Appears in the Proc. of the Conf. on Artificial Neural Networks In Engg. (ANNIE10), Nov 1-3, St. Louis, MO

THE ADVICEPTRON: GIVING ADVICE TO THE PERCEPTRON

Gautam Kunapuli
University of Wisconsin–Madison
Madison, WI 53705
kunapg@wisc.edu

Kristin P. Bennett
Rensselaer Polytechnic Institute
Troy, NY 12180
bennek@rpi.edu

Richard Maclin
University of Minnesota, Duluth
Duluth, MN 55812
rmaclin@d.umn.edu

Jude W. Shavlik
University of Wisconsin–Madison
Madison, WI 53705
shavlik@cs.wisc.edu

Abstract
We propose a novel approach for incorporating prior knowledge into the perceptron. The goal is to update the hypothesis taking into account both label feedback *and* prior knowledge, in the form of soft polyhedral advice, so as to make increasingly accurate predictions on subsequent rounds. Advice helps speed up and bias learning so that good generalization can be obtained with less data. The updates to the hypothesis use a hybrid loss that takes into account the margins of both the hypothesis and advice on the current point. Analysis of the algorithm via mistake bounds and experimental results demonstrate that advice can speed up learning.

1 INTRODUCTION

We propose a novel online learning method that incorporates advice into a perceptron learner that we call the *Adviceptron*. Prior work has shown that advice is an important and easy way to introduce domain knowledge into learning; this includes work on knowledge-based neural networks (Towell & Shavlik 1994) and prior knowledge in support vector kernels (Schölkopf et al. 1998). More specifically, for SVMs (Vapnik 2000), prior knowledge can be incorporated in three ways (Schölkopf & Smola 2001): by modifying the training data, the kernel or the formulation to be optimized. While we focus on the last approach, we direct readers to a recent and extensive survey (Lauer & Bloch 2008) on prior knowledge in SVMs.

Despite advances to date, research has not addressed how to incorporate advice into incremental SVM algorithms from either a theoretical or computational perspective. In this work, we show that it is possible to effectively incorporate advice as in Knowledge-Based Support Vector Machines (KBSVMs) (Fung et al. 2003) into the classical perceptron (Kivinen 2003). The main motivation for adding advice is that it can introduce bias and reduce the number of samples required. We present a framework for generalizing SVM-type losses to online algorithms with simple, closed-form updates and known convergence properties. We focus on incorporating advice into the binary classification problem and which leads to a new algorithm called the *Adviceptron*. However, these techniques are readily generalized to other learning tasks such as multi-class classification, regression and others.

In the Adviceptron, as in KBSVMs, advice is specified for convex, polyhedral regions in the input space of data. Advice takes the form of simple, possibly conjunctive, implicative rules. Advice can be specified about *every* potential data point in the input space that satisfies certain advice constraints, such as the rule

$$(\texttt{feature}_7 \geq 5) \wedge (\texttt{feature}_{12} \geq 4) \Rightarrow (\texttt{class} = +1),$$

which states that the class should be $+1$ when $\texttt{feature}_7$ is at least 5 *and* $\texttt{feature}_{12}$ is at least 4. Advice can be specified, not only for individual features as above but also for linear combinations of features, while the conjunction of each of these rules allows more complex advice sets.

2 KNOWLEDGE-BASED SVMs

We now describe knowledge-based SVMs as introduced in (Fung et al. 2003). We learn a linear classifier ($\mathbf{w}'\mathbf{x} = b$) given data $(\mathbf{x}^t, y_t)_{t=1}^T$ with $\mathbf{x}^t \in \mathbb{R}^n$ and labels $y_t \in \{\pm 1\}$. In addition, we also have *prior knowledge* specified as follows: *all points* that satisfy constraints of the polyhedral set $D_1\mathbf{x} \leq \mathbf{d}^1$ belong to class $+1$. That is, the advice specifies that $\forall \mathbf{x}$, $D_1\mathbf{x} \leq \mathbf{d}^1 \;\Rightarrow\; \mathbf{w}'\mathbf{x} - b \geq 0$. Advice can also be given about the other class using a second set of constraints: $\forall \mathbf{x}$, $D_2\mathbf{x} \leq \mathbf{d}^2 \;\Rightarrow\; \mathbf{w}'\mathbf{x} - b \leq 0$. Combining both cases using *advice labels*, $z = \pm 1$, advice is given as $D\mathbf{x} \leq \mathbf{d} \;\Rightarrow\; z(\mathbf{w}'\mathbf{x} - b) \geq 0$. We assume that m advice sets $(D_i, \mathbf{d}^i, z_i)_{i=1}^m$ are given.

Advice in implication form cannot be incorporated into an SVM directly; this is done by exploiting theorems of the alternative. Observing that $p \Rightarrow q$ is equivalent to $\neg p \vee q$, we require that that the latter be true; this is same as requiring that the negation $(p \wedge \neg q)$ be false or that the system of equations

$$\left\{\; D\mathbf{x} - \mathbf{d}\,\tau \leq 0,\;\; z\mathbf{w}'\mathbf{x} - zb\,\tau < 0,\;\; -\tau < 0 \;\right\} \text{ has no solution } (\mathbf{x}, \tau). \tag{1}$$

The variable τ is introduced to bring the system to nonhomogeneous form. Applying the nonhomogeneous Farkas theorem of the alternative (Mangasarian 1969) to (1) we have

$$\left\{\; D'\mathbf{u} + z\mathbf{w} = 0,\;\; -\mathbf{d}'\mathbf{u} - zb \geq 0,\;\; \mathbf{u} \geq 0 \;\right\} \text{ has a solution } \mathbf{u}. \tag{2}$$

The set of (hard) constraints above incorporates the advice specified by *a single advice set*. If there are m advice sets, each of the m rules is added as the equivalent set of constraints of the form (2). When these are incorporated into a standard SVM, the formulation becomes a hard-KBSVM (hard because the advice is assumed to be linearly separable). As with data, linear separability of advice is a limiting assumption and can be relaxed by introducing slack variables to *soften the constraints* (2). If $\mathcal{P}$ and $\mathcal{L}$ are convex regularization and loss functions respectively, the KBSVM is

$$\begin{array}{lll} \underset{(\boldsymbol{\xi}, \mathbf{u}^i, \boldsymbol{\eta}^i, \zeta_i) \geq 0, \mathbf{w}, b}{\text{minimize}} & \mathcal{P}(\mathbf{w}) & +\lambda\, \mathcal{L}_{data}(\boldsymbol{\xi}) + \mu \displaystyle\sum_{i=1}^{m} \mathcal{L}_{advice}(\boldsymbol{\eta}^i,\, \zeta_i) \\ \text{subject to} & & Y(X\mathbf{w} - b\mathbf{e}) + \boldsymbol{\xi} \geq \mathbf{e}, \\ & & D_i'\mathbf{u}^i + z_i\mathbf{w} + \boldsymbol{\eta}^i = 0, \\ & & -\mathbf{d}^{i\prime}\mathbf{u}^i - z_i b + \zeta_i \geq 0,\; i = 1, \ldots, m, \end{array} \tag{3}$$

where $Y = diag(\mathbf{y})$ and $\mathbf{e}$ is a vector of ones of the appropriate dimension. There are two regularization parameters λ, $\mu \geq 0$, which tradeoff the data and advice errors with the regularization. While converting the advice from implication to constraints, we introduced new variables: the *advice vectors* $\mathbf{u}^i \geq 0$. These perform the same role as the dual multipliers α in the classical SVM. Recall that points with non-zero α's are the *support vectors* which additively contribute to $\mathbf{w}$. Here, for *each* advice set, the constraints of the set which have non-zero $\mathbf{u}^i$s are called *support constraints*.

2.1 Learning From Knowledge Only

We assume that the vectors $\mathbf{u}^i$ are computed based on the advice alone *before* any data are available. This is possible because we can learn purely from the advice or by sampling the advice regions to generate pseudo-data. Given only the advice, the following formulation is used to generate the advice vectors in a pre-processing step:

$$\begin{array}{ll} \underset{\mathbf{u}^i \geq 0, \mathbf{w}, \boldsymbol{\eta}^i, \zeta_i}{\text{minimize}} & \frac{1}{2}\|\mathbf{w}\|_2^2 + \mu \sum_{i=1}^{m} \left(\left\|\boldsymbol{\eta}^i\right\|_2^2 + \zeta_i^2 \right) \\ \text{subject to} & D_i' \mathbf{u}^i + z_i \mathbf{w} + \boldsymbol{\eta}^i = 0, \\ & -\mathbf{d}^{i\prime} \mathbf{u}^i - z_i b + \zeta_i \geq 0,\ i = 1, \ldots, m. \end{array} \tag{4}$$

We denote the optimal solution to (4) as $(\mathbf{w}^\star, \mathbf{u}^{i,\star})$. It provides two important pieces of the Adviceptron: the advice-only hypothesis which is used to initialize the algorithm ($\mathbf{w}^1 = \mathbf{w}^\star$, rather than $\mathbf{w}^1 = 0$ as in most approaches without advice), and $\mathbf{u}^{i,\star}$, the advice vectors, that are incorporated into the perceptron at every iteration under certain conditions (see Section 3). These advice vectors constrain the current hypothesis, $\mathbf{w}^t$ at every iteration during learning via $D_i' \mathbf{u}^{i,\star} + z_i \mathbf{w} + \boldsymbol{\eta}^i = 0,\ i = 1, \ldots, m$. The equation above is the primary mechanism through which advice is incorporated into the perceptron.

As in most online settings, we drop the bias term, b. Once it is dropped and the $\mathbf{u}^i$'s are fixed to $\mathbf{u}^{i,\star}$, the second constraint drops out the formulation (4). Thus, in order to incorporate advice in the online setting, we just need to ensure that the current hypothesis $\mathbf{w}^t$ is "consistent" with advice constraints.

3 DERIVING THE ADVICEPTRON

For the rest of the section, we assume that at trial t, we have a current hypothesis $\mathbf{w}^t$ as well as the pre-computed advice vectors $\mathbf{u}^{i,\star}$. We are given a labeled instance $(\mathbf{x}^t,\, y_t)$ which is used to update (if necessary) and obtain a new hypothesis $\mathbf{w}^{t+1}$. As in the classical perceptron case, we assume that in the event of misclassification that is, if $y_t \mathbf{w}^{t\prime} \mathbf{x}^t \leq 0$, there is an update and in the event of correct classification, $y_t \mathbf{w}^{t\prime} \mathbf{x}^t > 0$, there is no update. Define σ_t^h as follows

$$\sigma_t^h = \begin{cases} 1, & \text{if } y_t \mathbf{w}^{t\prime} \mathbf{x}^t \leq 0, \quad \text{(misclassification)} \\ 0, & \text{if } y_t \mathbf{w}^{t\prime} \mathbf{x}^t > 0, \quad \text{(correct classification)} \end{cases} \tag{5}$$

In the classical perceptron, updates happen only when $\sigma_t^h = 1$. Our advice-taking approach differs from the perceptron case in that even when $\sigma_t^h = 0$, the learner might wish to update $\mathbf{w}^t$ if there is a misclassification *according to the advice*.

3.1 Advice Updates When $\mathbf{w}^t$ Classifies Correctly ($\sigma_t^h = 0$)

Consider the case where, according to $\mathbf{w}^t$, the current point $(\mathbf{x}^t, y^t)$ is correctly classified ($y_t \mathbf{w}^{t\prime} \mathbf{x}^t > 0$). In the classical perceptron, there is no update. To extend the problem to include advice, we consider the following optimization problem which tries to ensure that the current hypothesis $\mathbf{w}^t$ is consistent with the given m pieces of advice, $(D_i, \mathbf{d}^i, z_i)_{i=1}^m$. We wish to update $\mathbf{w}^t$ to

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}^t\|_2^2 + \frac{\mu}{2} \sum_{i=1}^{m} \|\boldsymbol{\eta}^i\|_2^2, \quad \text{s. t. } D_i' \mathbf{u}^{i,\star} + z_i \mathbf{w} + \boldsymbol{\eta}^i = 0,\ i = 1, \ldots, m. \quad (6)$$

Eliminating $\boldsymbol{\eta}^i$ results in an unconstrained problem whose optimal solution can computed from the gradient condition. We denote the *advice-estimate of the hypothesis* according to the i-th advice set to be $\mathbf{r}^i = -z_i D_i' \mathbf{u}^{i,\star}$. Denote the overall *average advice-estimate* to be $\mathbf{r} = \frac{1}{m} \sum_{i=1}^m \mathbf{r}^i = -\frac{1}{m} \sum_{i=1}^m z_i D_i' \mathbf{u}^{i,\star}$. The advice-estimates, $\mathbf{r}^i$, represent information about each advice set as a point in the *hypothesis space* i.e., each $\mathbf{r}^i$ provides an estimate of the hypothesis according to that particular advice set. The centroid of these points is the *average advice-estimate*, $\mathbf{r}$. Finally, define the *advice-influence parameter* to be $\nu = \frac{1}{1+m\mu}$.

Now, the optimal solution to (6) can be compactly represented as $\mathbf{w} = \nu \mathbf{w}^t + (1-\nu)\mathbf{r}$, which is a *convex combination* of the current hypothesis and the average advice-estimate. The margin of the updated hypothesis $\mathbf{w}^{t+1}$ is $\gamma_a = \nu\, y_t \mathbf{w}^{t\prime} \mathbf{x}^t + (1-\nu)\, y_t \mathbf{r}' \mathbf{x}^t$, a convex combination of the margin of the current data point $\mathbf{x}^t$ w.r.t. $\mathbf{w}^t$, and the margin w.r.t. $\mathbf{r}$. This provides a condition under which we could apply the update rule (6) if $y_t \mathbf{w}^{t\prime} \mathbf{x}^t > 0$ *and*:

$$\begin{array}{ll} \text{if } \gamma_a \leq 0, & \text{perform update according to } \mathbf{w} = \nu\, \mathbf{w}^t + (1-\nu)\, \mathbf{r}, \\ \text{if } \gamma_a > 0, & \text{there is no advice update.} \end{array}$$

Thus, even if the classical perceptron performs no update, if, according to the advice, the new point is misclassified, $\mathbf{w}^t$ is updated to reflect this. Also note that for any $\mu > 0$, the value of $\nu \in [0, 1)$ and advice influence can be tuned by choosing μ.

3.2 Advice Updates When $\mathbf{w}^t$ Misclassifies ($\sigma_t^h = 1$)

The derivations in the previous subsection can be extended to deriving updates when $\mathbf{x}^t$ is misclassified by both the current hypothesis *and* advice. Such an update should ensure that the new hypothesis classifies the current point correctly and is consistent with the advice-estimate $\mathbf{r}$. This update is the optimal solution to:

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}^t\|_2^2 - \lambda y_t \mathbf{w}' \mathbf{x}^t + \frac{\mu}{2} \sum_{i=1}^{m} \|\boldsymbol{\eta}^i\|_2^2 \text{ s. t. } D_i' \mathbf{u}^{i,\star} + z_i \mathbf{w} + \boldsymbol{\eta}^i = 0 \ \ i = 1, \ldots, m. (7)$$

Here, the formulation has the additional term $-\lambda y_t \mathbf{w}'\mathbf{x}^t$, where $\lambda > 0$ is the learning rate. The new term introduces an additional restriction that the updated hypothesis not only classify the point $(\mathbf{x}^t, y_t)$ correctly, but do so with as large a margin as possible while being consistent with the advice-estimate. In a manner similar to the last subsection, we can derive the update rule for the case when $\sigma_t^h = 1$ to be $\mathbf{w} = \nu\,(\mathbf{w}^t + \lambda y_t \mathbf{x}^t) + (1-\nu)\,\mathbf{r}$.

3.3 A Unified Update Rule

We can combine the two update rules using σ_t^h to write down a unified update rule. Thus, if $\nu\, y_t \mathbf{w}^{t\prime}\mathbf{x}^t + (1-\nu)\, y_t \mathbf{r}'\mathbf{x}^t \leq 0$, update using $\mathbf{w}^{t+1} = \nu(\mathbf{w}^t + \lambda\sigma_t^h y_t \mathbf{x}^t) + (1-\nu)\mathbf{r}$. Recall that $\sigma_t^h = 1$ if there is a perceptron update. Analogous to this, we can define σ_t^a, which is 1 when there is an *advice update*:

$$\sigma_t^a = \begin{cases} 1, & \text{if } y_t(\nu\,\mathbf{w}^t + (1-\nu)\,\mathbf{r})'\mathbf{x}^t \leq 0, \\ 0, & \text{if } y_t(\nu\,\mathbf{w}^t + (1-\nu)\,\mathbf{r})'\mathbf{x}^t > 0. \end{cases} \tag{8}$$

We can now formulate the Adviceptron (see Algorithm 1).

Algorithm 1 The Adviceptron Algorithm

input data $(\mathbf{x}^t, y_t)_{t=1}^T$, advice sets $(D_i, \mathbf{d}^i, z_i)_{i=1}^m$, λ, $\mu > 0$
pre-compute $(\mathbf{u}^{i,\star}, \mathbf{w}^\star)$ as optimal solution to (4)
let $\mathbf{r}^i = -z_i D_i' \mathbf{u}^{i,\star}$, $\mathbf{r} = 1/m \sum_{i=1}^m \mathbf{r}^i$
let $\nu = 1/(1+m\mu)$
let initial hypothesis, $\mathbf{w}^1 = \mathbf{w}^\star$
for $(\mathbf{x}^t, y_t)$ **do**
  predict label $\hat{y}_t = \text{sign}(\mathbf{w}^{t\prime}\mathbf{x}^t)$
  receive correct label y_t
  compute σ_t^h and σ_t^a using (5) and (8)
  if $\sigma_t^a = 0$ (there is no advice update) **then**
    update $\mathbf{w}^{t+1} = \mathbf{w}^t + \lambda\sigma_t^h y_t \mathbf{x}^t$
  else if $\sigma_t^a = 1$ (there is an advice update) **then**
    update $\mathbf{w}^{t+1} = \nu(\mathbf{w}^t + \lambda\sigma_t^h y_t \mathbf{x}^t) + (1-\nu)\mathbf{r}$
  end if
end for

We can analyze the performance of the Adviceptron by comparing to some hypothesis $\mathbf{w}^*$ which has a margin γ on the given instances, i.e., $y_t \mathbf{w}^{*\prime}\mathbf{x}^t > \gamma$ holds for $t = 1, \ldots, T$. Let the number of mistakes made by the current hypothesis $\mathbf{w}^t$ be $M = \sum_{t=1}^T \sigma_t^h$ and the number of mistakes made by the advice-influenced hypothesis, $\nu\mathbf{w}^t + (1-\nu)\mathbf{r}$, be $M_a = \sum_{t=1}^T \sigma_t^a$.

Theorem 1. *Let $\mathfrak{S} = \{(\mathbf{x}^t, y_t)\}_{t=1}^T$ be a sequence of examples with $(\mathbf{x}^t, y_t) \in \mathbb{R}^n \times \{\pm 1\}$, and $\|\mathbf{x}^t\|_2 \leq X \quad \forall t$. Let $\mathfrak{A} = \{(D_i, \mathbf{d}^i, z_i)\}_{i=1}^m$ be m advice sets with advice vectors $\mathbf{u}^{i,\star} \geq 0$, with $\mathbf{r}^i = -z_i D_i' \mathbf{u}^{i,\star}$ the i-th advice-estimate of the hypothesis and let $\mathbf{r} = \frac{1}{m}\sum_{i=1}^m \mathbf{r}^i$, be*

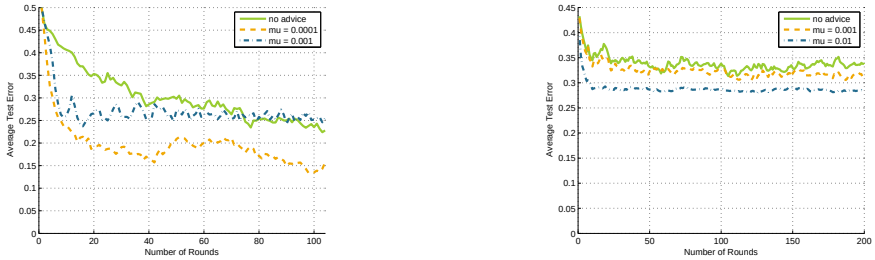

Figure 1: (**left**) `promoters` (**right**) `diabetes`. For a sufficiently small choice of μ, initial error is much lower and the Adviceptron converges faster than the perceptron.

the average advice-estimate. If some $\mathbf{w}^* \in \mathbb{R}^n$ *with* $\|\mathbf{w}^* - \mathbf{r}\|_2 \leq A$ *has a margin* γ *on* $\mathfrak{S}$*, the* Adviceptron *makes at most*

$$M \leq \frac{X^2 A^2}{\gamma^2} \left(1 + (1-\nu) M_a\right)$$

mistakes on $\mathfrak{S}$*, where* $\nu = 1/(1+m\mu)$*,* $\mu > 0$*.*

The proof is not presented here due to lack of space. The algorithm's behavior can be analyzed by investigating the bound closely; the following conclusions can be drawn:

- Smaller values of A tighten the bound because $\mathbf{w}^*$ is more "consistent" with the average advice-estimate $\mathbf{r}$. If $\mathbf{w}^* = \mathbf{r}$, we recover the original perceptron bound.
- Fewer advice updates, M_a, tighten the bound. The more consistent the current hypothesis $\mathbf{w}^t$ is with the advice $\mathbf{r}$, the less likely it is that there will be an advice update. Intuitively, this will be because an advice update occurs only when the convex combination of the margins according to $\mathbf{w}^t$ and $\mathbf{r}$ is negative ($\nu\, y_t \mathbf{w}^{t\prime} \mathbf{x}^t + (1-\nu)\, y_t \mathbf{r}' \mathbf{x}^t \leq 0$). If at the t-th trial, if $\mathbf{w}^t$ is sufficiently influenced by the advice, there will be no mistake according to the advice ($\sigma_t^a = 0$) and no advice update.

4 EXPERIMENTS

We show experiments using three real world data sets: the Promoters-106 data set and Pima Indians Diabetes data set, both available from the UCI repository (Asuncion & Newman 2007) and the USPS digit recognition data set (Hull 1994). `promoters` consists of 106 data points for 57 characters of a gene sequence. The data set is translated into 228 features, a binary bit for each location in the sequence indicating whether that location is an `A`, `G`, `T`, or `C`. A domain theory for the data set is also available from UCI and consists of 14 rules, which can easily be transformed into a set of 64 rules; see (Fung et al. 2003). The `diabetes` consists of 768 points with 8 attributes. For domain advice, we constructed two rules based on statements from the NIH web site on risks for Type-2 Diabetes[1]. A person who is obese, characterized by high body mass index (`BMI` ≥ 30) and a high `bloodglucose` (≥ 126) is at strong

[1] `http://diabetes.niddk.nih.gov/DM/pubs/riskfortype2`

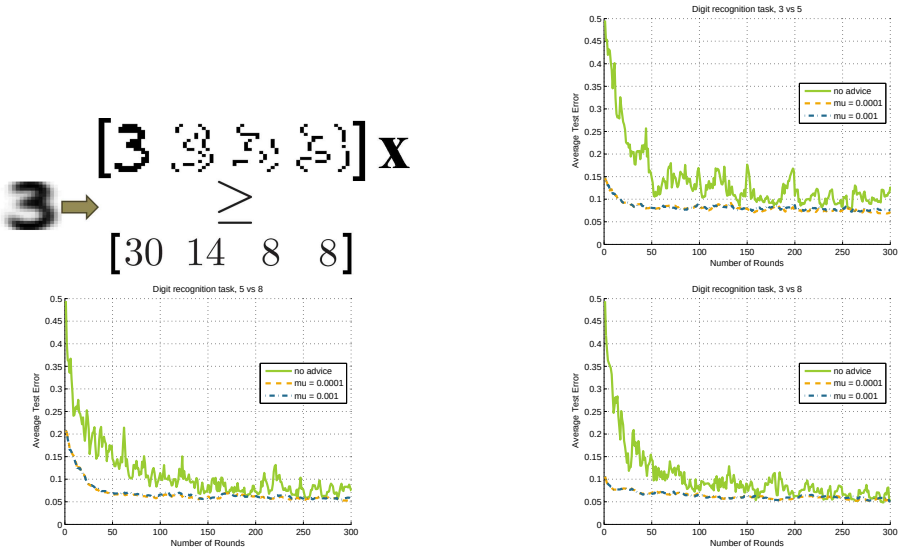

Figure 2: Results for three pairwise digit recognition tasks from the `USPS` data set. (**top right**) 3 vs 5. (**bottom left**) 5 vs 8. (**bottom right**) 3 vs 8.

risk for diabetes, while a person who is at normal weight (`BMI` ≤ 25) and a low `bloodglucose` level (≤ 100) is unlikely to have diabetes. Rules are constructed based on the above advice: e.g., (`BMI` ≤ 25) $\wedge$ (`bloodglucose` ≤ 100) $\Rightarrow \neg$`diabetes`.

The `usps` data consists of 9298 points described by 16×16 pixel greyscale images to represent handwritten digit images. We investigate 3 tasks: predicting 3 vs 5, 3 vs 8 and 5 vs 8. To create rules for these tasks we hand drew ten images each of the digits (more specifically, our view of what the canonical digits should look like) and then performed a blurring of the images together using PCA. We took the principal components of the resulting image and used them as templates for rules based on matching expected and observed pixels (see Figure 2, top left).

For each experiment, we ran 25 iterations (for `promoters`, 100 iterations owing to small sample size), each using a different random 15% of the data as a test set (1% for `promoters`). The advice leads to significantly faster convergence of accuracy over the no-advice perceptron learner. This reflects the intuitive idea that a learner, when given prior knowledge that is useful, will be able to more quickly find a good solution. In each case, note also that the learner is able to use the learning process to improve on the starting accuracy (which would be produced by advice only). Thus, the Adviceptron is able to learn effectively from both data *and* advice.

A second point is that the knowledge allows the learner to converge on a level of accuracy that is not achieved by the perceptron, which does not have the benefit of advice. Although we would expect that with large amounts of data, the perceptron might eventually be able to achieve performance similar to the Adviceptron, this shows that advice can provide large improvements over just learning with data.

5 CONCLUSIONS

In this paper we have presented a new online learning method, the Adviceptron, that is a novel approach that makes use of prior knowledge in the form of polyhedral advice. Our approach differs from previous polyhedral advice-taking approaches such as KBSVM (Fung et al. 2003) and the neural-network-based KBANN system (Towell & Shavlik 1994) in two significant ways: it is an online method with closed-form solutions and it provides a theoretical mistake bound. We have shown theoretically and experimentally, that that a learner with reasonable advice can significantly outperform a learner without advice. We believe our approach can serve as a template for other methods to incorporate advice into online learning methods.

KBSVMs can be extended to kernels as demonstrated in (Fung et al. 2003) and we can directly use this approach to kernelize the Adviceptron. Other research directions include extending the approach to other tasks such as regression and multiclass problems. Yet another avenue is the case of *advice refinement* wherein the advice is no longer fixed but also updated incrementally based on the data.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge support of the Defense Advanced Research Projects Agency under DARPA grant FA8650-06-C-7606. Views and conclusions contained in this document are those of the authors and do not necessarily represent the official opinion or policies, either expressed or implied of the US government or of DARPA.

References

Asuncion, A. & Newman, D. (2007), 'UCI machine learning repository'.
URL: *http://archive.ics.uci.edu/ml/*

Fung, G., Mangasarian, O. L. & Shavlik, J. W. (2003), Knowledge-based support vector classifiers, *in* 'NIPS', Vol. 15, pp. 521–528.

Hull, J. J. (1994), 'A database for handwritten text recognition research', *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(5), 550–554.

Kivinen, J. (2003), Online learning of linear classifiers, *in* 'Advanced Lectures on Machine Learning', Springer-Verlag New York, Inc., pp. 235–257.

Lauer, F. & Bloch, G. (2008), 'Incorporating prior knowledge in support vector machines for classification: A review', *Neurocomp.* **71**(7-9), 1578–1594.

Mangasarian, O. L. (1969), *Nonlinear Programming*, McGraw-Hill.

Schölkopf, B., Simard, P., Smola, A. & Vapnik, V. (1998), Prior knowledge in support vector kernels, *in* 'NIPS', Vol. 10, pp. 640–646.

Schölkopf, B. & Smola, A. (2001), *Learning with kernels: Support Vector Machines, Reuglarization Optimization and Beyond*, MIT Press, Cambridge, MA, USA.

Towell, G. G. & Shavlik, J. W. (1994), 'Knowledge-based artificial neural networks', *AIJ* **70**(1-2), 119–165.

Vapnik, V. (2000), *The Nature of Statistical Learning Theory*, Springer-Verlag.