Explanation-Based Feature Construction

Shiau Hong Lim, Li-Lun Wang and **Gerald DeJong**
Dept. of Computer Science
University of Illinois, Urbana-Champaign
{shonglim,lwang4,mrebl}@cs.uiuc.edu

Abstract

Choosing good features to represent objects can be crucial to the success of supervised machine learning algorithms. Good high-level features are those that concentrate information about the classification task. Such features can often be constructed as non-linear combinations of raw or native input features such as the pixels of an image. Using many nonlinear combinations, as do SVMs, can dilute the classification information necessitating many training examples. On the other hand, searching even a modestly-expressive space of nonlinear functions for high-information ones can be intractable. We describe an approach to feature construction where task-relevant discriminative features are automatically constructed, guided by an explanation-based interaction of training examples and prior domain knowledge. We show that in the challenging task of distinguishing handwritten Chinese characters, our automatic feature-construction approach performs particularly well on the most difficult and complex character pairs.

1 Introduction

Sensitivity to appropriate features is crucial to the success of supervised learning. The appropriateness of a feature set depends on its relevance to the particular task. Various notions of relevance have been proposed [Blum and Langley, 1997; Kohavi and John, 1997] and statistical tools to evaluate and select relevant feature sets are available (e.g. hypothesis testing, random probes, cross-validation). Unfortunately, most feature construction strategies focus on feature subset selection (e.g. filters [Almuallim and Dietterich, 1991; Kira and Rendell, 1992], wrappers [Kohavi and John, 1997], embedded methods [Guyon et al., 2006]) from either a predetermined set or a space defined by some feature construction operators [Markovitch and Rosenstein, 2002] and require many training examples to evaluate.

In challenging domains, the "native", observable features (e.g. pixel values of an image) are high-dimensional and encode a large amount of mostly irrelevant information. There are standard statistical techniques such as principal components analysis (PCA) and linear discriminant analysis (LDA) to reduce the dimensionality of the input features. But often the high-level features that simplify the task are complex nonlinear combinations of the native features. We believe the most effective approach is to incorporate additional information in the form of prior world knowledge. In this direction, one approach [Gabrilovich and Markovitch, 2005] utilizes the large repository of the Open Directory Project to aid in natural language classification. We address a very different problem of handwritten Chinese character recognition. Our additional world knowledge is derived from a relatively small, imperfect, and abstract prior domain theory.

Explanation-based learning (EBL) is a method of dynamically incorporating prior domain knowledge into the learning process by explaining training examples. In our character recognition task for example, we know that not all pixels in the input image are equally important. [Sun and DeJong, 2005] used this observation to learn specialized Feature Kernel Functions for support vector machines. These embody a specially constructed distance metric that is automatically tailored to the learning task at hand. That approach automatically discovers the pixels in the images that are more helpful in distinguishing between classes; the feature kernel function magnifies their contribution.

However, we know that it is not the raw pixels that intrinsically distinguish one character from one another. Rather, the pixels are due to strokes of a writing implement and these strokes in relation to one another distinguish the characters. Not telling the learner of these abstract relationships means that it must perform the moral equivalent of inventing the notion of strokes from repeated exposure to patterns of raw pixels, making the learning task artificially difficult. Our system learns a classifier that operates directly on the native features but which appreciates the abstractions of the domain theory as illustrated by the training examples.

In section 2, we describe what makes a good high-level feature. We then detail the role of explanations and give our general algorithm in section 3. In section 4 we illustrate our approach with detailed feature construction algorithms for Chinese character recognition, and in section 5 we show some experimental results. Section 6 concludes.

2 High Level Features and Generalization

The criteria for feature construction are usually based on how well a particular set of features retains the information

about the class while discarding as much irrelevant and redundant information as possible. Information-theoretic methods, based on mutual information, channel coding and rate-distortion theorems have been applied to this problem [Battiti, 1994; Tishby et al., 1999; Torkkola, 2003]. Let X and Y be random variables that represent the input and the label respectively, and there is an underlying joint-distribution on (X, Y). It can be shown [Feder and Merhav, 1994] that the optimal Bayes error is upper bounded by $\frac{1}{2}H(Y|X)$. Therefore, a feature $F(X)$ that retains as much information as possible about the label will have $H(Y|F(X)) \leq H(Y|X)+\epsilon$, where $\epsilon > 0$ represents the amount of information loss that we are willing to tolerate. On the other hand, discarding irrelevant information can be achieved by minimizing $H(F(X)|Y)$ while satisfying the condition on $H(Y|F(X))$. Intuitively, this implies that most information about the class label is preserved, while irrelevant information (e.g. within-class variations) is discarded. We show how this bounds the actual risk for the binary case in the Appendix, regardless of what hypothesis space is used.

When comparing alternative features, each satisfying $H(Y|F(X)) \leq H(Y|X) + \epsilon$, we therefore prefer the one with the smallest $H(F(X)|Y)$. Alternatively, we may aim at minimizing the following functional:

$$J(F) = H(F(X)|Y) + \frac{1}{\epsilon}H(Y|F(X))$$

Unfortunately, without additional knowledge about the underlying probability distribution, it is impossible to accurately estimate the conditional entropy empirically from the data when the training set is small. For example, when all the training examples have different X, one can simply define a feature F based on the nearest neighbor rule (itself a classifier) and empirically, with a naive estimator, achieve $\hat{J}(F) = 0$. There is no reason to believe that the feature $F(X)$ and the resulting classifier will generalize to unseen examples. In this sense, building a right feature is as hard as building the right classifier, that is, it does not work without any inductive bias.

However, it is possible that for some tasks, there exist features that are known *a priori* to have $J(F) \approx 0$. Such features capture our knowledge about patterns that are unique to a particular class of objects. What is not known, however, is a reliable way to *detect* or *extract* the feature from any unlabeled input. The problem of constructing a good feature can therefore be viewed as the problem of building a good detector for such "high-level" features, based on the training data. If we can ensure that the detector produces the right output *for the right reason*, i.e., it detects the intended high-level feature, then we will have high confidence that the resulting feature will generalize well.

How do we verify that a detector produces the right output for the right reason? Doing so statistically, if possible at all, will require too many training examples for many tasks. Instead, if available domain knowledge can be used to build *explanations* for the training examples, then they can be used to verify whether a feature's output is consistent with our prior knowledge. We show how to use this idea to automatically construct features that focus on the most informative part of any input object with regard to the particular task.

3 Explanation-based Feature Construction

In classical EBL, an "explanation" is a logical proof that shows how the class label of a particular labeled example can be derived from the observed inputs. But our version is weaker. We only require the explanation to identify potential low level evidence for the assigned classification label. We then use training data to calibrate and evaluate the strength of that evidence.

Our prior knowledge includes ideal stroke models of the characters of interest (roughly of the sort one obtains from a vector font) and the model of a stroke as a connected straight line of finite width.

Feature construction is performed by the following steps which we state abstractly and describe specifically in the context of the Chinese character domain.

1. **Explain each training example to obtain the association between the assigned label and the observed native features mediated by high-level domain theory concepts.** After this step, each pixel is associated with a stroke (the line that most likely made it) constrained so that the line configurations match the stroke requirements of the assigned character.
2. **Using the prior knowledge, identify 1) high-level features that are similar in both categories, and 2) high-level features that are different between the categories.** The first set form our reference strokes. These can be confidently found in new test images since a similar stroke should be present for either instance type. The second set are information-bearing; they are character dependent.
3. **With the generated explanations, evaluate each potential similarity statistically using the training set, keeping the ones that can be detected efficiently and with high confidence.** These are strokes that are easily found in an unlabeled image and form a frame of reference to guide us to the high class-distinguishing information.
4. **Using the training examples, optimize the process of finding detection features from the reference features.** This identifies the high information image regions w.r.t. the reference strokes. The regions chosen to be larger if over the training set there is greater variance of the location of the detection strokes w.r.t. the reference strokes and tighter otherwise.

Generally, finding lines in an image is problematic. Many lines will be missed and often non-lines will be found. The process can be expensive. However, this is only done for *labeled* examples during the learning phase. Thus, we know what lines we should find and their approximate geometrical configuration. This greatly improves the line-finder's performance. But what if we can find no reference strokes? If there are no easily-found similarities between the categories, then the two classes must be very different; the classification task should be simple. Many features and algorithms should be able to differentiate them, and our knowledge-based approach

is unnecessary. Next we examine this process in more detail for Chinese characters.

4 Classifying Handwritten Chinese Characters

Offline handwritten Chinese character recognition remains a challenging task due to the large variability in writing styles and the similarity of many characters. Approaches are either structural or feature-based [Suen et al., 2003]. The former extract strokes from a new input image and try to match the extracted strokes to the known stroke-level models. Extracting strokes is unreliable and consequently the model-matching process is problematic. Feature-based approaches utilize statistics on well-designed features and have proven to be more robust. However, the features are hand-crafted and it is not easy to exploit prior knowledge during the learning process; similar characters are difficult to differentiate using such globally-defined features.

In this paper, we focus our attention to the task of distinguishing pairs of similar characters. In our approach automatically constructed features are tailored to best differentiate the characters.

Consider the pair of Chinese characters in Figure 1.

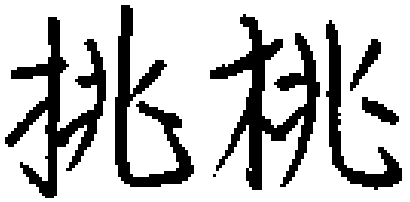

Figure 1: Two very similar Chinese characters

They are almost identical except the leftmost radical. Extracting a global feature that summarizes the whole character dilutes the class-relevant information concentrated on the far left of the image.

Once the informative region has been identified, there is still much variability in the *exact* location of the informative region. Figure 2 illustrates the variability among the first character of the pair.

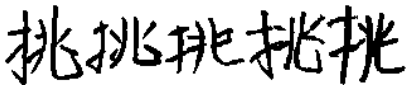

Figure 2: Within-class Variability

Attempting to define an absolute set of pixels (say, one 3rd of the image from the left) would result in noisy features. Too small the region we risk missing the important stroke for some of the characters, too large the region our advantage of focused feature is lost. This is where we utilize our knowledge about similarities between the two characters. The three long, roughly vertical strokes present in both characters may serve as "reference strokes." Finding them allows the target region to be more accurately identified.

4.1 Building Explanations

Our prior model of each character is a graph, where nodes represent a stroke and edges represent the relationship between strokes. Each stroke is itself modeled as a line segment with 5 parameters (x, y, θ, l, t) denoting its center, direction, length and thickness. Such models can either be hand-specified, or obtained from existing character-stroke database.[1] The model need not be highly accurate as the explanation process relies primarily on its structural information. The model is used to explain each training example by finding the most likely parameters for each requisite character stroke.

In general, searching for the best set of parameters is a combinatorial problem for the general graph. This may still be acceptable since the size of the graph is small and the process is done only once for each character during training.

For efficiency, we structure these graphs into trees to employ a dynamic programming approach to produce the explanations. We use an algorithm based on [Felzenszwalb and Huttenlocher, 2000]. Our implementation uses two automatically generated trees, focusing on horizontally and vertically oriented strokes separately. Figure 3 shows a character model and an example explanation.

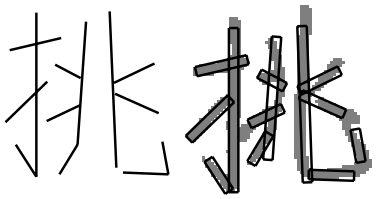

Figure 3: A Character Model and an Explained Example

4.2 Identifying Potential Similarities

Given the models for a particular pair of characters, we perform graph matching to identify strokes that are similar in terms of location, direction and length. The result of this process is the identification of a set of strokes $\mathcal{M}$ which have a match in both characters. We refer to this set as the matching set $\mathcal{M}$. These are the candidates for reference strokes. Figure 4 shows an example.

4.3 Finding Efficient Reference Stroke Detector

Any efficient feature extractor can be used in this step. Since we are concerned with lines, we use a Hough transform [Forsyth and Ponce, 2002]. In particular, we perform a Hough transform on an input image, and look for a local minimum in a specified region which reflects the variability of the matching set stroke in the training data. We refer to this as the "Hough detector". Not every stroke in $\mathcal{M}$ can be reliably detected by the Hough detector. We use the following algorithm

[1] For example, the Wen Quan Yi Project, http://wenq.org/

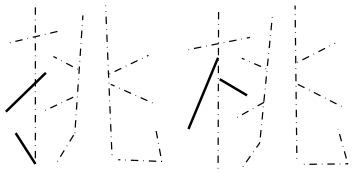

Figure 4: The strokes in $\mathcal{M}$ are shown as dotted lines

to select from the matching set a set of reference strokes that can be reliably detected. The explanation for each training example is used to measure how accurately the Hough detector detects a particular stroke.

1. Initialize the set of reference stroke $\mathcal{R}$ to empty
2. For each stroke S in $\mathcal{M}$
   (a) Find the range of directions and offsets for this stroke among all the training examples (from the explanations), namely, find the smallest bounding rectangle with the center $(\bar{\theta}, \bar{\rho})$, the width $\Delta\theta$, and the height $\Delta\rho$.
   (b) For each $\alpha \in \{0.8, 1, 1.2, 1.4, 1.6\}$ and each $\beta \in \{0.8, 1, 1.2, 1.4, 1.6\}$
      i. Define the bounded region in Hough space as a rectangle centered at $(\bar{\theta}, \bar{\rho})$ with width $\alpha\Delta\theta$ and height $\beta\Delta\theta$.
      ii. For each training example
         A. Search for the highest peak in the region
         B. Check whether the peak is within a threshold distance τ from the actual stroke orientation
      iii. Record the hit ratio $H(\alpha, \beta)$ (percentage of reference strokes correctly detected using the specified parameters)
   (c) Find α^* and β^* with the highest $H(\alpha, \beta)$
   (d) if $H(\alpha^*, \beta^*) > H_0$ then add S to $\mathcal{R}$

The range of the detector window (α and β) in the above algorithm is chosen for simplicity. The thresholds τ (distance in Hough space) and H_0 are optimized using cross-validation.

4.4 Learning the final Feature

Once reference strokes are identified, the system estimates the informative region relative to the parameters of the reference strokes. We use a simple definition for our "informative region". In each character, every stroke that is not in $\mathcal{M}$ is considered a potentially informative stroke. From the explanations, we know the location of these strokes in the training examples. Using these locations, we find the smallest rectangle that includes each stroke. Whenever there are overlapping strokes, the two rectangles are combined into a single larger rectangle bounding both strokes. Figure 5 illustrates this.

Feature points, which can be the center or endpoints of a reference stroke, receive a distance score with respect to each edge of the target window, where the score is define as $a\delta + b\sigma$. This combines the mean distance (δ) to the window edge and its standard deviation given the reference strokes. The

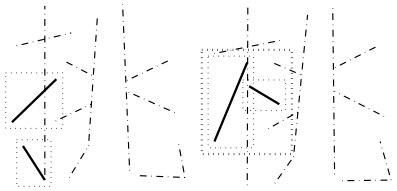

Figure 5: The ideal "target" rectangles

feature point with the smallest score is selected. If none of the feature points qualifies, then the edge of the image is used as the definition of the target window. Figure 6 illustrates this.

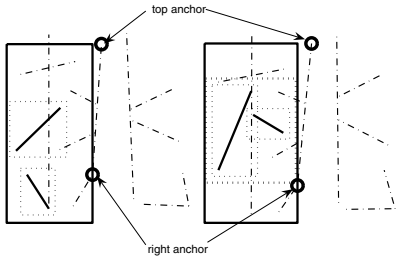

Figure 6: The actual "target" rectangles with respect to the feature-points. Note that there are no feature-points for the left and the bottom edge.

We assume a joint-distribution on the location of the reference strokes and the target "window" as defined by the feature-points. Each reference stroke is parameterized by the direction and the offset $R_i = (\rho_i, \theta_i)$ obtained from the Hough detector. Each target window is parameterized by 4 parameters $L = (l_1, l_2, l_3, l_4)$ which correspond to left, top, right and the bottom of the window. For example, k reference strokes and one target window form a joint-distribution on (R, L) where $R = (R_1, R_2, \ldots, R_k)$. The joint distribution is estimated from the training examples, since we know both R and L from the explanations. For unlabeled examples, we first apply the Hough detector to localize the reference strokes, $r = (r_1, \ldots, r_k)$, then find the maximum-likelihood location of the window according to the conditional probability $P(L|R = r)$. Assuming that the joint-distribution is Gaussian with mean

$$\begin{pmatrix} \mu_R \\ \mu_L \end{pmatrix}$$

and covariance

$$\begin{pmatrix} \Sigma_{RR} & \Sigma_{RL} \\ \Sigma_{LR} & \Sigma_{LL} \end{pmatrix} \quad ,$$

the conditional is itself Gaussian with mean

$$\mu_{L|R} = \mu_L + \Sigma_{LR}\Sigma_{RR}^{-1}(r - \mu_R)$$

and covariance

$$\Sigma_{L|R} = \Sigma_{LL} - \Sigma_{LR}\Sigma_{RR}^{-1}\Sigma_{RL} \quad .$$

Pair	SVM	SVM(EBL)	Pair	SVM	SVM(EBL)
1	18.0	18.0	16	6.8	6.8
2	17.8	**17.0**	17	6.8	6.8
3	14.3	14.3	18	6.5	6.5
4	13.0	**5.8**	19	6.5	6.5
5	13.0	13.0	20	6.3	**5.3**
6	11.0	**8.3**	21	6.3	6.3
7	8.5	8.5	22	6.3	6.3
8	8.0	**7.0**	23	6.0	6.0
9	7.8	7.8	24	5.8	5.8
10	**7.5**	8.8	25	5.8	**5.0**
11	**7.5**	8.5	26	5.8	**5.3**
12	7.0	**3.0**	27	5.5	**3.5**
13	7.0	**5.8**	28	5.5	5.5
14	7.0	**2.0**	29	5.5	5.5
15	7.0	**3.8**	30	5.5	**5.3**

Table 1: Error Rate (%) (5-fold cross-validation)

5 Experiments

We evaluate our system on pairwise classifications between difficult pairs of characters. We use the ETL9B database (a popular database of more than 3000 Chinese and Japanese characters, each with 200 examples). We first learn a literature-standard multiclass classifier using linear discriminants to identify 100 most difficult pairs (i.e. pairs of characters that result in greatest confusion). These are, as expected, pairs of very similar characters. We use the weighted direction code histogram (WDH) [Kimura et al., 1997] as features. These features are generally the best or among the best for handwritten Chinese character recognition [Ding, 2005].

For each pair of characters, we learn a classifier using a linear support vector machine. We observe that the support vector machine performs significantly better than those with linear discriminants. For our system, we use the same classifier, but the input to the SVM are the WDH features extracted only from the *target* feature window found with respect to the detected reference strokes. Table 1 shows the results of the experiment on the 30 most challenging pairs of characters. Even though the SVM is generally robust in the presence of irrelevant features, our system managed to achieve significant improvement on many of these pairs.[2]

6 Conclusion

We believe that the key to generalization is to incorporate available domain knowledge into the learning process. We show that tailored discriminative features can be constructed from comparisons of generative models built according to the prior domain knowledge. This approach is particularly attractive in problems where training examples are few but high level domain knowledge is available, as demonstrated in the task of classifying handwritten Chinese characters.

[2]We note that in several cases the examples in the database are mislabaled and our algorithm could achieve better results if these are corrected, but we decided not to modify the original database to retain its integrity.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Award NSF IIS 04-13161. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix

We show how a feature $f(X)$ with small $H(f(X)|Y)$ and small $H(Y|f(X))$ leads to better generalization bound in terms of Rademacher complexity, for the specific case of binary classification. We make use of the following theorem:

Theorem 1. *[Bartlett and Mendelson, 2002] Let P be a probability distribution on $\mathcal{X} \times \{\pm 1\}$, let F be a set of $\{\pm 1\}$-valued functions defined on $\mathcal{X}$, and let $(X_i, Y_i)_{i=1}^n$ be training samples drawn according to P^n. With probability at least $1-\delta$, every function f in F satisfies*

$$P(Y \neq f(X)) \leq \hat{P}_n(Y \neq f(X)) + \frac{R_n(F)}{2} + \sqrt{\frac{ln(1/\delta)}{2n}}$$

where $\hat{P}_n$ is the empirical risk. $R_n(F)$ is the Rademacher complexity of F, given by

$$R_n(F) = \mathbf{E}_{\sigma_{1\ldots n}} \mathbf{E}_{X_{1\ldots n}} \Big[\sup_{f \in F} |\frac{2}{n} \sum_{i=1}^{n} \sigma_i f(X_i)| \,\Big| X_1, \ldots, X_n \Big]$$

where $\sigma_1, \ldots, \sigma_n$ are independent $\{\pm 1\}$-valued random variables.

The following lemma shows that $R_n(\mathcal{G})$ is bounded if $H(f(X)|Y)$ is bounded.

Lemma 1. *Given discriminative feature f^* with $H(f^*(X)|Y) \leq \beta < 1$, where Y takes values from $\{\pm 1\}$, the Rademacher complexity of a set $\mathcal{G}$ of $\{\pm 1\}$-valued functions is bounded as follows*

$$R_n(\mathcal{G}) \leq \beta + \frac{4}{n}$$

Proof. From the concavity of entropy, it can be shown that given the label $Y = y$, there exists v such that

$$\begin{aligned} P_{v|y} &= Pr(f^*(x) = v_y|y) = \max_{u: f^*(x)=u} Pr(f^*(x) = u|y) \\ &\geq 1 - \tfrac{\beta}{2} \quad . \end{aligned} \tag{1}$$

We will call this v_y the "typical" value of $f^*(x)$ given y. Given the training samples $(X_1, Y_1), \ldots, (X_n, Y_n)$, and the Rademacher random variables $\sigma_1, \ldots, \sigma_n$, let n^- and n^+ denote the expected number of examples with $Y = -1$ and $Y = +1$ according to $P(Y = y)$. The expected number of examples where $f^*(x)$ is typical is then given by $n_t^- = n^- P_{v|-1}$ and $n_t^+ = n^+ P_{v|+1}$ respectively.

Consider the Rademacher average defined in Theorem 1. Suppose that the hypothesis space $\mathcal{G}$ contains all $\{\pm 1\}$-valued functions, that is, there is always a hypothesis $g^* \in \mathcal{G}$ that attains the maximum Rademacher average. However, *regardless* of the function g^* chosen, only one label can be assigned

to X_i with the same value, in particular, those with the typical value. The value $\sigma_i = \pm 1$ is assigned with equal probability to all X_i, therefore the expected number of examples that are wrongly labeled is $N = \frac{n_t^- + n_t^+ - 2}{2}$. Since there are at least N other examples with typical value that will be correctly labeled, the expected sum of these will cancel each other. From equation (1), $n_t^- \geq n^-(1 - \frac{\beta}{2})$ and $n_t^+ \geq n^+(1 - \frac{\beta}{2})$. The Rademacher average is therefore

$$\begin{aligned} R_n(\mathcal{G}) &\leq \tfrac{2}{n}\big(n - 2N\big) = \tfrac{2}{n}\big(n - 2(\tfrac{n_t^- + n_t^+ - 2}{2})\big) \\ &\leq \tfrac{2}{n}\big(n - (n^- + n^+)(1 - \tfrac{\beta}{2}) + 2\big) = \beta + \tfrac{4}{n} \quad . \end{aligned}$$

□

Theorem 2. *Given discriminative feature f^* with $H(f^*(X)|Y) \leq \beta < 1$, where Y takes values from $\{\pm 1\}$. Let $\mathcal{G}$ be a set of $\{\pm 1\}$-valued functions defined on $\{f^*(x) : x \in \mathcal{X}\}$. With probability at least $1 - \delta$, every function g in $\mathcal{G}$ satisfies*

$$P(Y \neq g(X)) \leq \hat{P}_n(Y \neq g(X)) + \frac{\beta}{2} + \frac{2}{n} + \sqrt{\frac{ln(1/\delta)}{2n}}$$

Proof. Apply Lemma 1 to Theorem 1. □

Corollary 1. *Given discriminative feature f^* with $H(Y|f^*(X)) \leq \alpha$ and $H(f^*(X)|Y) \leq \beta < 1$, where Y takes values from $\{\pm 1\}$. Given a sufficiently rich space of hypothesis space $\mathcal{G}$, the risk bound will be minimized if both α and β are minimized.*

Proof. From the risk bound in Theorem 2, the first term on the right hand side, which is the empirical risk, can be minimized (given the assumption that $\mathcal{G}$ is sufficiently rich) by minimizing $H(Y|f^*(X))$, or equivalently, by minimizing α. The second term on the right hand side can be minimized by minimizing β. □

We note that the bounds in this section are only meaningful when $H(f^*(X)|Y) < 1$. In practice, this cannot be easily achieved while preserving a near zero $H(Y|f^*(X))$, and this suggests that the interaction between domain knowledge and data should play an important role in the construction of features that may eventually approach this ideal quality. This motivates our algorithm.

References

[Almuallim and Dietterich, 1991] Almuallim, H., Dietterich, T.G. Learning with many irrelevant features, in Ninth National Conference on AI, MIT Press, pp.547-552, 1991.

[Bartlett and Mendelson, 2002] Bartlett, P.L., Mendelson, S. Rademacher and Gaussian complexities: Risk bounds and structural results. *J. of Machine Learning Research*, 3:463-482, 2002.

[Battiti, 1994] Battiti, R. Using mutual information for selecting features in supervised neural net learning. *Neural Networks*, 5(4):537-550, 1994.

[Blum and Langley, 1997] Blum, A., Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245-271, 1997.

[Ding, 2005] Ding, X., Chinese Character Recognition, in *Handbook of Pattern Recognition and Computer Vision*, 3rd ed., pp. 241-257, World Scientific, 2005.

[Felzenszwalb and Huttenlocher, 2000] Felzenszwalb, P., Huttenlocher, D. Efficient Matching of Pictorial Structures, CVPR 2000, pp. 66-73, 2000.

[Feder and Merhav, 1994] Feder, M., Merhav, N. Relations between entropy and error probability. *IEEE Trans. on Information Theory*, 40:259-266, 1994.

[Forsyth and Ponce, 2002] Forsyth, D.A., Ponce, J. *Computer Vision: A Modern Approach*, Prentic Hall, 2002.

[Gabrilovich and Markovitch, 2005] Gabrilovich, E., Markovitch, S. Feature Generation for Text Categorization Using World Knowledge, IJCAI 2005, pp. 1048-1053, 2005.

[Guyon et al., 2006] Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. *Feature Extraction, oundations and Applications*. Springer 2006.

[Kimura et al., 1997] Kimura, F., Wakabayashi, T., Tsuruoka, S., Miyake, Y. Improvement of Handwritten Japanese Character Recognition using Weighted Direction Code Histogram, *Pattern Recognition*, Vol. 30, No. 8, pp. 1329-1337, 1997.

[Kira and Rendell, 1992] Kira, K, Rendell, L.A. A practical approach to feature selection, in Proc. of the 9th ICML, 1992.

[Kohavi and John, 1997] Kohavi, R., John, G. Wrappers for feature selection. *Artificial Intelligence*, 97(1-2):273-324, December 1997.

[Lu et al., 1991] Lu, S.W., Ren, Y., Suen, C.Y. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, Vol. 24, No. 7, pp. 617-632, 1991.

[Markovitch and Rosenstein, 2002] Markovitch, S., Rosenstein, D. Feature Generation Using General Constructor Functions, *Machine Learning*, Vol. 49, No. 1, pp. 59-98, 2002.

[Suen et al., 2003] Suen, C.Y., Mori, S., Kim, S.H., Leung, C.H. Analysis and Recognition of Asian Scripts - the State of the Art, ICDAR '03, pp. 866-878, 2003.

[Sun and DeJong, 2005] Sun, Q., DeJong, G. Feature Kernel Functions: Improving SVMs Using High-Level Knowledge. CVPR (2) 2005: 177-183, 2005.

[Tishby et al., 1999] Tishby, N., Pereira, F., Bialek, W. The information bottleneck method. In *Proc. of the 37th Ann. Allerton Conf. on Communication, Control and Computing*, pp. 368-377, 1999.

[Torkkola, 2003] Torkkola, K. Feature extraction by non-parametric mutual information maximization. *J. of Machine Learning Research*, 3:1415-1438, 2003.