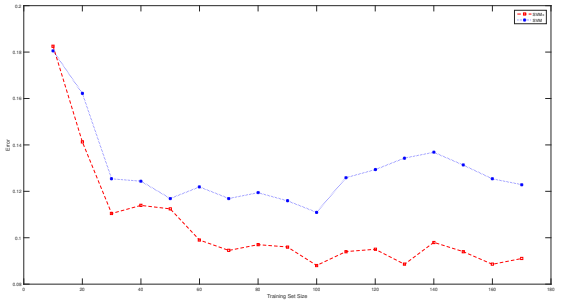

Fig. 1. SVM+ v.s. SVM: Ionosphere data

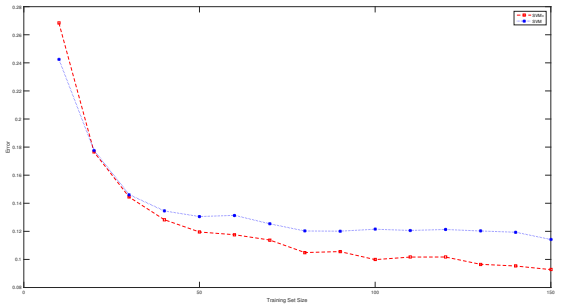

Fig. 2. SVM+ v.s. SVM: Ring data

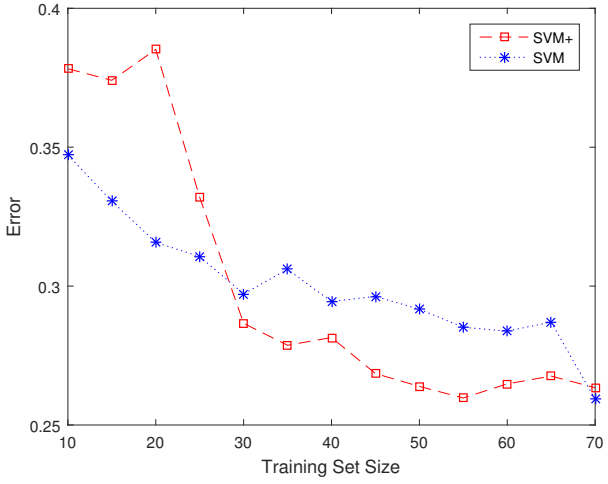

Fig. 3. SVM+ v.s. SVM: Wine Age data

As a brief remark, we note that not only does SVM+ converge faster, but surprisingly in some cases it converges to a better answer, which is observed very distinctly in the Ring experiment. We also observed that SVM+ needs a different solver than SVM and is quite sensitive to the hyper-parameters, which makes it very difficult to get it working for complex datasets.

B. Manually Weighted SVM v.s. SVM+

The second experiment that we conducted aimed to evaluate the performance of Manually Weighted SVM. The aim of the experiment was to ascertain the intuition that difficulty of examples helps in improving the learning. Thus, we considered the ease/difficulty of the training set itself as the privileged information.

We used the following datasets:

TABLE II
DATASETS

Data	Test set size	d^a
Abalone	3178	7
Wine age	118	7

[a] The number of the normal features

The difficulty levels are determined as follows:

- **Abalone Dataset**: In this experiment an abalone is assigned label +1 if its age is above some threshold otherwise the label is -1. We considered the examples the age of which is equal to the threshold to be difficult.
- **Wine age Dataset**: In this experiment a label is +1 if the age of wine is above some threshold otherwise it is -1. We considered the examples the age of which is between the threshold and 0.25 times the standard deviation of the whole dataset age to be difficult.

For both datasets, we used linear kernel. The resulted graphs are as follows:

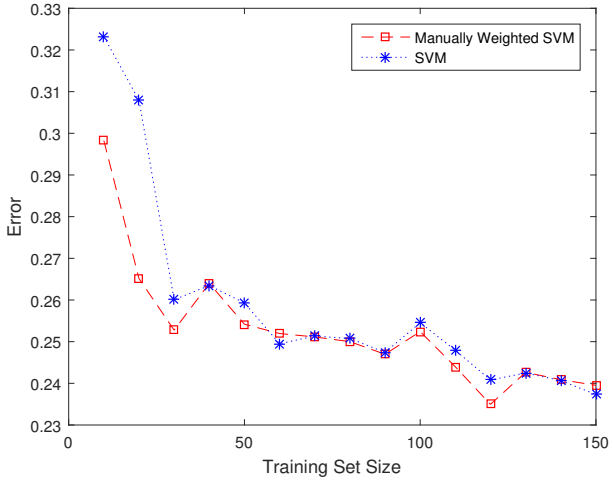

Fig. 4. Manually Weighted SVM v.s. SVM+: Abalone data

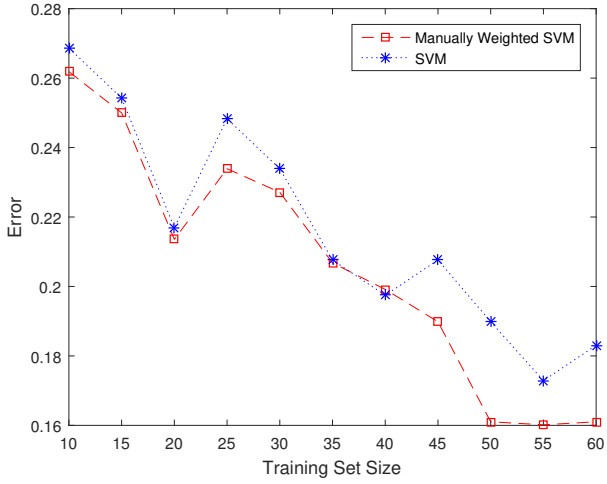

Fig. 5. Manually Weighted SVM v.s. SVM+: Wine age data

Overall, we observed that the privileged information related to difficulty indeed does help the learning in lot of scenarios. Although, for higher data sizes, the improvement is not significant. This in some sense confirmed the intuition stated earlier.

C. LUPI-FNN

In this experiment, we used the intuition outained from the Weighted SVM and the MArgin-Transfer methods to the case of Neural Networks. The basic idea, which is applicable to more general learning frameworks is that: weights can be used to modify the learning rate per-example while applying training procedures based on gradient descent (like: SGD, momentum update, RMS-Prop etc.).

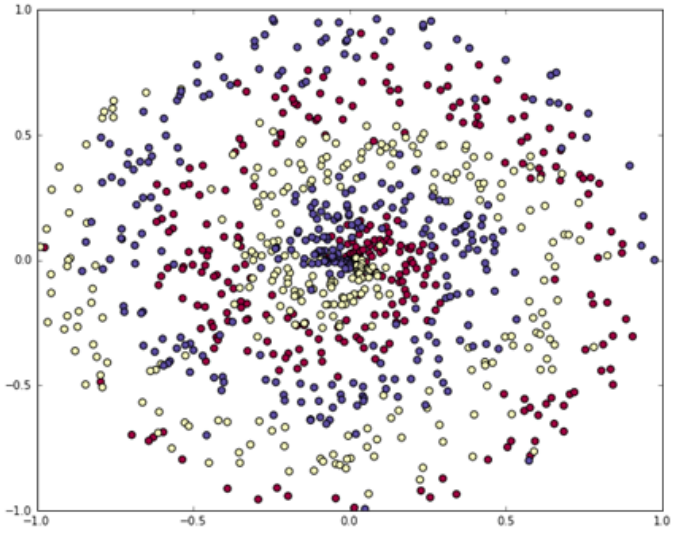

Fig. 6. Input Data (X,y)

In this specific example, we have a spiral dataset containing 3 classes (each denoted by different colors). The

aim is to use neural networks to perform classification. As we observe, although the input dataset itself is complex, the privileged information, which is captured by the polar coordinates (unwarped) representation of the input dataset, is much more easier to classify.

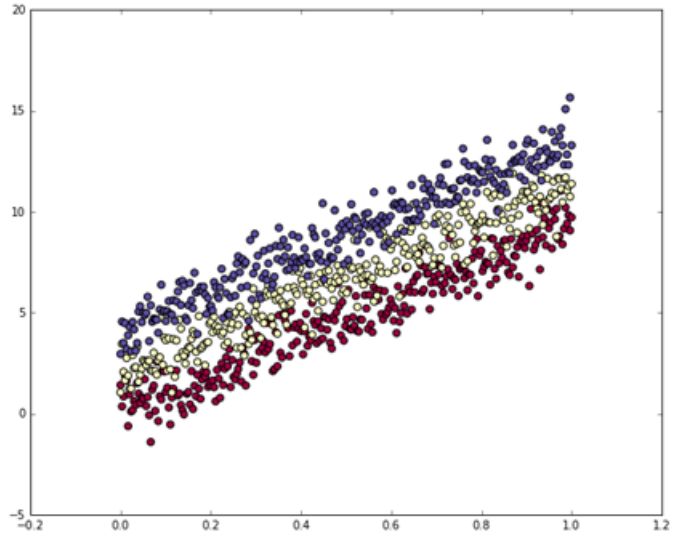

Fig. 7. Privileged Information (X*,y)

Our First step is to fit a 0-layer FCNN in the privileged space $(X*, Y)$. The FCNN consists of a linear classifier followed by a softmax layer to determine the class probabilities. In the experiment, we determined the example weights based on the softmax probability of the correct class. Thus, lower the probability, the harder the example and vice-versa.

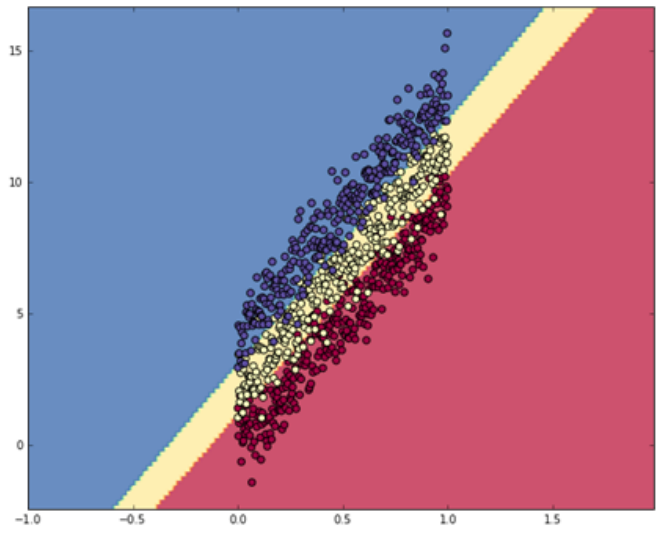

Fig. 8. Learning Weights

The weights obtained from the privileged information were used to train a 1-layer neural network (Linear-ReLU-Linear-Softmax) [Fig9] for the problem. As compared with the baseline neural network [Fig10], we observed improved generalization performance improvement of on an average 3%.

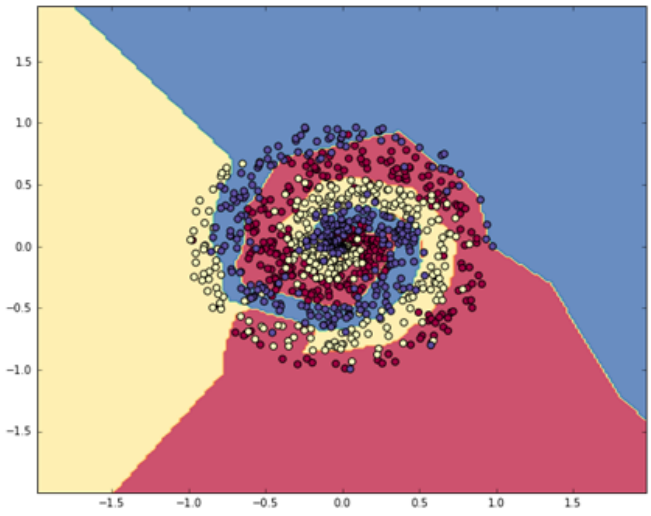

Fig. 9. Weighted NN training

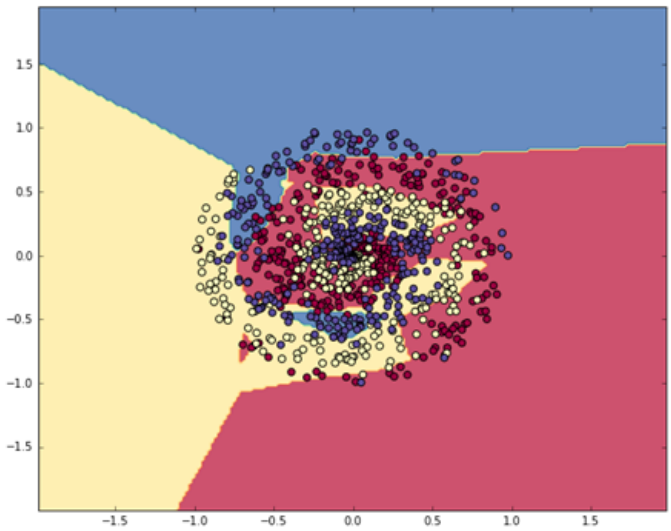

Fig. 10. Reference training without privileged information

III. Conclusion

From the experiments, we gained a lot of intuition into how to use LUPI in practical scenarios. We were also able to formulate LUPI algorithm for neural networks. However, more experiments with real-life data is necessary to confirm the performance of the heuristics applied.

IV. Code

All the source code, including interactive matlab and ipython notebooks are available at: `https://github.com/kedartatwawadi/LUPI`.

We plan to update the github repo with more experiments/tutorials on LUPI.

REFERENCES

[1] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Networks*, vol. 22, no. 5, pp. 544–557, 2009.

[2] M. Lapin, M. Hein, and B. Schiele, "Learning using privileged information: Svm+ and weighted svm," *Neural Networks*, vol. 53, pp. 95–108, 2014.

[3] V. Sharmanska, N. Quadrianto, and C. H. Lampert, "Learning to transfer privileged information," *CoRR*, vol. abs/1410.0389, 2014.