10-708: Probabilistic Graphical Models 10-708, Spring 2014

28 : Maximum-Margin Learning of Graphical Models

Lecturer: Eric P. Xing *Scribes: T.Hoang Ngan Le and Felipe Hernández*

1 Introduction

The goal of classical predictive models like logistic regression or kernel-based approaches such as support vector machines (SVM) is to find a function which assigns a label to objects given a set of features. Even though these techniques have strong theoretical guarantees and work well on high-dimensional spaces, they ignore the underlying structure that the features may possess as in the cases of sequences, spatial entities, and images. Graphical models like Markov networks are able to overcome those weaknesses by directly encoding interdependencies and thus being able to exploit the structure. However, they cannot handle high-dimensional feature spaces and lack strong theoretical guarantees.

The strengths of these two approaches, the classical predictive models and the graphical models, can be merged to create classifiers that both perform well in high-dimensional cases and that take advantage of the additional information provided by the structure of the variables. Additionally, further modifications can provide these methods with a straightforward probabilistic interpretation. In this way, additional advantages of graphical models, such as induced sparsity, and the incorporation of latent variables and structure, can be also be exploited.

In these notes we first review two classical predictive models: logistic regression and SVMs. Then, we show how they were extended, first by showing how SVM maximum-margin approaches can be modified to account for structure information. This idea lead to the development of the Maximum Margin Markov Networks (M^3N). Second, we explore how a probabilistic interpretation can be added to such models as in the case of Maximum Entropy Discrimination (MED). Finally, we show how both the structured and probabilistic perspectives can be merged to create the Maximum-Entropy Discrimination Networks (MaxEnDNet).

The roadmap of these developments is shown in Figure 1. In order to illustrate the benefits of M^3N and MaxEnDNet, experimental results on Optical Character Recognition (OCR) and hypertext classification problems are shown.

2 Unstructured Predictive Models

The purpose of the classical predictive models is to learn a predictive function $h(\boldsymbol{x})$ that maps an input vector $\mathcal{X} \in R^M$ in a high-dimensional feature space to an output $\mathcal{Y} \in \{-1, +1\}$ in a labeling space (which is a discrete set of labels, most commonly binary). This is also called unstructured prediction. Generally, the predictive function $h(\boldsymbol{x})$ can be represented by Eq. 1

$$h(\boldsymbol{x}) = \boldsymbol{y}^* = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) \tag{1}$$

such that we can infer the optimal $\boldsymbol{y}^*$ which maximizes the predictive loss function. In this equation,

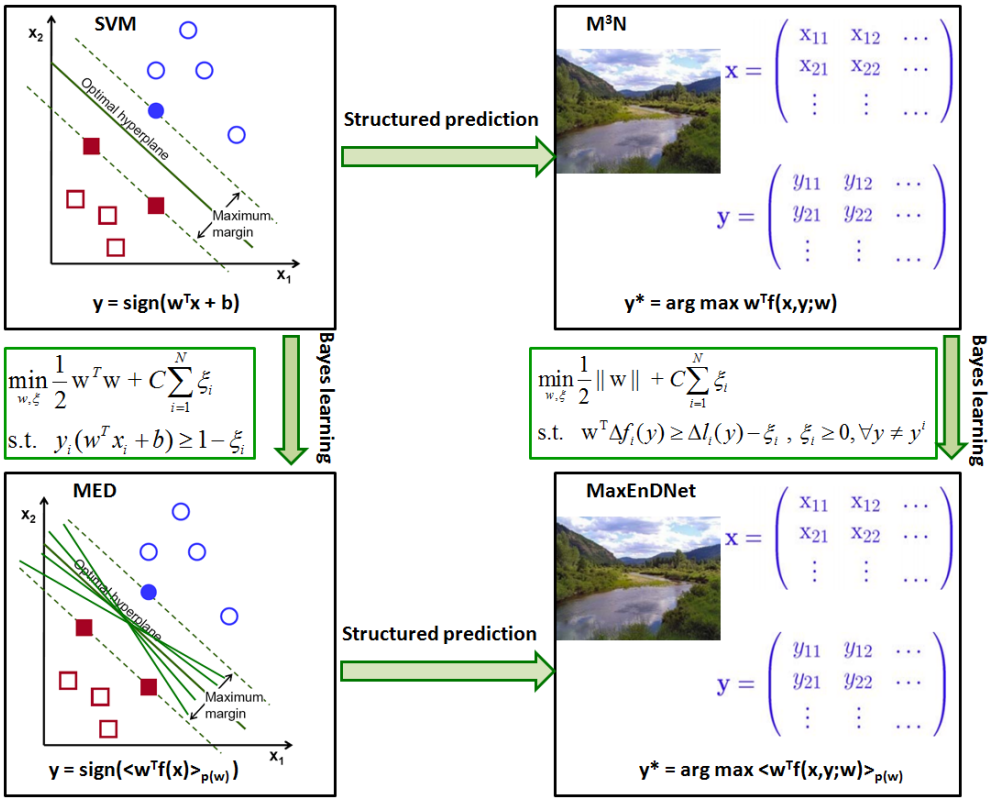


Figure 1: Roadmap of the development of structured probabilistic variants of the maximum-margin Support Vector Machines (SVM). The Maximum Margin Markov Networks (M^3N) are a structured version of SVMs, while Maximum Entropy Discrimination (MED) turns the problem into a probabilistic one. The combination of both enhancements results in the Maximum-Entropy Discrimination Networks (MaxEnDNet).

$F(\boldsymbol{x}, \boldsymbol{y}; w) = g(\boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}))$ is usually defined as a log linear model which is translated to linear combination of features defined on the data, and $g(\cdot)$ is a cover function on top of this. Our goal is to define $\hat{\boldsymbol{w}}$ as follows:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w} \in W} l(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) + \lambda R(\boldsymbol{w}) \tag{2}$$

where $l(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$ is an arbitrary loss function, and $R(\boldsymbol{w})$ is a shrinking or regularizing function to prevent overfitting. Sometimes, we use it to define a constraint such as $\boldsymbol{w}$ being sparse. It is also considered to be a prior in a Bayesian sense and thus $\lambda R(\boldsymbol{w})$ is a prior distribution of the weights. To make it more concrete, let's consider two classical predictive models: logistic regression and Support Vector Machines (SVM) [1]:

- **Logistic regression**: Assume that we choose the log function to be the conditional probability of the label giving the data, meaning that we are working on a Bayesian logistic regression. In this case the optimization problem is defined as in Eq. 3:

$$\max_{\boldsymbol{w}} \mathcal{L}(\mathcal{D}; \boldsymbol{w}) \hat{=} \sum_{i=1}^{N} \log \mathrm{p}(y^i | x^i; \boldsymbol{w}) + \mathcal{N}(\boldsymbol{w}) \tag{3}$$

  In this equation, we introduce a prior distribution $\mathcal{N}(\boldsymbol{w})$ on the weights and then we perform a

Bayesian estimation of the weights. Basically, Eq. 3 is equivalent to maximizing the likelihood or maximizing a posterior distribution (MAP). Notably, the function $g(\cdot)$ takes the form of a sigmoid function, $g = \frac{\exp(\boldsymbol{w}^T \boldsymbol{f})}{1+\exp(\boldsymbol{w}^T \boldsymbol{f})}$. This corresponds to the predictive function defined in Eq. 2. The log loss (as shown in Figure 2) is defined as:

$$l_{LL}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) \hat{=} \sum_{y \in \mathcal{Y}} \exp[\boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}')] - \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) \tag{4}$$

In the logistic regression approach, we can have all the benefits from the probabilistic semantics model, such as introducing a prior distribution and adding hidden variables, if the hierarchical model is built.

- **Support Vector Machines (SVM)**: SVMs are also known as support vector networks and have been successful in their ability of using kernels, allowing for classification in high-dimensional feature spaces. In addition, SVMs are also appealing due to the existence of strong generalization guarantees, derived from the margin-maximizing properties of their learning algorithm. In SVMs, we solve the following problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2} \boldsymbol{w}^T \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & \forall i, \forall \boldsymbol{y}' \neq y^i : \boldsymbol{w}^T \Delta f_i(y') \geq 1 - \xi_i \xi_i \geq 0 \end{aligned} \tag{5}$$

In this equation, we are minimizing an l_2 regularized weight vector $\boldsymbol{w}^T \boldsymbol{w}$ while minimizing the predictive margin loss $\sum_{i=1}^{N} \xi_i$. We also consider all training data to induce margin constraints. If we have N training samples, there are N constraints. In term of SVMs, the function $g(\cdot)$ is defined as an indicator function ($g(\cdot) = 1$), since SVMs are linear predictors. The corresponding predictive function is defined in Eq. 2, and the hinge loss (as shown in Figure 2) is defined as:

$$l_{MM}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) \hat{=} \max_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}') - \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + l'(\boldsymbol{y}', \boldsymbol{y}) \tag{6}$$

In machine learning research, SVMs have a number of key advantages like the kernel tricks, which make the computation fast even with a huge numbers of features and also helps us work with nonparametric predictors. In addition, it has dual sparsity which is related to the use of a small subset of the training data. This means that only a few support vectors are used for determining the decision boundary. Furthermore, there are many strong empirical results in the literate on a wide range of application areas.

Besides the unstructured predictive models mentioned above, which are not every useful in some real applications where the data has an underlying structure and where labels are dependent in complex ways. Such is the case of speech tagging and image segmentation. An alternative approach is offered by the probabilistic framework, and specifically by probabilistic graphical models. For example, we can learn a hidden Markov model (HMM) [1] or a conditional random field (CRF) [2] over the labels and features of a sequence, and then use a probabilistic inference algorithm to find the label. In such structure predictive models, the input is a feature matrix and the output is label matrix. Let's consider a case of sequentially structure data with handwriting recognition problem, a sample of which is shown in Figure 3. An example of a graphical model applied in this case is shown in Figure 4.

We have a sequence of $\boldsymbol{x}$ variables, each of which is a vector of pixel values, and a corresponding sequence of $\boldsymbol{y}$ variables, each of which can take on 26 values (the letters of the alphabet). The conditional probability of $\boldsymbol{y}$ given $\boldsymbol{x}$ is a product of node and edge potentials as follows:

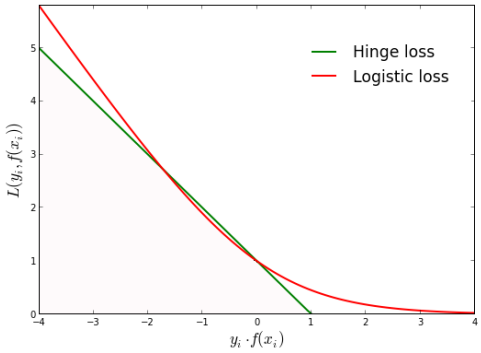


Figure 2: An illustration of log loss and hinge loss functions.

Figure 3: Handwriting recognition example (a) entire word (b) character (part of the word).

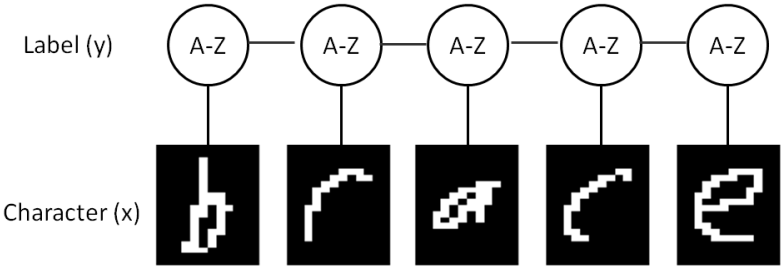


Figure 4: A graphical model over characters.

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \prod_i \phi(x_i, y_i)\phi(y_i, y_{i+1}) \tag{7}$$

where $\phi(x_i, y_i) = \exp[\sum_\alpha w_\alpha f_\alpha(x_i, y_i)]$ and $\phi(y_i, yi+1) = \exp[\sum_\beta w_\beta f_\beta(y_i, y_{i+1})]$.

The node potentials roughly correspond to emission probabilities in a HMM, while the edge potentials correspond to transition probabilities. Notably, these potentials do not need to sum to 1 as in HMMs, but simply need to be positive functions. A natural way to represent potentials is by using a log linear combination of basic functions. The basic functions here are indicator functions asking a question like is "current the letter a 'z' and the next one an 'a'?"

From the example in Figure 3, it is easy to recognize that the word is "brace"; however, it is hard to label the character in Figure 3(b) as to be 'r' or 'c' in isolation. In this example, there is a model over characters that is already explicitly built (as shown in Figure 4), which helps to tell how confident a current label is based on the previous one. For example, if we already recognize 'b' at the beginning, then 'r' occurring after 'b' is more confident than 'c' occurring after 'b'.

Image segmentation using CRF is another example of a predictive problem in which different labels can connect with some kind of potential function that encourages them to share labels among neighboring pixels. Parsing of sentences is also an example which has to follow syntactic validity of a tree. It is hard to make correct predictions if the tree structure is ignored by only considering the leaves independently.

3 Structure Predictive Models

In the case of probabilistic graphical models, we can define and learn a joint probabilistic model over the set of label variables. For example, we have already learnt Hidden Markov Model (HMM) [1] or Conditional Random Field (CRF) [2] on previous lectures. These distributions are defined over the labels and features of a sequence, and then a probabilistic inference algorithm is used to classify the instances. It is easy to see that these methods are able to exploit the correlation between different labels and effectively take advantage of the structure of the problem. However, the performance of these probabilistic graphical models is not on the same level of generalization accuracy as SVMs, especially when kernel features are used.

Max-Margin Markov Networks (M^3N) are then proposed. They make use of both frameworks of kernel-based and probabilistic classifiers. M^3N is also known as a counterpart of SVM in the structured prediction domain, meaning that M^3N is trained with the same procedure as CRFs using a large margin principle. That's why these networks are sometimes called structured SVMs instead of Markov Networks. M^3N defines a log-linear Markov network over a set of label variables. A margin-based optimization problem is also defined in this approach. In the M^3N model, the goal is to learn the predictive function $h(\cdot)$ which is built on maximizing the score function. It is defined as follows:

$$h(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(x)} s(\boldsymbol{x}, \boldsymbol{y}) \tag{8}$$

where the score function can be defined as a linear combination of feature as given by Eq. 9. In the simple case, the function $g(\cdot)$ is set as an indicator ($g(\cdot) = 1$).

$$s(\boldsymbol{x}, \boldsymbol{y}) = g[\boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y})] = g\left[\sum_p \boldsymbol{w}^T f(x_p, y_p)\right] \tag{9}$$

Let's consider the training procedure of M^3N with $\boldsymbol{x}$ being the vector representing the input, and $\boldsymbol{y}$ the vector representing the labels. Let $\boldsymbol{y^*}$ be the predicted labels. When we are predicting multiple labels, we incorporate not only the {0, 1} loss I but also the proportion of the incorrect labels predicted. If $\{\boldsymbol{x}, \boldsymbol{y}^*\}$ are the training examples, the objective function is to maximize the loss function in Eq. 10:

$$\boldsymbol{y^*} = \arg\max_y \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) \tag{10}$$

Furthermore, we also want to make the prediction risk (the SVM risk) as small as possible as shown in Eq. 11. That means, among many solutions for $\boldsymbol{w}$, the best one is also maximizing the margin γ.

$$\boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}^*) > \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + \gamma, \forall \boldsymbol{y} \neq \boldsymbol{y^*} \tag{11}$$

On the other hand, we would like the margin between $\boldsymbol{y^*}$ and $\boldsymbol{y}$, $l(\boldsymbol{y^*}, \boldsymbol{y})$, to scale linearly to the number of wrong labels in $\boldsymbol{y}$. The M^3N framework is then defined as follows:

$$\begin{array}{ll} \min & \gamma \\ \text{s.t.} & ||\boldsymbol{w}|| \leq 1; \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}^*) \geq \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + \gamma l(\boldsymbol{y}^*, \boldsymbol{y}), \forall \boldsymbol{x} \in \mathcal{X} \text{ and } \forall \boldsymbol{y} \end{array} \tag{12}$$

where $l(\boldsymbol{y}^*, \boldsymbol{y}) = \sum_i^l I(y_i \neq y_i^*)$ in which y_i are the labels in the training data and y_i^* are the predicted labels. Using a standard transformation to eliminate γ, we can get the following quadratic program:

$$\begin{array}{ll} \max & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t.} & \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}^*) \geq \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + l(\boldsymbol{y}^*, \boldsymbol{y}), \forall \boldsymbol{x} \in \mathcal{X} \text{ and } \forall \boldsymbol{y} \end{array} \tag{13}$$

However, the number of constraints is exponential in size of the structure. An alternative option is to add only the most violated constraint. Let

$$\boldsymbol{y}' = \arg\max_{\boldsymbol{y} \neq \boldsymbol{y}^*} \boldsymbol{w}^T f(x_i, \boldsymbol{y}) + l(y_i, \boldsymbol{y}) \tag{14}$$

We can then add the constraint:

$$\boldsymbol{w}^T f(x_i, \boldsymbol{y}) \geq \boldsymbol{w}^T f(x_i, \boldsymbol{y}') + l(y_i, \boldsymbol{y}) \tag{15}$$

Then the optimization problem becomes:

$$\begin{array}{ll} \max & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t.} & \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}*) \geq \max_{\boldsymbol{y} \neq \boldsymbol{y}^*} \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + l(\boldsymbol{y}^*, \boldsymbol{y}), \forall \boldsymbol{x} \in \mathcal{X} \text{ and } \forall \boldsymbol{y} \end{array} \tag{16}$$

This formulation only needs a polynomial number of constraints and can handle a more general loss function. In order to solve the above problem, the minimization in the constraint needs to be converted from a discrete to a continuous form before we can invoke a QP solver.

Let $z_i(m)$ be the indicator variable for the class corresponding to node i and label m. Let $z_{ij}(m, n)$ be the indicator variable for the joint class of nodes (i, j) with labels (m, n). Then the right hand side of the constraint in Eq.16 can be rewritten as follows:

$$\begin{array}{rl} & \max_{y \neq y*} \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}) + l(\boldsymbol{y}^*, \boldsymbol{y}) \\ = & \max_z \sum_{j,m} z_j(m) \left[\boldsymbol{w}^T f_{node}(x_j, m) + l_j(m)\right] + \sum_{j,k,m,n} z_{jk}(m,n) \left[\boldsymbol{w}^T f_{edge}(x_{jk}, m, n) + l_{jk}(m,n)\right] \\ = & \max_z (F^T \boldsymbol{w} + l)^T \boldsymbol{z} \\ & \text{s.t.} \sum_m z_j(m) = 1; \sum_n z_{jk}(m,n) = z_j(n); z_j(m) \geq 0; z_{jk}(m,n) \geq 0 \end{array} \tag{17}$$

Since all the constraint in Eqs. 17(d-g) can be represented as $A\boldsymbol{z} = \boldsymbol{b}$, the optimization problem now becomes:

$$\begin{array}{ll} \max & (F^T \boldsymbol{w} + l)^T \boldsymbol{z} \\ \text{s.t.} & A\boldsymbol{z} = \boldsymbol{b} \end{array} \tag{18}$$

Thus, the problem defined in Eq. 16 is defined in the new formulation as follows:

$$\begin{array}{ll} \max & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t.} & \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y}*) \geq \max\limits_{\boldsymbol{z} \geq 0, A\boldsymbol{x} = \boldsymbol{b}} \boldsymbol{q}^T \boldsymbol{z} \end{array} \tag{19}$$

where $\boldsymbol{q}^T = \boldsymbol{w}^T F + l^T$. This has integer $\boldsymbol{z}$ solutions for chains and trees, but general fractional solutions for untriangulated networks. Using strong Lagrangian duality, we can write:

$$\max_{\boldsymbol{z} \geq 0, A\boldsymbol{x} = \boldsymbol{b}} \boldsymbol{q}^T \boldsymbol{z} = \min_{A^T \mu \geq \boldsymbol{q}} \boldsymbol{b}^T \mu \tag{20}$$

The problem now becomes:

$$\begin{array}{ll} \max & \frac{1}{2}||\boldsymbol{w}||^2 \\ \text{s.t.} & \boldsymbol{w}^T f(\boldsymbol{x}, \boldsymbol{y^*}) \geq \boldsymbol{b}^T \mu; A^T \mu \geq \boldsymbol{q} \end{array} \tag{21}$$

In order to show the advantages of M^3N over the unstructured predictive models, let's consider two following experiments on Optical Character Recognition (OCR) and hypertext classification.

- **Experimental result 1 (OCR)**: Let's consider the OCR problem with three different models, namely multi-class SVMs, CRFs, and M^3N under three cases: raw pixels, quadratic kernels, and cubic kernels. With the first model, multi-class SVMs (26 outputs), the dependencies between the labels are ignored since there is no way to coordinate them and the test error is 27% for the raw pixels. Herein, the average per-character test error is used over 10 folds.

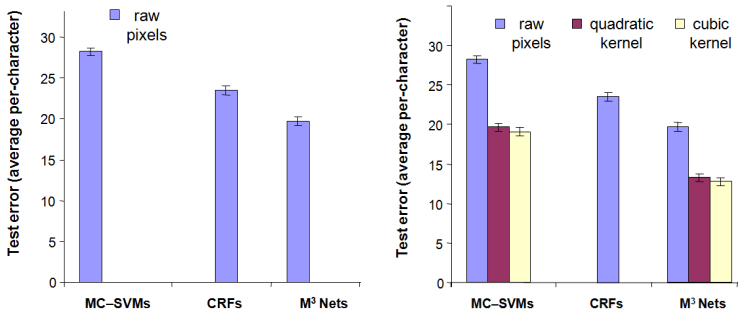
  The second model as a first order CRFs which exploits correlations between adjacent letters. Next, we use a M^3N with the same set of basic functions as the CRFs, achieving a significant reduction in error even before using kernels as can be seen in Figure 5(a).

  Then we compared kernel-based approaches using simple polynomial kernels which consider all pixel pairs and triplets. Kernel SVMs do significantly better than even the CRF. When we incorporate these kernels into M^3N, we get a large additional boost.

  Overall, our model exploits the advantages of both the high-dimensional kernels and the sequential structure of the problem to gain a 45% error reduction over CRFs and 33% over kernel SVMs as shown in Figure 5(b).

- **Experimental result 2 (Hypertext classification)**: In this experiment, a WebKB dataset is used, which contains four CS departments websites (Cornell, Texas, Washington, and Wisconsin) with 1300 pages and 3500 links. Each page is labeled with one of course, faculty, student, project, and other. Three CS departments are used for training and the fourth one is used for testing.

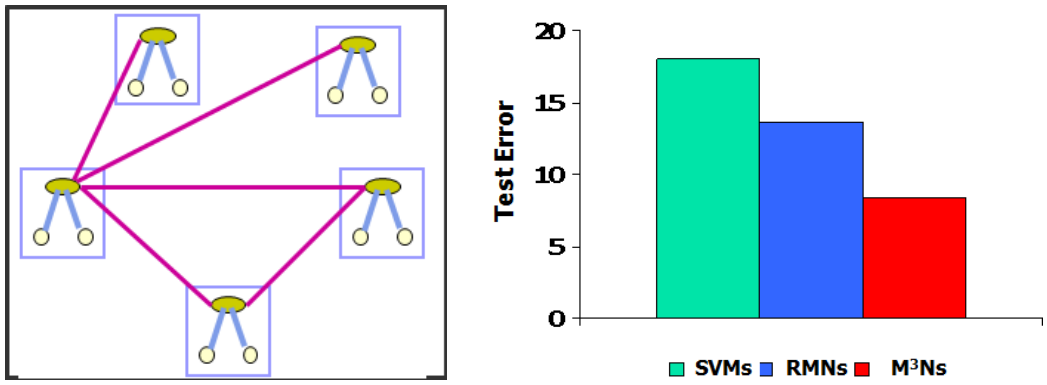
  There are three different models used in this experiments. The first one is a linear SVM that classifies each page based on the bag of words it contains. The second one is a relational Markov network (RelMN) which has an edge between hyperlinked pages. This model captures very strong correlations between the labels of linked pages and achieves a significant gain over the SVM. Note that inference



(a) (b)

Figure 5: (a) Test error of multi-class SVMs, CRFs, and M^{3}N with the raw pixels feature; (b) Test error of multi-class SVMs, CRFs, and M^{3}N with the raw pixels feature, quadratic kernel and cubic kernel.

in this model is intractable, so we used loopy belief propagation. The third model is M^{3}N which uses the relaxed dual and is used without the clique-tree constraints. It achieves an error reduction of 19% over the Markov network with the same features as shown in Figure 6.



(a) (b)

Figure 6: (a) Loopy belief propagation; (b) Test error of SVM, RelMN and M^{3}N

4 Maximum Entropy Discrimination Graphical Models

4.1 Maximum Entropy Discrimination

Maximum Entropy Discrimination (MED) was proposed by [3] looking to combine the strengths of both likelihood-based estimation and maximum margin learning. This fused methodology is desirable in cases where you would like to apply classification with a kernel or a non-parametric approach, like in SVMs, and that is also able to handle partially observed data or hidden variables as in graphical models like HMMs or CRFs [4]. Instead of simply finding the weight vector $\boldsymbol{w}$ for classification, MED is made probabilistic by learning the distribution of the weights $P(\boldsymbol{w})$. Inference can then be performed through model averaging:

$$\hat{\boldsymbol{y}} = \text{sign} \int p(\boldsymbol{w}) F(\boldsymbol{w}) d\boldsymbol{w} \tag{22}$$

The learning problem, in the case of binary classification, can be formulated as a minimum entropy problem through the minimization of the divergence of the posterior from an arbitrary prior. This approach helps in directly reducing the randomness (entropy) of the transformation, which is the analogous process of minimizing the norm of the weights in the original SVM formulation. Additionally, this is done while satisfying the expected margin constraint, which is the probabilistic version of SVM's margin constraint:

$$\begin{aligned} \min_{p(\Theta)} \quad & KL[p(\Theta)||p_0(\Theta)] \\ \text{s.t.} \quad & \int p(\Theta)[y_i F(\boldsymbol{x}; \boldsymbol{w}) - \xi_i] d\Theta \geq 0, \forall i. \end{aligned} \tag{23}$$

Θ is the parameter $\boldsymbol{w}$ when ξ are kept fixed or the pair $(\boldsymbol{w}, \xi)$ when we want to optimize over ξ. This innovative approach showed to perform slightly better than SVMs and presented higher stability. However, the improvement in performance was not significant enough to overcome the additional complexity of the model and, therefore, did not become very popular.

4.2 Maximum Entropy Discrimination Markov Networks

MED was successful in proposing a probabilistic version of SVMs. Is it possible to use a similar process to create a probabilistic version of maximum-margin Markov networks like the M^3N? Answering this question led to the creation of the Maximum Entropy Discrimination Markov Networks (MaxEnDNet or MEDN) by [5]. A similar design principle was used to formulate the learning problem of the structured maximum entropy discrimination (SMED) model:

$$\begin{aligned} \min_{p(\boldsymbol{w}), \xi} \quad & KL[p(\boldsymbol{w})||p_0(\boldsymbol{w})] + U(\xi) \\ \text{s.t.} \quad & p(\boldsymbol{w}) \in \mathcal{F}_1, \xi_i \geq 0, \forall i. \end{aligned} \tag{24}$$

In this case, instead of learning $\boldsymbol{w}F(\boldsymbol{x})$, the structured features are used $[\boldsymbol{w}^{\mathrm{T}} F(\boldsymbol{x}, \boldsymbol{y})]$. The constraint is defined over the space of all possible distributions, and each of these must respect the expected margin constraint:

$$\mathcal{F}_1 = \left\{ p(\boldsymbol{w}) : \int p(\boldsymbol{w}) [\Delta F_i(\boldsymbol{y}; \boldsymbol{w}) - \Delta \ell_i(\boldsymbol{y})] d\boldsymbol{w} \geq -\xi_i, \forall i, \forall \boldsymbol{y} \neq \boldsymbol{y}^i \right\} \tag{25}$$

The resulting posterior distribution can then be used in a structured SVM fashion:

$$h_1[\boldsymbol{x}; p(\boldsymbol{w})] = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \int p(\boldsymbol{w}) F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) d\boldsymbol{w} \tag{26}$$

Lagrangian theory could be used to find a closed-form solution for the general learning problem. A Lagrangian needs to be built with both the loss and the constraint functions and then a variational derivative is taken. The closed-form solution for the posterior distribution is as follows:

$$p(\boldsymbol{w}) = \frac{1}{Z(\alpha)} p_0(\boldsymbol{w}) \exp\left\{ \sum_{i,\boldsymbol{w}} \alpha_i(\boldsymbol{y}) \left[\Delta F_i(\boldsymbol{y}; \boldsymbol{w}) - \Delta \ell_i(\boldsymbol{y}) \right] \right\} \tag{27}$$

Notice that the solution is not strictly Bayesian in the sense that the factor that multiplies the prior is not the likelihood of the observations. Instead, it is defined in terms of the margins over the training data. The resulting posterior distribution thus belongs to the exponential family and corresponds to a weighted sum of the support vectors. The weights α_i are obtained by solving the following dual optimization problem, which corresponds to solving a SVM problem where only a few α_i are non-zero, as given by the complementary slackness effect of the KKT conditions:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & -\log Z(\boldsymbol{\alpha}) - U^*(\boldsymbol{\alpha}) \\ \text{s.t.} & \alpha_i(\boldsymbol{y}) \geq 0, \forall i, \forall \boldsymbol{y}. \end{array} \tag{28}$$

$U^*(\cdot)$ is the conjugate of the $U(\cdot)$, i.e., $U^*(\alpha) = \sup_{\xi}[\sum_{i,y} \alpha_i(y)\xi_i - U(\xi)]$. Different instances of MaxEnDNet can be used by selecting the function $F(\cdot)$ and the desired prior $p_0(\boldsymbol{w})$.

4.2.1 Gaussian MaxEnDNet

If we assume a dot product function as in the structured SVM $F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} f(\boldsymbol{x}, \boldsymbol{y})$, $U(\xi) = C \sum_i \xi_i$, and a standard Gaussian prior $p_0(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, I)$, we obtain the following solution to the posterior distribution:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mu_{\boldsymbol{w}}, I) \tag{29}$$

Here $\mu_{\boldsymbol{w}} = \sum_{i,\boldsymbol{w}} \alpha_i(\boldsymbol{y}) \Delta f_i(\boldsymbol{y})$. The weights α_i are computed in the same way as in the structured SVM M^{3}N. The predictive rule becomes:

$$h_1(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \boldsymbol{w}^{\mathrm{T}} f(\boldsymbol{x}, \boldsymbol{y}) \tag{30}$$

Therefore, the Gaussian MaxEnDNet is, in essence, a probabilistic version of M^{3}N that computes the posterior probability of the weights $\boldsymbol{w}$ instead of just their mean $\boldsymbol{w}^*$. This means that MaxEnDNet is a more general framework that can take the form of proven models like the M^{3}N but offers the advantages of being probabilistic.

These advantages include the guarantees offered by the possibility of obtaining a generation bound on the error using the PAC-Bayesian theory, the ability to being generalized to incorporate hidden variables and structures given the probabilistic definition of the model, and, similarly, the ability to deal with partially labeled data.

4.2.2 Laplace MaxEnDNet

A Laplacian prior can be used instead of a Gaussian one:

$$p_0(\boldsymbol{w}) = \prod_{k=1}^{K} \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w^k|} = \left(\frac{\sqrt{\lambda}}{2} \right)^K e^{-\sqrt{\lambda}\|\boldsymbol{w}\|} \tag{31}$$

This version of MaxEnDNet is called Laplace MaxEnDNet [5]. The learning problem becomes:

$$\begin{aligned} \min_{p(\boldsymbol{w},\xi),\xi} \quad & \sqrt{\lambda}\sum_{k=1}^{K}\left(\sqrt{\mu_k^2+\frac{1}{\lambda}}-\frac{1}{\sqrt{\lambda}}\log\frac{\sqrt{\lambda\mu_k^2+1}+1}{2}\right)+C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \mu^{\mathrm{T}}\Delta f_i(\boldsymbol{y}) \geq \Delta\ell_i(\boldsymbol{y})-\xi_i; \xi_i \geq 0, \forall i, \forall y \neq y^i \end{aligned} \tag{32}$$

The spiked nature of the prior has a regularization effect over the weights $\boldsymbol{w}$, resulting in an l_1 shrunk version of M^{3}N. The parameter λ controls the values of the coefficients; as λ increases, the model becomes more regularized. This can be seen in the means of the posterior parameter vector $\boldsymbol{w}$:

$$\forall k, \langle w_k \rangle_p = \frac{2\eta_k}{\lambda - \eta_k^2} \tag{33}$$

The vector $\boldsymbol{\eta}$ is a linear combination of the "support vectors":

$$\boldsymbol{\eta} = \sum_{\alpha} \alpha_i(\boldsymbol{y})\Delta f_i(\boldsymbol{y}) \tag{34}$$

The KL norm imposed by this model is parameterized by λ, and can be adjusted to represent any intermediate step between an equivalent l_1 norm and an l_2 norm as shown in Figure 7 for a two-dimensional case. The smooth transition between these two types of regularization gives the Laplace MaxEnDNet great flexibility.

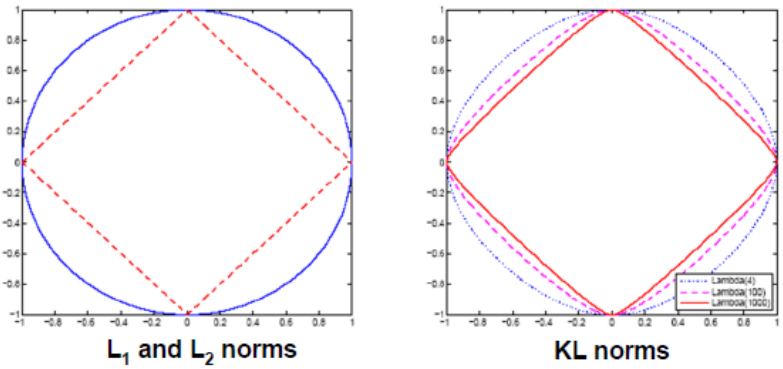


Figure 7: The KL norm on the right allows for a smooth transition between the l_1 and l_2 norms (left) depending on the value of the parameter λ.

The Laplace MaxEnDNet is a model that is primal sparse due to the Laplace shrinkage effect. This means that the weight vector $\boldsymbol{w}$ is forced to have a big number of coefficients equal to zero. Additionally, the model is also dual sparse, as the M^{3}N, thanks to being a maximum-margin Markov network. This means that the posterior is decided by a controlled number of support vectors, an aspect that efficiently selects only the most important features [6].

Figure 8 shows the comparison in accuracy of the Laplace MaxEnDNet in the problem of Optical Character Recognition (OCR). The algorithm is compared to different versions of Conditional Random Fields (CRF) and to M^{3}N. The algorithms shown include regularized versions of CFRs with both an l_1 norm and an l_2 norm, and an l_1 version of M^{3}N [7]. M^{3}N original formulation already uses an l_2 norm. The comparison shows that Laplace MaxEnDNet achieves the smallest error rate in all of the four scenarios.

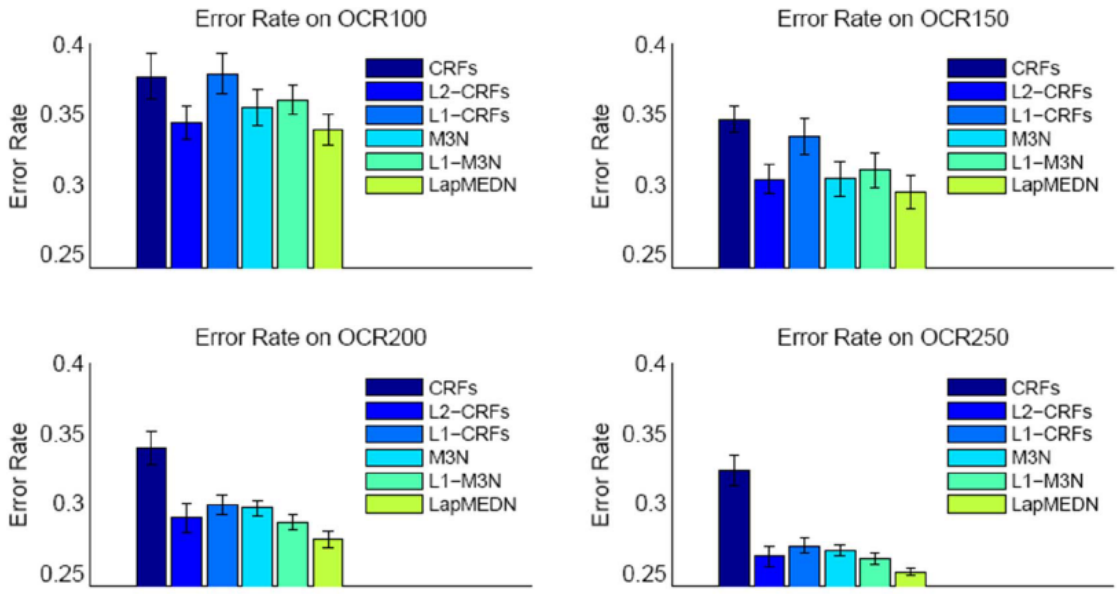


Figure 8: Error rate of different classification algorithms on four different randomly constructed Optical Character Recognition (OCR) problems.

Figure 9 shows the actual features used for generating the model and the features selected by each of the algorithms. We can appreciate the improved response of the Laplace MaxEnDNet over the competing algorithms through its ability to efficiently assign weights of zero to irrelevant features while maintaining a dominant performance.

References

[1] V. N. Vapnik., "The nature of statistical learning theory," 1995.

[2] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[3] Tommi Jaakkola, Marina Meila, and Tony Jebara, "Maximum entropy discrimination," 1999.

[4] Jun Zhu, Eric P Xing, and Bo Zhang, "Partially observed maximum entropy discrimination markov networks," in *NIPS*, 2008, pp. 1977–1984.

[5] Jun Zhu and Eric P Xing, "Maximum entropy discrimination markov networks," *The Journal of Machine Learning Research*, vol. 10, pp. 2531–2569, 2009.

[6] Jun Zhu and Eric P. Xing, "On primal and dual sparsity of markov networks," in *In ICML*, 2009.

[7] Jun Zhu, Eric P. Xing, and Bo Zhang, "Primal sparse max-margin markov networks," in *In International Conference on Knowledge Discovery and Data Mining (KDD*, 2009.

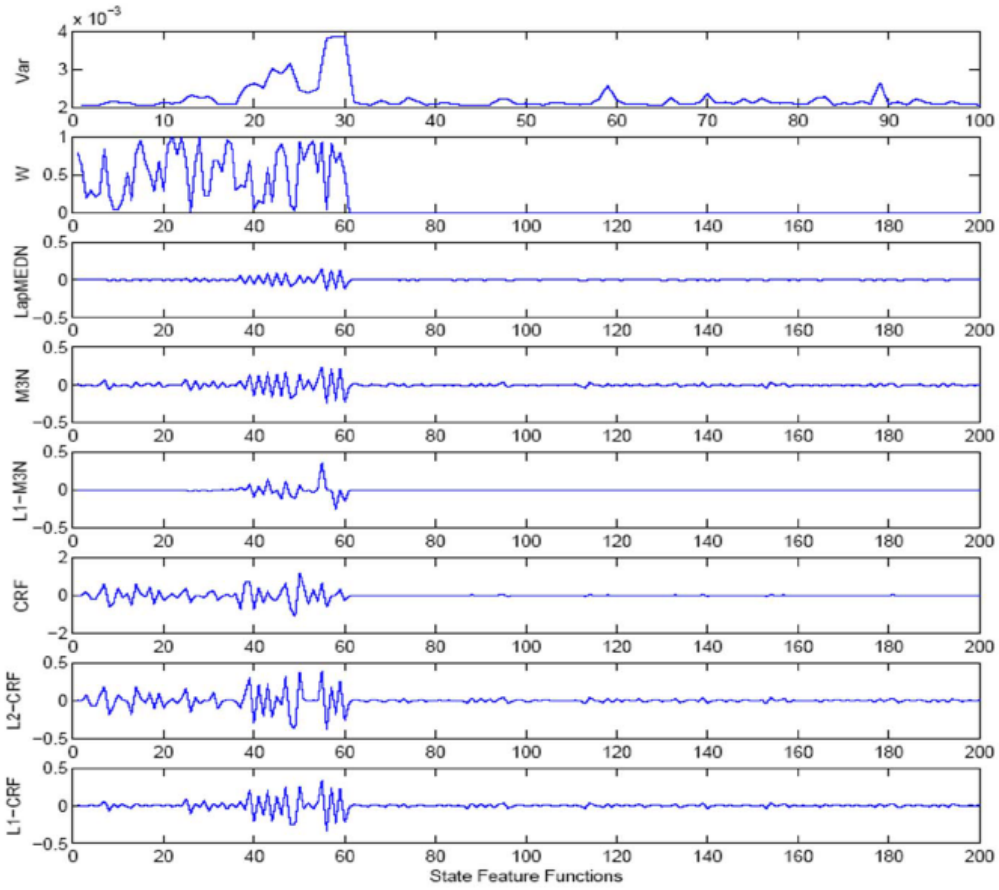


Figure 9: Feature selection in the OCR problem. Var and W at the top two panels are the parameters of the model used to generate the data; note that most of the features do not contribute to the classification problem but can lead to overfitting by non-regularized models. The other panels show the features selected by each of the competing algorithms.