Explanation-Based Learning for Image Understanding

Qiang Sun and **Li-Lun Wang** and **Gerald DeJong**
Department of Computer Science, University of Illinois at Urbana-Champaign
201 N. Goodwin Ave., Urbana, IL 61801, USA
qiang@alexa.com, lwang4@uiuc.edu, mrebl@uiuc.edu

Abstract

Existing prior domain knowledge represents a valuable source of information for image interpretation problems such as classifying handwritten characters. Such domain knowledge must be translated into a form understandable by the learner. Translation can be realized with Explanation-Based Learning (EBL) which provides a kind of dynamic inductive bias, combining domain knowledge and training examples. The dynamic bias formed by the interaction of domain knowledge with training examples can yield solution knowledge of potential higher quality than can be anticipated by the static bias designer without seeing training examples. We detail how EBL can be used to dynamically integrate domain knowledge, training examples, and the learning mechanism, and describe the two EBL approaches in (Sun & DeJong 2005a) and (Sun & DeJong 2005b).

Introduction

Pre-existing domain knowledge represents an attractive source of classification information. When examples are expensive or limited the information in a training set may be insufficient to confidently learn a classifier. Incorporating available prior knowledge may result in a more accurate and more confident classifier.

The choice of a kernel function in support vector machines (SVMs) is often cited as an opportunity to employ domain knowledge. Here we report two previously published methods for incorporating domain knowledge into SVMs. One learns a specialized kernel function while the other employs a kernelized explanation bias utilizing a conventional polynomial kernel function. These were part of Sun's thesis work, and Wang is continuing to pursue this project.

Figure 1 illustrates the opportunity for additional domain information in SVM classification of handwritten digits. The top two lines represent one learning problem, and the bottom two lines represent another. In both the task is to say which images correspond to the numeral "3" and which to the numeral "6." The second problem is derived from the first by applying a fixed random permutation to all of the image pixels. Since the SVM kernel function operates on the dot product of feature vectors, these two learning problems are *indistinguishable* to the SVM. It performs equally well on both (LeCun *et al.* 1995). But clearly, the first affords additional information to humans. Indeed, the second is impossible for humans to learn, although they achieve nearly perfect performance with very little training on tasks like the first one using simple foreign shapes.

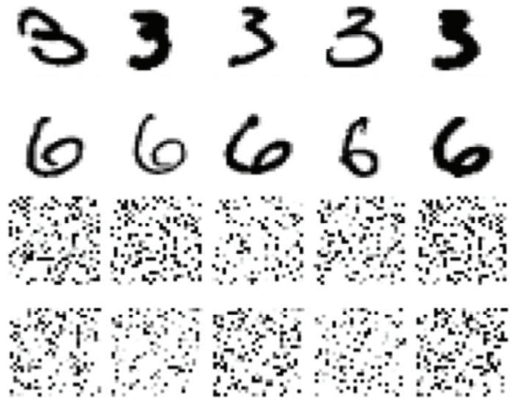

Figure 1: Sample handwritten images of numerals "3" and "6." The bottom two lines are the top two with pixels permuted.

Sun & DeJong (2005a) consider two sorts of prior knowledge. The first, which they call *solution knowledge*, concerns the target of learning itself and is specific to the learning task at hand. Examples of solution knowledge include the structure of a Bayes net, the kernel function of an SVM, the topology of a neural network, etc. The other sort, *domain knowledge*, describes objects of the world. For example, in handwritten character recognition one may believe that the pixels in the input images arise from strokes of a writing implement.

One possible way to exploit prior knowledge is through inductive bias. Inductive bias is an unavoidable part of any learning system (Mitchell 1997). Solution knowledge can be easily integrated into the machine learning process as inductive bias. Domain knowledge, while generally more reliable and more easily articulated by a human expert, cannot often be expressed as an inductive bias. For example, although we know that the pixels of a handwritten character are manifestations of the srokes from a writing implement and that these strokes are intended to portray some idealized figure, it is difficult to translate such knowledge into an effective inductive bias (e.g., a better kernel function or a more appropriate neural network topology). This knowledge alone cannot help to classify any particular image as a "3" or a "6" or anything else.

Explanation-Based Learning dynamically integrates domain knowledge into the learning process by allowing its interaction with training examples. An *explanation* justifies, in terms of the domain knowledge, why a particular training example might merit its assigned training label. If the explanation is correct, then other examples that satisfy its conditions should form a conceptual equivalence class; they all should be given the same classification label for the same reason. Through conjecturing and confirming such equivalence classes, additional guidance can be given to the machine learner: a learner should prefer classifiers whose labels better respect the partitionings induced by confirmed explanations.

The approach in (Sun & DeJong 2005b) learns specialized *Feature Kernel Functions*. Any kernel function is essentially a distance metric. The SVM finds a large margin classifier, one that maximizes the distance to the most constraining training examples. A feature kernel function is a specially constructed distance metric that is automatically tailored to the learning task at hand. The feature kernel approach estimates which image regions will likely be more helpful in distinguishing between the classes. The specialized distance metric assesses a greater distance contribution from pixels in regions of high expected discriminative information.

The approach in (Sun & DeJong 2005a), the Explanation-Augmented Support Vector Machine (EA-SVM), does not incorporate prior knowledge by crafting the kernel function. It uses a conventional kernel but imposes an additional preference over the space of possible classifiers. EA-SVMs use the fact that the equivalence class on examples that is induced by an explanation forms a lower dimensional surface in the hypothesis space: since all of the examples that satisfy the explanation are (ideally) labeled the same way for the same reason, they should have the same margin. If the domain knowledge were perfect, in the SVM's high dimensional space, an explanation would be a lower dimensional hyperplane to which the correct classifying hyperplane must be parallel. Due to imperfect prior knowledge, noise, etc. a soft loss function is used to asses a penalty on classifiers not parallel to explanation surfaces. Thus, the explanation surfaces impose an additional preference over the space of high-dimensional linear classifiers. Through kernelization, the explanation preferences can be handled efficiently in the dimensionality of the input space.

Feature Kernel Functions

EBL requires a domain theory of prior knowledge to drive the explanation process. While images of characters are composed of pixels, it is natural to break the span from pixels to characters into two subdomains. The first relates handwritten characters to the *strokes* that are used to form them. The notion of a stroke is not intrinsic to this classification problem. A conventional SVM has no place for such a notion. Rather, strokes are introduced as "hidden" features to organize prior knowledge. In addition, combinations of strokes form yet another level of derivable hidden features, called *stroke-level features*. For example, the two prototype Chinese characters shown in figure 2 are quite similar. However, some stroke-level features are quite informative for this discrimination task. We have circled these in figure 2. Described with these derived stroke-level features the characters are quite different.

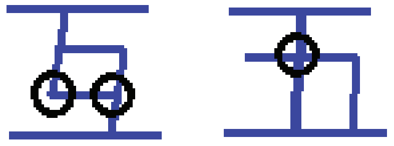

Figure 2: Two similar Chinese characters with the difference between them high-lighted.

The second sub-domain explains the correspondence between pixels and strokes. Strokes are modeled as straight lines of a particular width with a particular starting and ending location. The corresponding pixels are those that fall within the boundaries of a long thin rectangle which is the stroke. The correspondence is determined by applying a Hough transformation (Forsyth & Ponce 2002) to detect lines in the training images. Note that using a Hough transform on training images is far more reliable than to find lines in an unknown image. The label of the training image provides access to its prototype stroke representation. Thus, the procedure is reduced to finding image lines that best match the known stroke lines.

Given a pair of Chinese character labels and a set of training examples for each, the following procedure is performed to produce a feature kernel function:

1. Conjecture stroke-level features: The prototypes of the characters to be distinguished are examined, and distinctive stroke interactions are identified. These are candidates for the construction of component kernel functions.
2. Determine the best pixel representation for each stroke: This involves
   (a) Explaining labeled character images by using Hough transform to determine how each required stroke is realized by its pixels.
   (b) Estimating the correlation between pixels and strokes across the data set as $\Pr(x = 1, f = 1) - \Pr(x = 1, f = -1)$ where x is an input feature (pixel) and f is a high-level feature (stroke).
   (c) Constructing parameterized evidence pixel sets for each stroke. The parameter is a threshold specifying the minimal acceptable correlation for a pixel to be included.
3. Given evidence pixel sets for strokes, construct the set of *component kernel functions*. It also involves choosing values for the parameters mentioned above. They are evaluated so as to minimize stroke-level feature detection errors over whole training set.

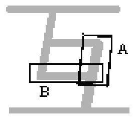

Figure 3: Two stroke-level features to detect a corner.

4. Build the feature kernel function: Component kernel functions of the previous step are weighted using (Kandola, Shawe-Taylor, & Cristianini 2002) and combined using *alignment* (Cristianini, Shawe-Taylor, & Elisseeff 2002) to produce the final feature kernel function.

The component kernel described in step 3 is a specialized kernel to serve as a detector for a stroke-level feature, and is used to assemble the final feature kernel function. Conceptually, component kernels operate over *monomials*. A monomial represents the conjunction of pixels, as evidence for a stroke-level feature.

Given the connections between pixels and strokes, it is straightforward to determine evidence monomials for different stroke-level features. This is illustrated in figure 3. When the pixels in the region A serve as evidence for a horizontal stroke, and the pixels in the region B serve as evidence for a vertical stroke, then those second-degree monomials with one pixel from region A and another from region B can be evidence for the corner feature composed with the horizontal and vertical strokes

Specifying a function to compute the dot product between two examples using those evidence monomials gives us a component kernel function for SVMs to detect the corresponding stroke-level features. The corner feature shown in figure 3 employs a component kernel function like the following:

$$k_{A,B}(\vec{x_1}, \vec{x_2}) = (\sum_{i \in A} x_{1i} x_{2i})(\sum_{i \in B} x_{1i} x_{2i})$$

It is easy to see that this kernel function computes the dot product between example x_1 and x_2 using the monomials of pixels from region A and region B:

$$\begin{aligned} k_{A,B}(\vec{x_1}, \vec{x_2}) &= \sum_{i \in A, j \in B} x_{1i} x_{2i} x_{1j} x_{2j} \\ &= \sum_{i \in A, j \in B} (x_{1i} x_{1j})(x_{2i} x_{2j}) \end{aligned}$$

Figure 4 shows the classification error rate classifying two characters using different kernels with different number of training examples in one experiment in (Sun & DeJong 2005b). It is demonstrated in the experiments that the feature kernel approach outperforms an SVM using polynomial kernel both in terms of accuracy and efficiency.

Explanation-Augmented SVM

EA-SVM uses *generalized* or *explained* examples and let SVMs treat them much as it treats conventional examples. In explained examples, only the important features identified by the explanations are allowed to contribute to the kernel

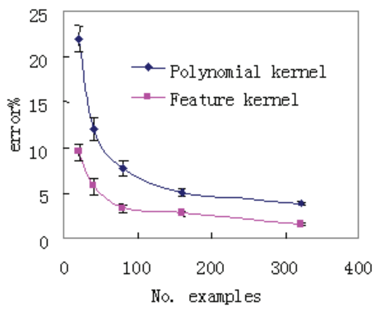

Figure 4: Feature kernel improves both accuracy and example efficiency.

computation. Given an original example x, and a subset of important features $e \subseteq x$ from an explanation, the explained example v is constructed thus:

$$\begin{cases} v_i = x_i, & \text{if } x_i \in e \\ v_i = *, & \text{otherwise} \end{cases}$$

The special symbol "$*$" indicates that this feature does not participate in the inner product evaluation. With numerical features one can simply use the value zero.

An explained example can be viewed as a generalization of an original example, in the sense that examples that satisfy the same explanations merit the same label for the same reasons and thus should be treated equivalently by the learner. Consider an SVM's linear separator in its high-dimensional feature space (Vapnik 1998). In the ideal case, an explanation is a lower dimensional linear surface to which the correct classifier *should be parallel*. To see why, consider the extensional definition of the explanation which is the set of all examples that satisfy the explanation's requirements. Assuming the ideal case, the SVM's feature space and linear separator are adequate to capture all of the relevant distinctions and relations. All of the examples that merit this label for the same reasons should be treated identically. In the high dimensional feature space they should have the same margin from the correct classifier. This means that they fall on a parallel linear surface. This surface will be of a lower dimension if there are any redundancies or irrelevancies in the high dimensional feature space with respect to this explanation. Such explanations constrain the correct classifier, and therefore, once discovered, can guide the learner. A simplified example in three dimensional space is illustrated in figure 5, where the explanation specifies that one feature as relevant, therefore the constraint surface is a two dimensional plane.

In EA-SVM the constructed explanations are treated as preferences or soft constraints rather than hard constraints on the correct classifier. Their effect are blended on the SVM classifier with the conventionally-treated training set.

The EA-SVM is formulated in a way analogous to the soft margin SVM. In the case of perfect explanations, the learned classifier evaluates the original example and the generalized example to the same value: $w \cdot x_i + b = w \cdot v_i + b$, or equivalently $w \cdot (x_i - v_i) = 0$.

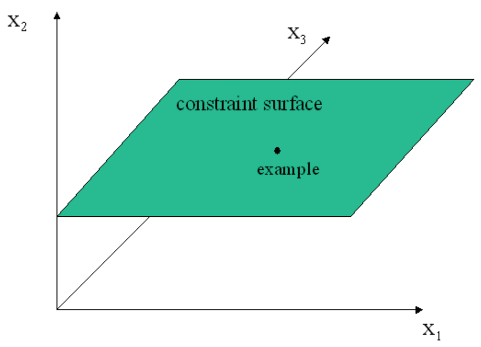

Figure 5: A simple example to illustrate the parallel constraint introduced by explanation. The example lives in $\{x_1, x_2, x_3\}$ space. When the explanation indicates that only feature x_2 is important, it suggests that every points on the 2-dimensional constraint surface perpendicular to x_2 should be treated the same by the classifier.

Geometrically, this requires the classifier hyperplane to be parallel to the direction $x_i - v_i$. These are called *parallel constraints*. The SVM quadratic problem becomes:

$$\begin{array}{ll} \min & \frac{1}{2}\|w\|^2 \\ \text{subject to} & y_i(w \cdot x_i + b) - 1 \geq 0, \forall i; \\ & w \cdot x_i - w \cdot v_i = 0, \forall i. \end{array}$$

If the domain knowledge is imperfect, the constraints cannot all be met. The explained examples should then be treated as a bias to be respected as much as possible. This is similar to the standard SVM algorithm for the non-separable case (Vapnik 1998). New slack variables (δ_i) measure the difference between the evaluations of the original examples and the generalized examples:

$$\forall i, w \cdot x_i - w \cdot v_i \geq -\delta_i, w \cdot x_i - w \cdot v_i \leq \delta_i, \delta_i \geq 0$$

To penalize violations of the constraints, the objective function is changed from $\|w\|^2/2$ to $\|w\|^2/2 + Q\sum_i \delta_i$; Q is called the *confidence parameter*. It reflects confidence in (or assessed quality of) the domain knowledge. It will be set automatically by cross validation. A larger Q corresponds to better knowledge and a greater penalty for disagreeing with the explanations. Now the primal problem becomes:

$$\begin{array}{ll} \min & \frac{1}{2}\|w\|^2 + Q\sum_i \delta_i \\ \text{subject to} & y_i(w \cdot x_i + b) - 1 \geq 0, \forall i; \\ & w \cdot x_i - w \cdot v_i \geq -\delta_i, \forall i; \\ & w \cdot x_i - w \cdot v_i \leq \delta_i, \forall i; \\ & \delta_i \geq 0, \forall i. \end{array}$$

It is also easy to combine with slack variables for non-separable data. Slack variables ξ_i are introduced to penalize the errors: $y_i(x_i \cdot w + b) \geq 1 - \xi_i, \forall i; \xi_i \geq 0, \forall i$. The objective function then becomes: $\|w\|^2/2 + Q\sum_i \delta_i + C\sum_i \xi_i$.

Sun & DeJong (2005a) show that the quadratic programming problem for EA-SVM described above is convex, and thus can be solved by the standard SVM methods. They also show empirically that explanations help more in difficult learning problems than in easy ones, that improvement graduates, with more accurate domain knowledge helping more than less accurate domain knowledge, and that even entirely inaccurate domain knowledge does not result in EA-SVM behavior that is unduly worse than the performance of a conventional SVM.

Conclusion

Explanation-Based Learning approaches can incorporate existing prior domain knowledge into statistical learning processes like the SVM via kernel functions, as well as via other means like additional preference over the space of possible classifiers. The experiments in (Sun & DeJong 2005a) and (Sun & DeJong 2005b) show that the interaction of domain knowledge with training examples introduced by EBL yields solution knowledge of higher quality and helps to obtain good classifiers effectively, especially in difficult learning problems even if the domain knowledge is not perfect.

The approaches described above are just two of the many possible EBL approaches to exploiting existing domain knowledge to benefit statistical learning. We are only beginning to explore the relative strengths and weaknesses of these approaches. Other very different approaches may well be found that possess more interesting tradeoffs and more attractive performance. Our future research will continue to explore this promising direction. We also plan to explore how these techniques can be applied to more demanding image processing tasks, where prior knowledge may be even more valuable.

References

Cristianini, N.; Shawe-Taylor, J.; and Elisseeff, A. 2002. On kernel-target alignment. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 367–373.

Forsyth, D. A., and Ponce, J. 2002. *Computer Vision—A Modern Approach*. Prentice-Hall.

Kandola, J.; Shawe-Taylor, J.; and Cristianini, N. 2002. Optimizing kernel alignment over combinations of kernels. Technical Report NC-TR-02-121, NeuroCOLT.

LeCun, Y.; Jackel, L.; Bottou, L.; Brunot, A.; Cortes, C.; Denker, J.; Drucker, H.; Guyon, I.; Muller, U.; Sackinger, E.; Simard, P.; and Vapnik, V. 1995. Comparison of learning algorithms for handwritten digit recognition. In Fogelman, F., and Gallinari, P., eds., *International Conference on Artificial Neural Networks*, 53–60.

Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.

Sun, Q., and DeJong, G. 2005a. Explanation-augmented SVM: an approach to incorporating domain knowledge into SVM learning. In de Raedt, L., and Wrobel, S., eds., *Proceedings of the 22nd International Machine Learning Conference*. ACM Press.

Sun, Q., and DeJong, G. 2005b. Feature kernel functions: Improving SVMs using high-level knowledge. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 177–183.

Vapnik, V. N. 1998. *Statistical Learning Theory*. New York: Wiley.