Selecting Shape Features Using Multi-class Relevance Vector Machine

Hao Zhang
Jitendra Malik

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2005-6
http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-6.html

October 10, 2005

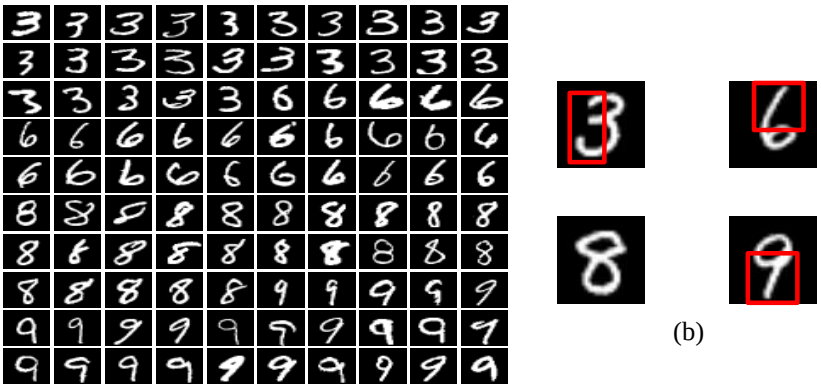
Selecting Shape Features Using Multi-class Relevance Vector Machine

Hao Zhang Jitendra Malik
Computer Science Division, EECS Dept
UC Berkeley, CA 94720

Abstract

The task of visual object recognition benefits from feature selection as it reduces the amount of computation in recognizing a new instance of an object, and the selected features give insights into the classification process. We focus on a class of current feature selection methods known as embedded methods: due to the nature of multi-way classification in object recognition, we derive an extension of the Relevance Vector Machine technique to multi-class. In experiments, we apply Relevance Vector Machine on the problem of digit classification and study its effects. Experimental results show that our classifier enhances accuracy, yields good interpretation for the selected subset of features and costs only a constant factor of the baseline classifier.

1. Introduction



(a) (b)

Figure 1: (a) Collection of digits (b) Prototype digits with highlighted discriminative part.

When looking at a slate of digits (fig. 1(a)), after compensating for the variation in writing style, one can see that there are distinctive parts of the shape which best tell apart each digit against other classes (fig. 1(b)). Being able to identify these discriminative parts is a remarkable ability for classifying the digits quickly and accurately. The aim of this work is to find those parts automatically using feature selection techniques.

Even though we illustrate it with digits, this problem is representative and is important to general object recognition because:

1. **Importance of shape cue** Studies in biological vision have suggested that the shape cues account for the majority of the necessary information for visual object recognition (among other cues such as texture and color) [1]. Therefore, it is worthwhile to study how to differentiate among shape cues and to derive methods to pick out the most useful shape cues from the data. Handwritten digits have natural variations in shape, thus provides a good setting for studying selection of shape cues.

2. **Fine discrimination** In object recognition, a common challenge is to classify visually similar objects. This demands a vision system to emphasize the more discriminative parts of the instances rather than comparing them on the whole. Digits

provide a good source of this problem because there are "borderline" instances (e.g. among 8s and 9s) that need detailed discrimination.

With currently available techniques, we choose to tackle the problem by phrasing it in terms of feature selection. Given the popularity of using feature selection techniques on genomic data, it is important to note that our aim is different than theirs: the ground truth in genomics is that only a few features (genes) are relevant for the property under study, but in shape classification, in fact, all features are relevant and useful for classification. In genomics, feature selection has the potential to yield a near-perfect classifier. But in shape classification, we do not expect a classifier built on feature selection to perform better than those that use all features. Our aim is to limit ourselves with the number of features and see what are the best ones to use in a classifier.

Following the survey of [2], feature selection approaches fall into three main categories: filter, wrapper and embedded methods. We describe those categories with examples from computer vision literature where possible:

1. Filter methods select the features independent of the classifier, and are a preprocessing step to prune features. Example in computer vision: In [3], scale-invariant image features are extracted and ranked by a likelihood or mutual information criterion. The most relevant features are then used in a standard classifier. However, we find the separation of the selection stage and the classification stage unsatisfactory.

2. Wrapper methods treat the classifier as a black box and provide it with varying subsets of features so as to optimize for the best feature set. In practice, wrapper methods often search exhaustively on all possible subsets which is exponentially slow or use greedy search which only explores a portion of all subsets.

3. Embedded methods, as an improvement on wrapper methods, incorporate feature selection in the training of the classifier. Example in computer vision: In [4], a pair-wise affinity matrix is computed on the training set and an optimization on the Laplacian spectrum of the affinity matrix is carried out. However, The construction of affinity matrix and its subsequent learning would be inefficient for moderate size training set (e.g. a few hundred examples).

In this paper, we focus on a new class of embedded methods for feature selection ([5] and [6]). Compared to other methods, they are distinctive in the sense that they integrate feature selection into the training process in an efficient manner:

1. Roughly speaking they optimize the training error as a sum of loss on data plus *a regularization penalty on the weights that promotes sparseness*. They optimize a clear error criterion that is tuned for the task of classification (in contrast, filter methods is not tied with the classifier).

2. Without the regularization term (or with a slight change of the term), they reduce to well-known classifiers such as SVM and logistic regression. Therefore, from a practical point of view they can be thought of as "improved" (or alternative) version of the baseline classifier.

3. Algorithmically, they cost only a constant factor of the time of the baseline classifier. ([5] costs about the same as a usual SVM, [6] costs about 10 times of a logistic regression, since it usually converges in 10 iterations).

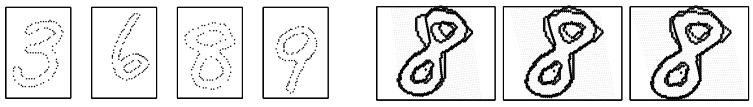
The theoretical part of this work is to extend [6] to multi-class setting because in shape discrimination the task almost always involves more than two classes. On the empirical side, we study the our classifier in both two-class(in comparison to [5]) and multi-class problems.

In light of the "vision for learning" motivation, we believe that shape cues provide an excellent data set to gain insights into a particular feature selection technique. The shape features selected by a technique can be visualized and evaluated against intuition, thus providing insights into the classification process. This is an advantage that other data sets (e.g. genomics) do not have.

The paper is organized as follows: Section 2 discusses the shape features extracted from the image. Section 3 introduces two embedded methods: (1) 1-norm Support Vector Machine (1-norm SVM) and (2) Relevance Vector Machine (RVM). There we develop a multi-class extension to RVM. Section 4 studies the experimental results on digits and we conclude in section 5.

2 Shape Features

The most straightforward experiment on feature selection takes the raw image as the feature vector. This leaves the learning machinery with the job of modelling shape variation which is often hard. Moreover, each pixel location is treated as a feature which may actually correspond to different parts on the shape in different images. In order to obtain features in correspondence, we choose to use the shape context descriptor obtained from aligning the images of the digits, following that



(a) (b)

Figure 2: (a) Anchor shapes for four digit classes (b) Visualization of the process of finding correspondences

of [7]. We choose this set of features over other image-based features (e.g. intensity) to focus on the shape aspect of the image.

From each class, we select an "anchor" digit that all other digits of that class are matched to. This digit is picked as the median in terms of shape context distance (so that the maximum of its distance to all digits of that class is minimum). Fig 2(a) shows the anchor shapes. When each digit is matched to the anchor, the algorithm establishes correspondence between the points on the anchor and the points on the digit (the process is illustrated in fig 2(b)). This enables us to define a "shape feature" as the point on each digit that corresponds to a particular point on the anchor shape.

At each "shape feature" location, we extract two numbers: the shape context distance to the corresponding shape context on the anchor digit, and the sum of squared differences to the corresponding image patch on the anchor digit. We use distances instead of original values to cut down dimensionality, so that there are enough training examples for the feature selection stage.

We have 100 "shape feature" locations on each shape and therefore we obtain 200 numbers from each digit. When studying the importance of a single feature, we add the weight on its shape context distance and its image patch distance.

3 Feature Selection: Embedded Methods

We study embedded methods that optimize the following objective function:

$$\frac{1}{n}\sum_{i=1}^{n} f_{\text{loss}}(w^T x_i) + \lambda f_{\text{penalty}}(w) \tag{1}$$

where x_i is the i'th data point, w is the weight vector, and λ is a relative weighting of the two terms.

The first term is the training error: an average of loss values at the data points. The second term is a regularizing penalty on the weights. By choices of the loss function (f_{loss}) and the penalty function (f_{penalty}), this general model reduces to well-known cases such as linear regression (f_{loss} being square loss and f_{penalty} being 0), logistic regression (f_{loss} being logistic loss and f_{penalty} being 0) or SVM (f_{loss} being hinge loss and f_{penalty} being $w^T w$).

It is also well-known that certain types of f_{penalty} promote sparseness in the weights. An example is the Lasso regression [8] which is similar to linear regression but puts an L_1 penalty on the weights. It pushes many components of the weight vector w to zero, effectively selecting features.

To understand the reason why those types of f_{penalty} promote sparseness, we illustrate in the special case of linear regression in two dimensions (where $w = (w_1, w_2)$ and $f_{\text{loss}} = (w_1 x_1 + w_2 x_2)^2$). First, note that the optimization problem (1) is equivalent as:

$$\frac{1}{n}\sum_{i=1}^{n} f_{\text{loss}}(w^T x_i) \qquad \text{s.t.} \quad f_{\text{penalty}}(w) = C \tag{2}$$

where C is a constant determined by λ. Then, the optimization problem can be thought of as finding the point on the penalty contour ($f_{\text{penalty}}(w) = C$) such that the it touches the smallest training error contour ($\sum_{i=1}^{n} f_{\text{loss}}(w^T x_i) = C'$). Note that in the linear regression case, the training error contour is always an ellipse.

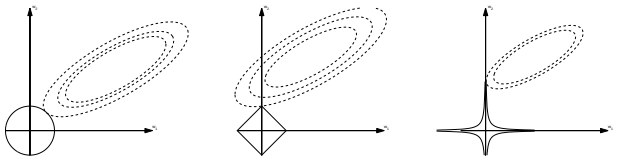
We consider three different types of f_{penalty}:

1.$f_{\text{penalty}} = w^T w$ (L_2^2 norm): also known as ridge regression, the penalty contour is a circle. The point on the penalty contour that minimizes the training error is where the two contours are tangential. Usually, at that point none of the weights are zero. (fig. 3(a))

2. $f_{\text{penalty}} = |w|$(L_1 norm): special case of lasso regression, the penalty contour is a diamond. Because the vertices of the diamond tend to "stick out", they are usually the spot on the penalty contour where the smallest training error contour is attained. (fig. 3(b)) At those vertices, one of the w_1, w_2 is pushed to zero.

3. $f_{\text{penalty}} = (\sum_k |w_k|^\epsilon)^{1/\epsilon}$($L_\epsilon$ norm): the penalty contour is concave and the vertices of the contour "sticks out" much more than the L_1 case, therefore it's more likely that one of the vertices minimizes the training error. (fig. 3(c))

This phenomenon is general for other types of f_{loss} as well: by choosing those types of f_{penalty}, we obtain a classifier which has feature selection capability. However, not all of the choice has a tractable solution. We study two particular variants that admits efficient computation: 1-norm SVM and multi-class Relevance Vector Machine.



(a) (b) (c)

Figure 3: The penalty contour (in solid line) and the training error ellipses (in dotted line) (a) L_2^2 norm (b) L_1 norm (c) L_ϵ norm

3.1 1-norm SVM

Here the loss function f_{loss} is the SVM hinge loss and the f_{penalty} is L_1-norm. In the optimal solution, the number of non-zero weights depends on the value of the relative weighting λ in Eq. 1: a larger λ drives more weights to zero. [5] proposes a solution to Eq. 1 in which each weight, as a function of λ, is computed in an incremental way as more weights are activated. Each weight w follows a piecewise linear path, and the computational complexity of all the paths is just slightly more than the baseline SVM classifier. In our experiments, we visualize the weights as they are being incrementally activated by the 1-norm SVM.

3.2 Multi-class Relevance Vector Machine

In the multi-class setting, the SVM loss does not extend naturally. For this reason, we turn to an alternative classification technique: multinomial logistic regression, which gives rise to the Relevance Vector Machine technique introduced in [6].

In this case, the f_{loss} is the multinomial logistic loss, and the f_{penalty} is the negative log of the student-t distribution on w, and the weighting λ is 1. The penalty contours, which are plotted in fig. 4(a), has the desirable property of being highly concave. (In practice, we use a limiting prior whose penalty contour is even more concave, see fig. 4(b).)

Also, in this case, the optimization problem Eq. 1 is equivalent(by taking the negative of the log of probability) to the problem of a maximum a posterior (MAP) estimate given the student-t prior and the multinomial logit likelihood . Therefore it can be casted as a estimation problem on the following hierarchical model, as shown in fig. 4(c): a hyper-parameter α is introduced for each weight (given α, w is distributed as a zero-mean, variance $1/\alpha$ Gaussian), and the α itself has a Gamma prior. (When the α is integrated out, the marginal distribution of w is a student-t density, as in the original setup.) The α parameter for each w is intuitively called the "relevance" of that feature, in the sense that the bigger the α, the more likely the feature weight w is driven to zero.

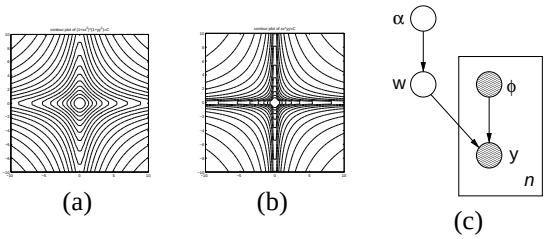
This additional layer of hyper-parameter yields an optimization process in two stages, in a fashion similar to Expectation-Maximization: (1) optimizing over w with fixed α, (2) optimize over α by using the optimal w from (1). This is the main iteration in the RVM technique. In our derivations below, we call them the "inner loop" and "outer loop". The details of the derivation and a discussion is in the appendix.

At the end of the iteration, we obtain a set of converged α's and w's. Typically, many of the w's are zero. However, even for those w's that are not zero, their associated α's vary much, which suggests the method of ranking the features by the α's and threshold at successive levels to select different size subsets of features. (Note that the ranking is not independent for each feature because the α's are obtained by considering all features.) This way of studying the effect of feature selection makes it comparable to the successive larger subsets of features in the 1-norm SVM case.

The original RVM is derived and experimented on two-class problems. While mentioning an extension to multi-class, the original formulation essentially treats the multi-class problems as a series of n one-vs-rest binary classification problems. This would translate into training n binary classifiers independently.[1] Instead, to fully exploit the potential of this technique,

[1] In [6] eqn. 28, the multiclass likelihood is defined as $P(t \mid w) = \prod_n \prod_k \sigma(y_k(x_n; w_k))^{t_{nk}}$, where x and w are input variables and weights,

we derive a multi-class RVM based on the first principles of the multinomial logistic regression and the prior on w, in the hierarchical model. (in appendix)



(a) (b) (c)

Figure 4: (a) The equal penalty contour for student-t prior on w when $a = b > 0$. (b) The equal penalty contour for Jeffrey's prior (density $\frac{1}{|x|}$) on w when $a = b = 0$. (c) Graphical model for the RVM. The response y is a multinomial logit function on the input data ϕ with the weights w's. Each weight w is associated a hyper-parameter α and $p(w \mid \alpha) = N(w \mid 0, \alpha^{-1})$. Each α is distributed as Gamma(a, b).

4 Experimental results

4.1 Setup

We experiment on digits from the MNIST database [9]. To extract features, each query digit is aligned against a collection of prototype digits, one for each class, resulting in a total number of $200C$ "shape context" features. By restricting our attention to only one prototype per class, we use an overly simple classifier so that we can study the feature selection problem in isolation. Here, we do not address the issue of feature selection across more prototypes per class but believe that our findings are indicative of the general case.

In our derivation of RVM, the features are class-specific (i.e. the $\phi_i^{(p)}$ in section A.1), which agrees with the process of getting features by matching against each class. In the 1-norm SVM setting we simply concatenate the features from all classes.

4.2 Two-class

We first study the effects in the two-class problem: 8s vs 9s. They are a easily confused pair and indeed yields worse error rate than a classifier trained on most other pairs of digits. For this problem, we run both RVM and 1-norm SVM on the shape context features. An overview plot of the error rate is in fig. 5(a), computed by 5-fold cross validation. The eventual error rate is not in the range of the state of the art(0.6% as in [7]), but is well-performing for a single unit in a prototype-based method, that will perform much better with additional units of other prototypes [10]. The error rate also drops noticeably with a few activated features, suggesting that a small portion of the features account for the overall classification. We look more closely as to what features are activated in this range, shown in fig. 6.

As a baseline, we also include results from the simple filter method of ranking features by its mutual information with the class label, shown in fig. 6(a). A few things can be noticed:

1. Since the mutual information criterion picks features independently, it selects features from the lower half of the digit 9 repeatedly. Those feature each has a high predictive power but as a collection, they are highly correlated with each other. In contrast, RVM or 1-norm SVM quickly exhausts the features there and moves onto other parts of the shape.

2. The two embedded methods, RVM and 1-norm SVM, agree fairly well on the features they selected, and agree well on their weights. One difference is that RVM also tend to pick out another spot of discrimination, the upper right corner of the digit 9 (which is useful for telling against 8s that are not closed in that region). In comparison, 1-norm SVM took longer to start assigning significant weights to that region.

---

respectively; t_{nk} is the indicator variable for observation n to be in class k, y_k is the predictor for class k, and $\sigma(y)$ is the logit function $1/(1 + e^{-y})$. The product of binary logit functions treats the class indicator variable y_k independently for each class. In contrast, a true multiclass likelihood is $P(t \mid w) = \prod_n \prod_k \sigma(y_k; y_1, y_2, ..., y_K)^{t_{nk}}$ where the predictors for each class y_k is coupled in the multinomial logit function (or the softmax): $\sigma(y_k; y_1, ..., y_K) = e^{y_k}/(e^{y_1} + ... + e^{y_K})$.

As an interesting side experiment, we also try to compare the shape context feature against the raw intensity values from the original MNIST image, using our feature selection methods. We first run the 1-norm SVM on the two separate feature sets. Fig. 7(a) shows that shape context outperforms intensity when a few features are activated, while their eventual performance are comparable. This suggests that shape contexts are the feature of choice when allowed only a limited number of features, which is confirmed in fig. 7(b): a run of 1-norm SVM on the combined pool of features selects mostly the shape contexts first. This effect is also present in the RVM, though the contrast between the two types of feature is less.

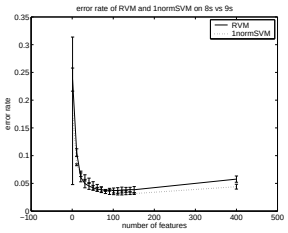


Figure 5: Error rate on two-class problem as more features are added

4.3 Four-class

We pick classes 3,6,8,9 for a similar reason as in the two-class case: They have visually similar shapes and some parts of their shapes we can identify as being discriminative in this four-class classification task. We are interested in seeing whether this notion is reflected in the results from the our method.

Since the problem is multi-class, 1-norm SVM is not applicable and we run only RVM, with mutual information as a baseline. Fig. 8(a) plots the error rate as a function of the number of features. Similar to the two-class problem, peak performance is reached with a similar percentage of features. The peak error rate is only slightly worse than that of the two-class case (6% from 4%), validating the multi-class extension.

In the well-performing range of number of features, we visualize the magnitude of the weights on each feature(fig. 8). Similar to the two-class case, we see that the features tend to spread out in RVM as it avoids selecting correlated features.

It is interesting to reflect on the features selected in fig. 8(b) to notice that those are the features good for telling the digit class apart from the rest of the classes: the upper left portion of 3, the tip of the stem of 6, the lower circle of 8(which has a different topology than the other classes), and the lower portion of 9 can be examined alone to decide which class it is. If we then look at the magnitude of the weights on those features, for example, the bigger weights on the lower portion of the 9 suggests that the compared to other cues, the multinomial logistic classifier can benefit the most from comparing the query digit to the prototype 9 and examine the lower portion of the matching score. We think it is important and interesting to interpret the learning mechanism in this way.

5 Conclusion

In this work, we have demonstrated how domain knowledge from shape analysis can be used to extract a good initial set of features suitable for selection algorithms. We extended one of the embedded feature selection methods to handle multi-class

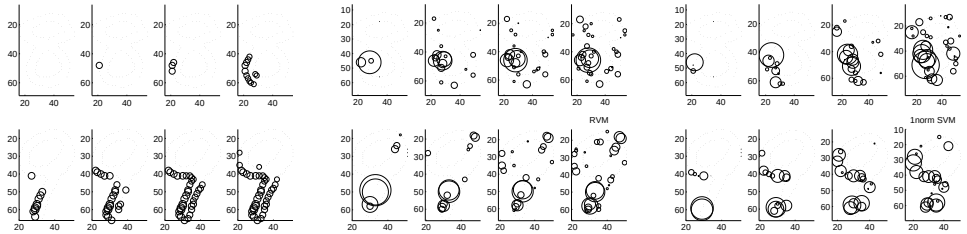
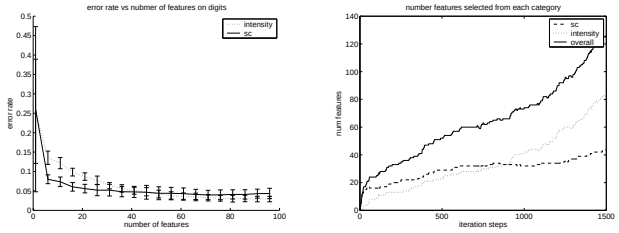


Figure 6: Feature weights as more features are activated: (a)mutual information (b)RVM (c)1-norm SVM



(a) (b)

Figure 7: Comparing shape context and intensity features by 1-norm SVM: (a) error rate as a function of number of activated features (b) number of features from each category

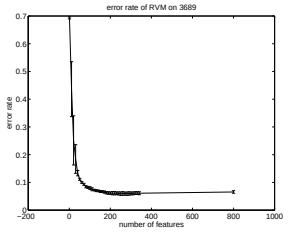
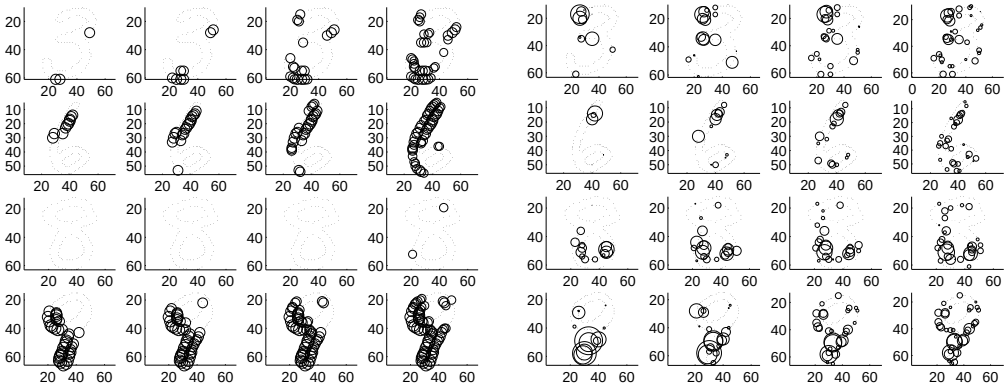


Figure 8: Error rate of RVM on four-class problem as more features are added



(a) (b)

Figure 9: Feature weights as more features are activated: (a)mutual information (b)RVM

problems , studied the performance of two variants of the embedded method, and showed that the selected features have an intuitive interpretation for the classification task.

Acknowledgments

We thank Matthias Seeger and Ji Zhu for fruitful discussions, and Li Wang for providing the 1-norm SVM code.

References

[1] Stephen E. Palmer. *Vision Science – Photons to Phenomenology*. The MIT Press, 1999.

[2] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[3] Gy. Dorkó and C. Schmid. Selection of scale invariant neighborhoods for object class recognition. In *Proceedings of the 9th International Conference on Computer Vision,*, pages 634–640, 2003.

[4] Lior Wolf and Amnon Shashua. Feature selection for unsupervised and supervised inference: the emergence of sparsity in a weighted-based approach. In *Proceedings of the 9th International Conference on Computer Vision*, pages 378–384, 2003.

[5] Ji Zhu, Saharon Rosset, Trevor Hastie, and Rob Tibshirani. 1-norm support vector machines. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[6] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, 2001.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[8] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58:267–288, 1996.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[10] Hao Zhang and Jitendra Malik. Learning a discriminative classifier using shape context distances. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 242–247, 2003.

[11] David J. C. MacKay. Bayesian interpolation. *Neural Comput.*, 4(3):415–447, 1992.

A Appendix

A.1 Inner loop: L_2 regularized logistic regression

This is where α's are fixed. Suppose the number of input data to be n, the number of classes to be C. The feature vector components are "class-specific", i.e., an input is mapped into a series of feature vectors, one corresponding to each class p: $\phi_i^{(p)}$. Following multinomial logistic modelling: $u_i^{(p)} = \langle w^{(p)}, \phi_i^{(p)} \rangle$. The output is a softmax over the u's: $\mu_i^{(p)} = \frac{e^{u_i^{(p)}}}{e^{u_i^{(1)}} + \ldots + e^{u_i^{(C)}}}$.

Suppose each $w_j^{(p)}$ is regularized by hyper parameter $\alpha_j^{(p)}$, i.e., $w_j^{(p)} \sim N(0, 1/\alpha_j^{(p)})$. Under the maximum a posterior (MAP) principle, we minimize the negative log of the posterior:

$$
\begin{aligned}
&- \log p(w \mid y) \doteq - \log p(y \mid w) - \log p(w \mid \alpha) := \Psi(w) \\
&\doteq - \sum_{i,p} y_i^{(p)} \log \mu_i^{(p)} + \sum_{j,p} \frac{1}{2} (\alpha_j^{(p)} (w_j^{(p)})^2 - \log \alpha_j^{(p)})
\end{aligned}
$$

where $\doteq$ denotes equality modulo constants(the first $\doteq$ is modulo $\log p(y \mid \alpha)$, a term which will become important in the outer loop but here is a constant since α is fixed), and $y_i^{(p)}$ is the binary indicator (i.e. $y_i^{(p)} = 1$ iff data i is of class p).

For a more intuitive derivation, we adopt a more succinct notation: let $\phi^{(p)}$ be the design matrix for class p, namely, $\phi^{(p)} = (\phi_1^{(p)} ... \phi_n^{(p)})^T$. And let $\phi = \text{diag}(\phi^{(p)})_{p=1..C}$. Let w be the concatenation of all the weights $w_j^{(p)}$ (in j-major order). Let w_k denotes its k'th component. Similarly, let α and α_k be that of the $\alpha_j^{(p)}$. Let K be the total number of features from all classes. Let $A = \text{diag}(\alpha_k)_{k=1..K}$. Let $B_{p,q} = \text{diag}(\mu_i^{(p)}(\delta_p^q - \mu_i^{(q)}))_{i=1..n}$ and B be a block matrix consisting of $B_{p,q}$. Then the derivatives of $\Psi(w)$ can be written in matrix form as:

$$
\begin{aligned}
\frac{\partial \Psi}{\partial \mathbf{w}} &= \boldsymbol{\phi}^T (\boldsymbol{\mu} - \mathbf{y}) + A\mathbf{w} \\
H(\mathbf{w}) := \frac{\partial^2 \Psi}{\partial \mathbf{w} \partial \mathbf{w}} &= \boldsymbol{\phi}^T B \boldsymbol{\phi} + A
\end{aligned}
$$

These first and second derivatives are used in the iterative reweighed least squares(IRLS) procedure, as in the ordinary logistic regression, until convergence.

A.2 Outer loop: MAP estimate of α

We want to minimize the negative log posterior for α: $f^*(\alpha) = - \log p(y \mid \alpha) - \log p(\alpha)$. To obtain $p(y \mid \alpha)$, we know by conditional probability definition that $p(y \mid \alpha) = \frac{p(y, \tilde{w} \mid \alpha)}{p(\tilde{w} \mid y, \alpha)}$. Take negative log of both sides, recall the definition of $\Psi(w)$ and it gives: $- \log p(y \mid \alpha) = \Psi(\tilde{w}) + \log p(\tilde{w} \mid y, \alpha)$. As justified in [6], assume saddle point approximation for $p(w \mid y, \alpha)$, i.e., $p(w \mid y, \alpha) \approx N(w \mid \tilde{w}, H(\tilde{w})^{-1})$. Then, $\log p(\tilde{w} \mid y, \alpha) \approx \log N(\tilde{w} \mid \tilde{w}, H(\tilde{w})^{-1}) = \frac{1}{2} \log \det H(\tilde{w}) - \frac{K}{2} \log(2\pi)$. Therefore the overall negative log posterior, dropping the constant of $-\frac{K}{2} \log(2\pi)$, is

$$
f(\alpha) \doteq \frac{1}{2} \log \det H(\tilde{w}) + \Psi(\tilde{w}) - \log p(\alpha)
$$

The α's tend to grow very large during estimation hence we will optimize w.r.t. to $\log(\alpha)$. (This reparametrization affects $p(\alpha)$ slightly via a change of variable.)

The derivative of $f(\alpha)$ has three parts: The first term is computed based on the matrix calculus result that $\frac{d \log\det X}{d X} = X^{-T}$ and the chain rule that $\frac{d\ f(X)}{d\,\alpha} = \text{tr}\left(\frac{\partial f}{\partial X}\frac{\partial X}{\partial \alpha}\right)$.

$$\begin{aligned}\frac{1}{2}\frac{\partial \log\det H(\tilde{w})}{\partial \alpha_k} &= \frac{1}{2}\text{tr}\left(\left(H(\tilde{w})\right)^{-1}\frac{\partial H(\tilde{w})}{\partial \alpha_k}\right) \\ &= \frac{1}{2}\left[\left(H(\tilde{w})\right)^{-1}\right]_{kk} := \frac{1}{2}\Sigma_k\end{aligned}$$

Here we have assumed that B is constant w.r.t. α. An exact derivative without assuming a constant B can be obtained which is more complicated. However, during our experiments, it produces negligible difference in the converged answer.

The second term $\Psi(\tilde{w})$ depends on α through two ways: in a direct way through the terms involving the prior on w and indirectly through the optimal $\tilde{w}$ which depends on the value of α. However, we exploit the fact that $\tilde{w}$ is optimal so that the second part has derivative zero:

$$\begin{aligned}\frac{\partial \Psi(\tilde{w})}{\partial \alpha_k} &= \frac{\partial \Psi(\tilde{w})}{\partial \alpha_k}\mid_{\text{fixed } \tilde{w}} + \frac{\partial \Psi(\tilde{w})}{\partial \tilde{w}}\mid_{\text{fixed } \alpha}\frac{\partial \tilde{w}}{\partial \alpha_k} \\ &= \frac{\partial \Psi(\tilde{w})}{\partial \alpha_k}\mid_{\text{fixed } \tilde{w}} + 0 \cdot \frac{\partial \tilde{w}}{\partial \alpha_k} \\ &= \frac{1}{2}(\tilde{w}_k^2 - \frac{1}{\alpha_k})\end{aligned}$$

The third term is simply the negative log of the Gamma prior:

$$\frac{\partial(-\log p(\alpha))}{\partial \alpha_k} = b - \frac{a}{\alpha_k}$$

Based on the derivative, we set them to zero to obtain a set of fixed-point iteration equations. This leads to the re-estimation rule for the α's, similar in form to [11]: define the "degree of well-determinedness parameter" γ_k to be $\gamma_k = 1 - \alpha_k\Sigma_k$, then the re-estimate update is:

$$\alpha_k' = \frac{\gamma_k + 2a}{\tilde{w}_k^2 + 2b}$$

A.3 Discussion of RVM

The choice of the values for a and b: When $a = b > 0$, the equivalent prior on w is a student-t distribution which approximates a Gaussian near the origin. This is undesirable as it puts an L_2 norm penalty on the weights when the weights become small. To avoid this, we set the parameters $a = b = 0$, which puts an improper, density $\frac{1}{|x|}$ prior that is independent of the scale of the weights and always has concave equal-penalty contours.

The algorithm is fast: In implementation, the inner loop returns along with w and b the inverse of the Hessian at the optimum (as is needed in logistic regression anyway). The simple updates on α virtually don't cost any time. In experiments, we see that the reestimate converges quickly. Those α's that correspond to suppressed features tend to grow very large after a few iterations. As in [6], we prune those features which also speeds up later iterations.