The Logistic Random Field – A Convenient Graphical Model for Learning Parameters for MRF-based Labeling

Marshall F. Tappen, Kegan G. G. Samuel, Craig V. Dean, David M. Lyle
University of Central Florida
School of Electrical Engineering and Computer Science Orlando, FL
{mtappen, kegan, cdean, dlyle}@eecs.ucf.edu

Abstract

Graphical models are fundamental tools for modeling images and other applications. In this paper, we propose the Logistic Random Field (LRF) model for representing a discrete-valued graphical model. The LRF model is based on an underlying quadratic model and a logistic function. The chief advantages of the LRF are its convenience and flexibility. The quadratic model makes inference easy to implement using standard numerical linear algebra routines. This quadratic model also allows the log-likelihood of the training data to be differentiated with respect to any parameter in the model, enhancing the flexibility of the LRF model. To demonstrate the usefulness of this model we use it to learn how to segment objects, specifically roads, horses, and cows. In addition, we demonstrate the flexibility of the LRF model by incorporating super-pixels. We then show that the LRF segmentation model produces segmentations that are competitive with recently published results.

1. Introduction

Low-level vision tasks can be roughly divided into estimation problems and labeling problems. In estimation problems, such as denoising [16, 18] or lightness estimation [20], the goal is usually to predict pixel values. In labeling tasks, which are the focus of this paper, the goal is to assign the correct label, such as segment or discrete depth value, to each pixel. Examples of labeling tasks include segmentation [11] and stereo disparity [4] . Markov Random Field (MRF) models have proven to be especially useful for low-level tasks because they enable local clique potential functions to describe distributions over large images. A number of useful behaviors, such as smoothing or propagating information into locally ambiguous image areas, can be implemented by setting the clique potentials correctly [22].

Two key problems encountered in the use of MRF models are inference and estimating the model parameters. The inference task can be viewed as either finding the solution with the highest probability, often referred to as the maximum a posteriori (MAP) solution, or the task of estimating the marginal probability distribution of every variable in the MRF. The fundamental difficulty in working with MRF models is the fact that when the graph representation of the MRF has loops, both types of inference are, in all but a few cases, NP-complete and intractable for the size of problems that are typically encountered in low-level vision[4, 25].

This practical intractability has large ramifications on the problem of estimating MRF model parameters. Maximum-likelihood learning in MRF models requires the ability to estimate the marginal probability distributions of the node cliques in the model. Other non-probabilistic methods, such as [5] or [24], require the ability to find the MAP solution.

In response to these intractability issues, much progress has been made on approximate inference algorithms. Today, a number of powerful algorithms are available that perform well [4, 26, 10] on the type of discrete-valued MRF . Using these algorithms, it is possible to compute the necessary quantities to learn parameters. Roughly speaking, learning with approximate inference algorithms can be viewed as first choosing a particular MRF model, then finding an approximate inference algorithm that can be used to perform the computations necessary to learn the model parameters.

In this paper, we propose an alternative to this approach. As mentioned above, in certain cases *exact* inference in MRF models is computationally feasible. Rather than finding an inference algorithm to match a model, we propose beginning with an MRF model where inference is exact, fast, and convenient, such as a Gaussian MRF (GMRF), then adapting it to perform labeling. As we will show, this leads to a model that is:

- Easy to train

- Computationally efficient

- Flexible

1

- Comparable in performance with more traditional MRF models for labeling pixels.

Below, Section 2 gives an overview of our approach, which we refer to as the Logistic Random Field (LRF). To demonstrate the effectiveness of the LRF model, we apply it to three segmentation tasks: segmenting roads, horses, and cows. The road experiments in Section 2.3 demonstrate the use of the LRF in a real-world vision application; specifically road segmentation for use by autonomous land vehicles in an urban environment. The horse and cow experiments in Section 5.3 and 5.4 show that, in addition to the usability advantages inherent in the LRF model, the LRF model compares favorably with existing MRF models.

2. The Logistic Random Field Model

To understand the Logistic Random Field model, we will first briefly review the logistic regression model for classification, then describe a practical LRF configuration for finding pixels belonging to roads in Section 2.3. Following this, we show how the underlying ideas can be generalized to a random field model and describe the actual LRF model in Sections 2.7 and 2.8.

2.1. Background: Logistic Regression

In a classic logistic regression model, the probability that a given sample, with feature vector ϕ, should be labeled $+1$ is:

$$p(+1|x) = \sigma(\mathbf{w}^T\phi). \qquad (1)$$

The vector $\mathbf{w}$ is a vector of weights that defines a line in the feature space. The function $\sigma(\cdot)$ is the logistic function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)} \qquad (2)$$

Logistic regression can be viewed as taking the linear function $\mathbf{w}^T\phi$, which ranges from $-\infty$ to $+\infty$, and converting it into a probability, which ranges from 0 to 1. The weight vector $\mathbf{w}$ can be found by directly optimizing the log-likelihood of a training set.

2.2. From Samples to Images

The same underlying ideas can be applied to labeling pixels in images. The probability of each pixel being labeled $+1$ could be calculated by applying $\sigma(\cdot)$ to each pixel in a continuous-valued response image $\mathbf{x}$.

The response image $\mathbf{x}$ could be computed in any manner, since we are particularly interested in MRF models for labeling the pixels, we will define the response image $\mathbf{x}$ to be the MAP solution of a continuous-valued MRF model, given the observations $\mathbf{y}$:

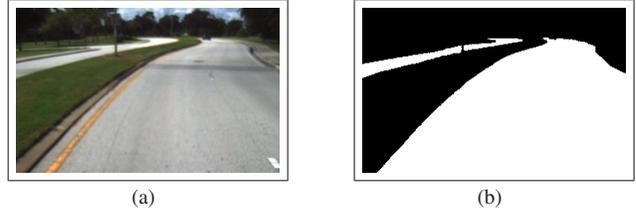$$\mathbf{x} \triangleq \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \qquad (3)$$

Figure 2. Road Example - image (a) shows a typical input image, image (b) shows a manual ground-truth segmentation.

Note that as $\mathbf{x}$ grows larger at a particular pixel, the probability of that pixel having label $+1$ goes to 1.

2.3. Practical Example: Configuring an LRF to Find Roads

Ideally, $\mathbf{x}$ should be generated from a model that is efficient and makes parameter learning relatively simple. For these reasons, we use the Gaussian Conditional Random Field (GCRF) model proposed in [21] to generate $\mathbf{x}$. In this section, we will describe an implementation example of configuring an LRF to solve the problem of correctly segmenting road pixels in an image. Figure 2 shows an example image of a typical scene and the desired segmentation.

Similar to standard MRF formulations, such as [6], the LRF model will have two major components. To show these components, we will construct a weighted least-squares system, $C(\mathbf{x}, \mathbf{y})$ that defines the LRF.

The first component in the LRF is based upon interactions between the observed image and the labeling. The weighted least-squares system based on these constraints is formulated as:

$$C(\mathbf{x}, \mathbf{y}) = \sum_i w(\mathbf{y}; \theta_1)(x_i - 10)^2 + \sum_i w(\mathbf{y}; \theta_2)(x_i + 10)^2 \qquad (4)$$

Each term x_i refers to the entry at pixel i in the response image $\mathbf{x}$.

These two types of terms pull each pixel to either -10 or $+10$. While the response image should technically vary between $-\infty$ and $+\infty$, setting a particular pixel to $+10$ makes it's probability of being equal to $1 - (4 \times 10^{-5})$. In practice, ranging the values between -10 and $+10$ is sufficient.

2.4. Weighting Functions in the LRF

The weighting functions $w(\mathbf{y}; \theta_1)$ and $w(\mathbf{y}; \theta_2)$ primarily determine the classification of a pixel. If $w(\mathbf{y}; \theta_1)$ grows large, relative to $w(\mathbf{y}; \theta_2)$, the pixels will tend toward the value of $+10$ and vice-versa. The vectors θ_1 and θ_2 contain the parameters that the weighting functions use to estimate the correct weight from the observation $\mathbf{y}$.

For the road segmenting task, we use a relatively simple function to generate the weights. From the training images,

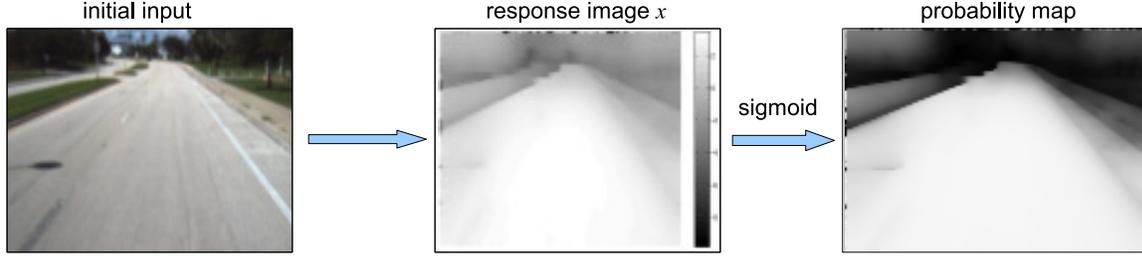initial input   response image x probability map

sigmoid

Figure 1. This figure shows the stages in the segmentation process. The leftmost image is the original image. The middle image shows the intensity of the pixel responses after being processed by the LRF system. The rightmost image shows the result after the responses are run through a sigmoid and thresholded.

we created 40,000 different 15×15 different patches for use in calculating the features for our LRF model. We then used k-means to greatly reduce the number of patches to 500 15×15 RGB cluster centers. The cluster centers returned by k-means are used as the features for the LRF.

Given a patch of the input image from location i, $\mathbf{y}_i$ the weight of the first term in Equation 4 is

$$w(\mathbf{y}_i; \theta_1) = b_1 + \sum_{q=1}^{500} \theta_1^q \exp\left(-||\mathbf{y}_i - \mu_q||^2\right) a \quad (5)$$

The term b_1 is a constant bias term. The vector μ_q is one of the 500 centers. Section 3 will discuss how the vector parameter θ_1^q can be learned from training data.

A similar weighting function is used for the second term in Equation 4:

$$w(\mathbf{y}_i; \theta_2) = b_2 + \sum_{q=1}^{500} \theta_2^q \exp\left(-||\mathbf{y}_i - \mu_q||^2\right) a \quad (6)$$

When the system is trained, the parameters, θ_1, and θ_2 will determine how these features are used to classify pixels.

2.5. Implementing Inter-Pixel Relationships

The power in MRF models lie in their ability to not only model how the input relates to pixel labels, but to also express relationships between pixels. In this case, we will incorporate a relationship similar to the Potts model, where nodes are encouraged to have the same value. This is done by adding the term:

$$\sum_{<i,j>} w(\mathbf{y}; \theta_3)(x_i - x_j)^2 \quad (7)$$

to $C(\mathbf{x}, \mathbf{y})$. This term, which is summed over all neighboring nodes i and j, has a smoothing effect that encourages neighboring nodes to have the same value. Just as above, this is also weighted by a weighting function.

The weighting function for neighboring constraints uses the same features described previously, in addition to horizontal and vertical gradient magnitudes at each pixel. As
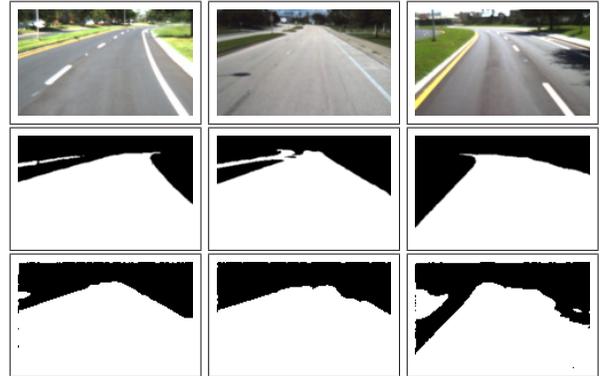

Figure 3. This figure shows some of the results when segmenting roads using the LRF system. Top: Original image. Middle: Ground-Truth segmentation mask. Bottom: Our segmentation mask.

in previous systems, incorporating the gradient magnitude prevents the system from smoothing across image boundaries [4, 14].

2.6. Results Segmenting Roads

With this basic model, we can train the weighting functions using the learning method described in Section 3. Figure 3 shows examples of an image taken from a vehicle, the ground-truth road segmentation, and the segmentation returned by our system. Overall the system performs well, correctly segmenting 89% of the road pixels. Using this dataset, we can also examine the benefit of incorporating the inter-pixel relationships into the LRF model. Figure 4 shows the ROC curves for models with and without inter-pixel interaction. The incorporation of inter-pixel interactions, which facilitate information being propagated across the image, improves the systems ability to correctly label pixels.
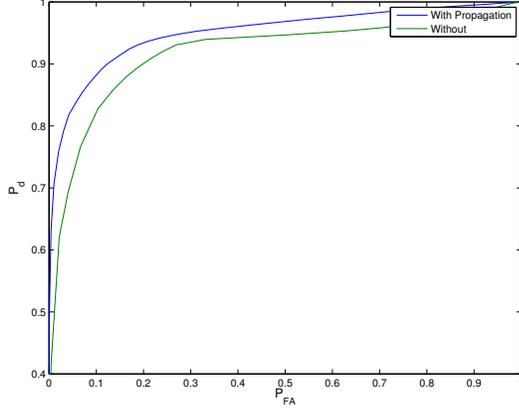
Figure 4. This figure shows the ROC curves for correctly detecting road pixels in a data. Curves are shown for both a model that uses only local evidence (shown in green) and a model that includes inter-pixel relationships, shown in blue. Including the inter-pixel relationships improves the performance of the system.

2.7. Formally Specifying Gaussian Conditional Random Fields

In this section we review the formal definition of the GCRF model. In the GCRF model, the negative log posterior distribution over $\mathbf{x}$, given an observation $\mathbf{y}$, is

$$- \log p(\mathbf{x}|\mathbf{y}) \triangleq C(\mathbf{x}, \mathbf{y}) - \log Z, \qquad (8)$$

where

$$C(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{N_f} \sum_{i,j} w_k(i,j; \mathbf{y}, \theta) \left((\mathbf{x} * f_k)(i,j) - r_k(i,j) \right)^2 \qquad (9)$$

The term $(\mathbf{x} * f_k)(i, j)$ denotes the value at location (i, j) in the image produced by convolving $\mathbf{x}$ with a linear filter f_k. We assume that this image only contains those pixels where the entire filter fits in $\mathbf{x}$. In other words, we neglect pixels where computing a convolution would require pixel values outside of the image $\mathbf{x}$.

The function $r_k(i, j)$ contains the estimated value of $(\mathbf{x} * f_k)(i, j)$ at each pixel. For each filter f_k, the function r_k uses the observed image $\mathbf{y}$ to estimate the value of the filter response at each pixel.

Each term in C is also associated with a positive weight $w_k(i, j; \mathbf{y}, \theta)$. Finally, the variable Z denotes the partition function or normalizing constant for the distribution.

The cost function $C(\mathbf{x}, \mathbf{y})$ can also be rewritten as in matrix form by creating a set of matrices $F_1 \dots F_{N_f}$. Each matrix F_i performs the same set of linear operations such as convolving an image with a filter f_i. In other words, if $\hat{\mathcal{X}}$ is a vector created by unwrapping the image $\mathcal{X}$, then $F_i \hat{\mathcal{X}}$ is identical to $\mathcal{X} * f_i$. These matrices can then be stacked and $C(\mathbf{x}, \mathbf{y})$ can be rewritten as

$$C(\mathbf{x}, \mathbf{y}) = (F\hat{\mathcal{X}} - r)^T W(\mathbf{y})(F\hat{\mathcal{X}} - r), \qquad (10)$$

where

$$F = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{N_f} \end{bmatrix} \quad \text{and} \quad r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{N_f} \end{bmatrix}.$$

The matrix $W(\mathbf{y})$ is a diagonal matrix that holds the weights $w(i, j; \mathbf{y})$. This diagonal matrix is a block-diagonal matrix constructed from the diagonal sub-matrices $W_1(\mathbf{y}; \theta) \dots W_{N_f}(\mathbf{y}; \theta)$. Each entry along the diagonal in the $W_k(\mathbf{y}; \theta)$ matrices is equal to the weight of a term at a particular pixel, $w_k(i, j; \mathbf{y}, \theta)$.

This matrix representation is used in Section 3 to describe the trainining methods.

2.8. Solving The Weighted Least-Squares System

Finding the image $\mathbf{x}$ that maximizes Equation 8 is equivalent to solving the weighted least-squares system, $C(\mathbf{x}, \mathbf{y})$. Thus, the probability that the label at pixel m, l_m, should have class $+1$ is:

$$p(l_m = +1|\mathbf{y}) = \sigma(x_m) \qquad (11)$$

where x_m is the pixel with index m from $\mathbf{x}$. Again, the response image $\mathbf{x}$ is the MAP solution to the distribution in Equation 8. This solution can be expressed as:

$$\mathbf{x} = \left(F^T W(\mathbf{y}) F \right)^{-1} F^T W(\mathbf{y}) r \qquad (12)$$

Expressing $\mathbf{x}$ as the above set of matrix-vector products simplifies computing the gradients of the loss function defined in Equation 13 when training.

3. Discriminative Training for LRFs

Similar to parameter fitting in the logistic regression model, the parameters of the LRF can be found by maximizing the likelihood of the training data. In this section, we will focus on describing how to optimize the vector parameter θ, which control the weight assigned to different terms. This model is similar to those models used in [14, 5, 23].

The vector parameter θ can be found by minimizing the negative log-likelihood of the training dataset. We define the negative log-likelihood of a training image, denoted $L(\theta)$ for a particular set of parameters θ, as

$$L(\theta) = \sum_{i=1}^{N_p} \log \left(1 + \exp(-t_i \mathbf{x}_i) \right) \qquad (13)$$

where t_i is the ground-truth of each pixel. Note that $L(\theta)$ depends on θ via $\mathbf{x}_i$.

Using $L(\theta)$ as the optimization criterion, the gradient with respect to a vector parameter θ_n can be easily computed:

$$\frac{\partial L(\theta)}{\partial \theta_n} = \frac{\partial L(\theta)}{\partial \mathbf{x}}^T \frac{\partial \mathbf{x}}{\partial \theta_n} \qquad (14)$$

For reference, $\mathbf{x}$ is the response image that minimizes $C(\mathbf{x}, \mathbf{y})$. With $\mathbf{x}$ being computed with a matrix inverse, it can be differentiated analytically. The derivation of this gradient is similar to the derivative of the gradient for a Gaussian Condition Random Field Model, which is described in [21]. The primary difference is that the GCRF model uses a squared-error loss function rather than the logistic function used here.

We will not derive the gradient here, but instead refer the reader to [21]. The gradient can be expressed succinctly using the matrix notation from Equation 10:

$$\frac{\partial L(\theta)}{\partial \theta_i} = \frac{\partial L(\theta)}{\partial \mathbf{x}}^T \left(F^T W(\mathbf{y}) F\right)^{-1} F^T \frac{\partial W(\mathbf{y})}{\partial \theta_i} (r - F\mathbf{x}) \tag{15}$$

An important practical issue regarding Equation 15 is that the product $\frac{\partial L(\theta)}{\partial \mathbf{x}}(A^T W(\mathbf{y})A)^{-1}$ only needs to be computed once per gradient iteration by taking advantage of associativity in matrix multiplication. This makes the overall complexity of computing the gradient $\mathcal{O}(N^2)$. The rest of the gradient calculations only involve convolutions and point-wise operations.

With these gradient calculations, it is possible to optimize the model parameters using gradient descent techniques.

4. Related Work on MRF Parameter Estimation

The MRF models that are the focus of this paper are a particular instance of a broader class of structured models. Parameter estimation for structured models has been a topic of considerable interest recently. This section reviews several significant algorithms in this area. Section 4.1 will discuss the advantages and disadvantages of the LRF models in comparison with the algorithms discussed here.

In [5], Collins shows how the perceptron algorithm can be adapted to discriminatively train a structured model for classification. At each iteration, the algorithms must recompute the optimal labeling under the model, then update the weights appropriately. While it can be shown that the perceptron algorithm converges when the data is linearly separable, convergence has not been proven when the data is not linearly separable.

Taskar et al. shows how the parameters of a structured model can be found in a maximum margin framework [23]. Using this framework, the weights in a model similar to that used in Section 2.8 can be found with convex optimization.

Taking a probabilistic approach, Wainwright et al. [27] have proposed upper-bounds on the log-partition function for discrete-valued graphical models. This allows for parameters to be estimated by minimizing an upper-bound of the negative log-likelihood of the training data. Levin et al. use this technique to learn parameters in their feature-

choosing algorithm. Hinton's Contrastive Divergence approach has also proven successful in learning MRF parameters [8, 7].

Pseudo-likelihood methods [1] are also a popular approach for finding MRF parameters [12]. However, Blake et al. have pointed out potential instabilities [2] and Vishwanathan et al. have found that the pseudo-likelihood leads to over-smoothed estimates.

4.1. Advantages and Disadvantages of the LRF

A key difference between our approach and these previous approaches is that the LRF formulation is based on a quadratic model. The quadratic model is specifically chosen to allow the optimal solution of the underlying quadratic cost model to be differentiated with respect to the model parameters. This enables the log-likelihood of the training data to be differentiated with respect to the model parameters.

The advantages of the LRF model include:

- **Convenience** – The LRF, as mentioned previously, is based on a quadratic model. This makes it possible to perform inference and parameter estimation using standard linear algebra routines.

- **Convergence** – Because the LRF model allows exact gradients to be computed, the training procedure can be guaranteed to converge by choosing the appropriate optimization algorithm.

- **Flexibility** – The fact that the log-likelihood is differentiable is a significant advantage because it allows greater flexibility in constructing the parameterization of the model. Any parameter can be optimized, as long as the cost function $C(\mathbf{x}, \mathbf{y})$ can be differentiated with respect to that parameter. For instance, parameters inside sigmoid-type functions can be directly optimized. In some other popular models for learning MRF parameters, such as the max-margin model, only weights on features can be optimized. This limits the type of parameterizations that can be used. In addition, the LRF model can accommodate arbitrarily large clique potentials.

The most significant disadvantage of the LRF model is that the optimization criterion is not convex, due to the matrix inversion step. This makes the training procedure susceptible to falling into local minima. Due to the underlying quadratic model, the LRF model is also limited to associative problems, similar to [23].

4.2. An Example of the LRF's Flexibility: Super-Pixels

Incorporating super-pixels into the MRF model is a good example of the LRF's flexibility and convenience. The basic

idea behind super-pixels is to over-segment the image into a large number of small segments then assume that all of the pixels in each segment will have the same label [17]. In this section, we will show how the LRF model can be adapted to use super-pixels through a trivial modification to the model and the system source code.

To incorporate super-pixels, we constructed a super-pixel matrix, which we will denote as S. This matrix converts a super-pixel image with N_S super-pixels into a standard image with N_p pixels by copying the value associated with each super-pixel image to all of the image pixels associated with that super-pixel. Practically if S is an $N_p \times N_S$ matrix, each row will have one column that is equal to 1, with all of the remaining columns being equal to 0.

Modifying the underlying GCRF model is trivial. Using the notation from Equation 10, the matrix F is replaced with F_S where $F_S = FS$. The only other modification involves the loss function. We wish to use the original cost function, so we will introduce an intermediate image, $\mathbf{x}_s$, which is the current super-pixel image. The image $\mathbf{x}$ will continue to represent the full image with $\mathbf{x} = S\mathbf{x}_s$. Using this intermediate representation, the gradient from Equation 14 can be rewritten as

$$\frac{\partial L(\theta)}{\partial \theta_n} = \frac{\partial L(\theta)}{\partial \mathbf{x}}^T S \frac{\partial \mathbf{x}_s}{\partial \theta_n} \qquad (16)$$

Hence, the full gradient can be rewritten as

$$\frac{\partial L(\theta)}{\partial \theta_i} = \frac{\partial L(\theta)}{\partial \mathbf{x}}^T S \left(F^T W(\mathbf{y})F \right)^{-1} F^T \frac{\partial W(\mathbf{y})}{\partial \theta_i}(r - F\mathbf{x}) \qquad (17)$$

4.2.1 Practical Benefits of Super-Pixels

The major benefit of super-pixels is an increase in speed. When used with the horses images (Section 5.3) we observed a 30% reduction in the time required to compute gradients. However, overall performance dropped to 91%. One could take advantage of the speed increase by using super-pixels to perform fast initial training to perform a general initialization to system weights; this could then be followed by more in-depth training to better refine the model weights.

5. Comparing the LRF Model with Traditional MRF Models

In this section, we demonstrate how the LRF model can be as expressive and useful as a standard MRF model. To do this, we implemented an object-segmentation system similar to that described by Levin [14].

Segmentation is a very active area of research. Recent systems have made advances such as incorporating 3D cues [9] and making it possible to be trained using unsupervised

data sets [28]. While we believe that our segmentation results are good, our primary motive is to see if utilizing an LRF instead of standard MRF model degrades the system's performance. Hence, we have focused on the model similar to that in [14]. This will enable us to ensure that the LRF formulation is not only convenient to work with, but also can also produce results that are just as good as traditional MRF models.

5.1. Datasets

To compare the performance of the LRF model with traditional MRF models we chose two datasets: cows and horses. The images in the cow dataset were from video sequences used by [13, 15] for lame horse identification and segmentation. The horse dataset was taken from the Weizmann Horse database used by Borenstein [3] for segmentation.

Our system is trained to maximize the probability that a given pixel is part of the desired object. To measure segmentation accuracy we labeled a pixel, x, as belonging to the object if $P(x = +1|y) > 0.5$ in the LRF result. Similar to Levin [14], we report the percentage of correctly classified pixels in the resulting segmented image.

5.2. Features

For the cow and horse datasets local evidence is gathered using features similar to those used in [14]. Each feature consists of an image patch, a segmentation mask, and a spatial window. It is helpful to understand the feature generation process by thinking in terms of generating a feature response image for each feature. These response images contain the response of a feature at every pixel in the image.

A feature response image is created using the following steps:

1. Compute the absolute value of the normalized correlation between the image and the feature's image patch at every pixel in the image.

2. Find the maximum of this image within the feature's spatial window.

3. Place the segmentation mask at this point. Those pixels in the sementation mask that belong to the object are labeled +1, while those that do not are labeled -1

4. Scale this image by the maximum absolute normalized correlation found in Step 2.

5. Generate a set of twenty binary feature images that compare every pixel against thresholds in the range [-1,1]. Each binary response retains its sign from Step 3.
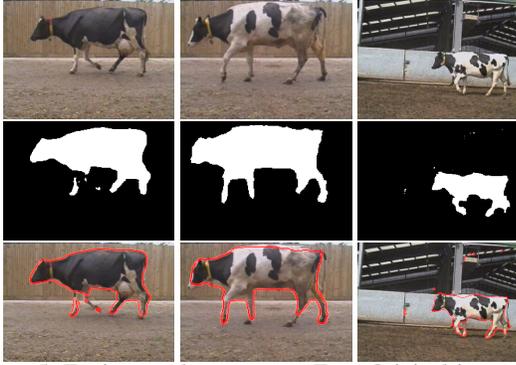
Figure 5. Testing results on cows. Top: Original images. Middle: LRF segmentation masks. Bottom: Resulting segmentations denoted by red outline.

The purpose of the final step is to allow the system to learn appropriate thresholds on the feature responses. One advantage of this approach is that all of the thresholds are considered in parallel.

The feature image patches, segmentation masks, and spatial windows were taken from patches in the training set. These features were chosen using a greedy method that maximized the probability of correct segmentations using only the features.

In addition to the base features, we also added features corresponding to the horizontal and vertical gradient magnitudes at each pixel in the image.

5.3. Horses

The horse dataset is comprised of 328 images from the Weizmann Horse Database. This set was randomly separated into 197 training images and 128 testing images, 3 of the images were removed due to poor ground truth representation. When measuring the LRF Model for segmentation we were able to produce an average classification of 94.6% across the testing set. Figure 6 shows some of the segmentation results obtained with horses. Our results were close to those reported in [14] at 95%.

5.4. Cows

The cow dataset is comprised of 110 images from the training data of Leibe [13]. This set was randomly separated into 80 training and 30 testing images. When we measured the performance of the LRF system on the testing portion of the cow dataset as done by Levin [14] we obtained an average classification rate of 97%. This compares well with Levin [14], which reported 92% accuracy. Figure 5 shows some of the segmentation results obtained on the cow dataset.

6. Conclusion

We have proposed a new model for representing a discrete-valued Markov Random Field. The Logistic Random Field, or LRF, model is computationally efficient, flexible and convenient. When using the LRF model, computing gradients for parameter estimation can be accomplished in polynomial time using basic linear algebra routines. Also, every parameter in the LRF model can be potentially differentiated and optimized. This flexibility and convenience makes the LRF a potentially valuable tool for applications relying on discrete-valued graphical models.

In addition to demonstrating the flexibility of the LRF model by incorporating super-pixels, we have shown that when applied to the task of segmenting specific objects, the LRF model produces results that are competitive with approaches that use more traditional MRF models.
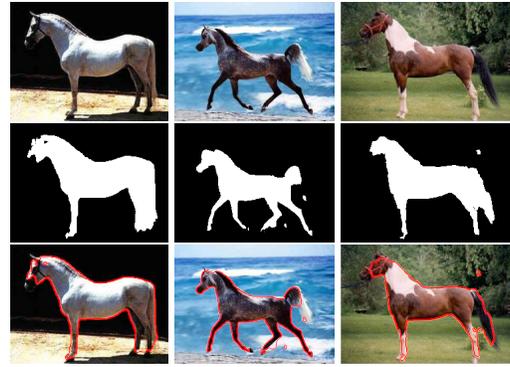
Several learning mechanisms for graphical models have been introduced recently, but no controlled comparisons between these different methods are available. In the future, it would be valuable to set up a rigorous comparison, similar to the recent evaluations of inference algorithms by Szeliski et al. [19].

References

[1] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195, Sept 1975.

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *European Conference on Computer Vision (ECCV)*, 2004.

[3] E. Borenstein and S. Ullman. Learning to segment. In *European Conference on Computer Vision (ECCV)*, May 2004.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[5] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*, 2002.

[6] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

[7] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labelling. In *In 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[8] G. Hinton. Training products of experts by minimizing contrastive divergence, 2000.

[9] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[10] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern*

Testing results on dark colored horses.   Testing results on patched or light colored horses.

Figure 6. Top: Original images. Middle: LRF segmentation masks. Bottom: Resulting segmentations denoted by red outline.

Analysis and Machine Intelligence (PAMI), 28(10):1568–1583, October 2006.

[11] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego*, 2005.

[12] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proceedings of the 2003 IEEE International Conference on Computer Vision (ICCV '03)*, volume 2, pages 1150–1157, 2003.

[13] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV'04 Workshop on Statistical Learning in Computer Vision*, pages 17–32, Prague, Czech Republic, May 2004.

[14] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *ECCV (4)*, pages 581–594, 2006.

[15] D. Magee and R. Boyle. Detecting lameness using 're-sampling condensation' and 'multi-stream cyclic hidden markov models'. *Image and Vision Computing*, 20(8):581–594, 2002.

[16] J. Portilla, V. Strela, M. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Image Processing*, 12(11):1338–1351, November 2003.

[17] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 10–17, 2003.

[18] S. Roth and M. Black. Field of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 860–867, 2005.

[19] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV (2)*, pages 16–29, 2006.

[20] M. F. Tappen, E. H. Adelson, and W. T. Freeman. Estimating intrinsic component images using non-linear regression. In *The Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1992–1999, 2006.

[21] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning gaussian conditional random fields for low-level vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR07)*, 2007.

[22] M. F. Tappen, B. C. Russell, and W. T. Freeman. Efficient graphical models for processing images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 673–680, 2004.

[23] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, 2005.

[24] B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction via the extragradient method. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006. MIT Press.

[25] M. J. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, 7:1829–1859, September 2006.

[26] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, November 2005.

[27] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, July 2005.

[28] J. Winn and N. Jojic. Locus:learning object classes with unsupervised segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 756–763 Vol. 1, 2005.