Journal of Law, Governance and Public Administration

https://www.harrisonacademicpress.com/index.php/JLGPA

Vol 1, Issue 1, 2026

Learning Generative Models via Discriminative Approaches

*Mrs.Padma Bhargavi1, Associate Professor, Department of ECE, Mallareddy Institute of Engineering and Technology, Maisammaguda, Secundrabad E-Mail:baruu.s@gmail.com
Mr.P SHEKER2, Assistant Professor, Department of ECE, St.Martin's Engineering College, Dulapally,Secundrabad. E-Mail: pshekerece@smec.ac.in
Mrs.Jansi Atluri3, Assistant Professor, Department of ECE, Mallareddy College of Engineering for Women, Maisammaguda, Secudrabad. E-Mail:jhansi.atluri@gmail.com

Abstract

Generative model learning is one of the key problems in machine learning and computer vision. Currently the use of generative models is limited due to the difficulty in effectively learning them. A new learning framework is proposed in this paper which progressively learns a target generative distribution through discriminative approaches. This framework provides many interesting aspects to the literature. From the **generative model** *side: (1) A reference distribution is used to assist the learning process, which removes the need for a sampling processes in the early stages. (2) The classification power of discriminative approaches, e.g. boosting, is directly utilized. (3) The ability to select/explore features from a large candidate pool allows us to make nearly no assumptions about the training data. From the* **discriminative model** *side: (1) This framework improves the modeling capability of discriminative models. (2) It can start with source training data only and gradually "invent" negative samples. (3) We show how sampling schemes can be introduced to discriminative models. (4) The learning procedure helps to tighten the decision boundaries for classification, and therefore, improves robustness. In this paper, we show a variety of applications including texture modeling and classification, non-photorealistic rendering, learning image statistics/denoising, and face modeling. The framework handles both homogeneous patterns, e.g. textures, and inhomogeneous patterns, e.g. faces, with nearly an identical parameter setting for all the tasks in the learning stage.*

1. Introduction

Generative model learning is one of the key problems in machine learning and computer vision. Generative models are desirable as they capture the underlying generation process of a data population of interest. In the context of image analysis, such a data population might be a texture or an object category. However, it is usually very hard to learn a generative model for data of high dimension since the structure of the data space is largely unknown. A collection of data samples (ensemble) may lie on a very complex manifold. Existing generative models include principle component analysis (PCA) [20], independent component analysis (ICA) [12], and mixture of Gaussians models [4]. These models assume simple formation of the data, and they have difficulty in modeling complex patterns of irregular distributions. General pattern theory [9], though nice in principle, requires defining complex operators and rules; how amenable it is to modeling a wide class of image patterns and shapes is still unclear.

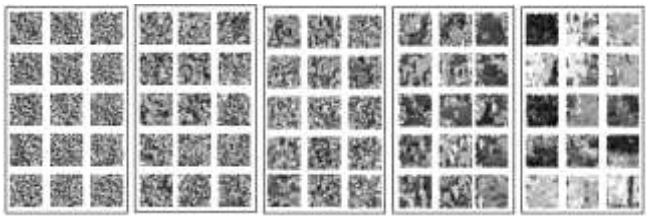


Figure 1. Image patches sampled at different stages by our algorithm for learning natural image statistics.

Discriminative models, often referred to as classification approaches, have been widely used in the literature. Many successful applications have been devised using methods like support vector machines (SVM) [24] or boosting [6]. Though these discriminative methods have strong discrimination/classification power, their modeling capability is limited since they are focusing on classification boundaries rather than the generation process of the data. Thus, they cannot be used to create (synthesize) samples of interest. Another disadvantage of the existing discriminative models is that they often need both positive and negative training samples, though a single-class classification was proposed in [16] using special kernels. Negative samples may not be obtained easily in some situations, e.g. it is very hard to obtain negative shapes. Nevertheless, situations occur where there is still room to improve the classification result but there are no negatives to use. Recent active learning strategies [1] help this problem slightly by including human subjects in a loop.

The existing generative model learning frameworks [11, 3, 21, 23] have difficulty in capturing patterns of high complexity. In this paper, a new learning framework is proposed which progressively learns a target generative distribution via discriminative approaches. The basic idea is to use *neg-*

ative samples as 'auxiliary' variables (we call them pseudo-negatives), either bootstrapped or sampled from reference distributions, to facilitate the learning process in which discriminative models are used. Our method is different from the importance sampling strategy [17] in which a reference distribution is used for sampling.

A given a set of image patches are treated as positives samples. We have an image database (5,000 natural images) from which pseudo-negative samples are randomly selected. We then use the positives and pseudo-negatives to train a discriminative model, and recursively obtain pseudo-negative samples either by bootstrapping or by sampling. The algorithm converges when the training error is bigger than a certain threshold, indicating that pseudo-negative samples drawn from the model are similar to the input positive samples. This learning framework provides several interesting aspects to the existing generative and discriminative learning literature.

For the generative models:

1. A reference distribution (image database) is used to assist the generative model learning process. We make use of the image database for bootstrapping pseudo-negative samples which removes the need for sampling processes in the early stages (this was necessary in [11, 3, 21]).
2. The discrimination/classification power of discriminative approaches, e.g. boosting, is directly utilized.
3. The ability of selecting/exploring features from a large candidate pool allows us to make nearly no assumptions about the training data. By using both position sensitive Haar features and position insensitive histogram features, the algorithm is able to handle both homogeneous and inhomogeneous patterns.

For the discriminative models:

1. This framework largely improves the modeling capability of existing discriminative models. Despite some recent efforts in combining discriminative models in the random fields model [13], discriminative models mostly have been popular for classification.
2. Though starting from a reference distribution largely improves the efficiency of our algorithm, our learning framework also works with positive training data only, and gradually invent pseudo-negative samples. Traditional discriminative models always need both positives and negatives.
3. We discuss various sampling schemes based on the discriminative models.
4. Our model can also be viewed as a classification approach. Different generative models learned are directly comparable if they use the same reference distribution. The progressive learning procedure helps to tighten the decision boundaries for the discriminative models, and therefore, improves their robustness. (we show this in the image denoising case in the experiments, and more experiments in other domains will be carried to further illustrate this point.)

Three other existing generative models are related to our framework, namely, the induction feature model [3], the MiniMax entropy model [21], and the products of experts model (POE) [11, 2, 25]. These algorithms are somewhat similar in that they are all learning a distribution from an exponential family. A feature selection stage appears in all these methods together with a sampling step to estimate the parameters for combining the features. Our model differs from the existing generative models due to the explicit adoption of discriminative models. The feature selection and fusing strategy embedded in the boosting algorithm are more efficient than those in these generative model learning algorithms. This is due to two reasons:(1) The loss function in boosting is based on classification error. (2) Positive and negative samples are given, and no sampling process is needed in the feature selection stage. Our model is not restricted on local cliques. By using both position sensitive Haar features and position insensitive histogram features, the algorithm is shown to be very flexible and general. The contrastive divergence learning algorithms [2, 11] emphasize on using less number of sampling steps to estimate the model parameters. A large body of energy-based classification models [15] are mostly focused on discriminative models. The purpose of hybrid models in [14] is to study different priors over parameters.

Our algorithm is also closely related to the self-supervised boosting algorithm by Welling et al. [26]. However, our algorithm differs from [26] in several aspects: (1) We derive our generative models from the Bayesian theory and give convergence proof. In this paper, we use boosting as discriminative model. But any discriminative models, e.g. SVM, can be applied in our model. [26] is restricted on Boltzmann distribution and boosting algorithm. (2)Our model combines a sequence of strong classifiers, whereas [26] focuses on the feature selection for weak classifiers, which makes it more closely related to [3]. We directly use boosting for feature selection and fusion. Our model is thus faster than [26] since sampling is not needed in training each weak classifier. (3)We provide many insights from both generative and discriminative models. (4) We use existing database and bootstrapping to improve the speed which is not in [26, 21, 19].

In this paper, we show a variety of applications including texture modeling and classification, non-photo-realistic rendering, learning image statistics/denoising, and face modeling. The framework handles both homogeneous patterns, e.g. textures, and inhomogeneous patterns, e.g. faces, with nearly an identical parameter setting for all the tasks in the learning stage.

2. Generative vs. discriminative models

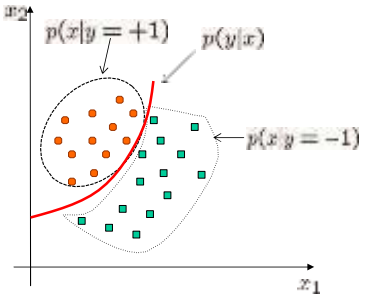


Figure 2. Illustration of generative v.s. discriminative models. Discriminative models focus on classification boundaries between the positives and negatives, whereas generative models emphasize the data generation process in each individual class.

Let x be data vector and $y \in \{-1, +1\}$ its label, indicating either a negative or a positive sample. In a multi-class problem with n classes y is in $\{1, ..., n\}$; in this paper, we focus on two-class models. Given input data point x, a discriminative model computes $p(y|x)$, the probability of x being positive or negative. Of course we only need to compute $p(y = +1|x)$ since $p(y = -1|x) = 1 - p(y = +1|x)$.

A generative model, on the other hand, often captures the generation process of x by modeling $p(x|y = +1)$ and $p(x|y = -1)$.[1] Figure (2) gives an illustration of discriminative model $p(y|x)$ and generative model $p(x|y)$. As we can see, discriminative models are mostly focused on how well they can separate the positives from the negatives. A sample far from the decision boundary in the positive region may not look like a positive sample at all. But a discriminative model will give a high probability to it being positive. Generative models try to understand the basic formation of the individual classes, and thus, carry richer information than discriminative models. Given the prior $p(y)$, one can always derive discriminative models $p(y = +1|x)$ from generative models based on Bayes rule by

$$p(y = +1|x) = \frac{p(x|y = +1)p(y = +1)}{\sum_{y \in \{-1,+1\}} p(x|y)p(y)}. \quad (1)$$

However, generative models are much harder to learn than discriminative models, and often, one makes simplified assumptions about the data formation, e.g. orthogonal basis in PCA.

It has been shown that AdaBoost algorithm and its variations [6] are approaching logistical regression [7] according to

$$p(y|x) = \frac{\exp\{\sum_{t=1}^{T} \alpha_t y h_t(x)\}}{\sum_y \exp\{\sum_{t=1}^{T} \alpha_t y h_t(x)\}}, \quad (2)$$

where h_t is a weak classifier. At each step, AdaBoost selects h_t from a set of candidate classifiers and estimates α_t by minimizing an exponential loss function.

Interestingly, generative models in [3, 21, 11] estimate a similar exponential function by

$$p(x|y = +1) = \frac{\exp\{-\sum_{t=1}^{T} \lambda_t H_t(x)\}}{\sum_x \exp\{-\sum_{t=1}^{T} \lambda_t H_t(x)\}}, \quad (3)$$

where $H_t(x)$ is a feature of x.

As we can see, both eqn. (2) and eqn. (3) have a feature selection stage and a parameter estimation procedure. However, it is much easier to learn eqn. (2) than eqn. (3) because the normalization term in the discriminative model is on $y \in \{-1, +1\}$ whereas the generative models requires integrating out over all possible x in the data space.

3. Learning framework

In this section, we show how to use discriminative models to derive generative models. For the remainder of this paper, the vector x represents an image patch. Our framework, however, is applicable to other problems such as shape, text, and medical data modeling.

3.1. From discriminative to generative models

Often, a positive class represents a pattern of interest and a negative class represents the background patterns. Thus, our goal is to learn a generative model $p(x|y = +1)$. Rearranging Eqn. (1) gives

$$p(x|y = +1) = \frac{p(y = +1|x)p(y = -1)}{p(y = -1|x)p(y = +1)} p(x|y = -1). \quad (4)$$

For notational simplicity, we assume equal priors ($p(y = +1) = p(y = -1)$).

$$p(x|y = +1) = \frac{p(y = +1|x)}{p(y = -1|x)} p(x|y = -1). \quad (5)$$

The above equation says that a generative model for the positives $p(x|y = +1)$ can be obtained from the discriminative model $p(y|x)$ and a generative model $p(x|y = -1)$ for the negatives. For clarity, we now refer to the distribution $p(x|y = -1) = p^r(x)$ as a **reference distribution** and call a set of samples drawn from $p^r(x)$ **pseudo-negatives**. We have

$$p(x|y = +1) = \frac{p(y = +1|x)}{p(y = -1|x)} p^r(x). \quad (6)$$

A trivial observation is that $p(x|y = +1) = p^r(x)$ when $p(y = +1|x) = p(y = -1|x)$. This is easy to understand. The positive and pseudo-negative samples are from the same distribution when a perfect classifier cannot tell them apart.

However, learning $p(x|y = +1)$ in eqn. (6) is a challenging task since we need pseudo-negative samples cover the entire space of x. We can only learn an approximated discriminative model $q(y|x) \sim p(y|x)$ on a given set of positives and a limited number of pseudo-negatives sampled from $p^r(x)$. Fig. (2) shows an illustration. Our basic strategy is to learn an approximated $p(x|y = +1)$ and then plugged it back into the right side of eqn (6). Since $p(x|y = +1)$ will be used to draw negatives, we write it in the form of p^r as well to make it less confusing. Next, we give detailed explanations.

Let $p_1^r(x)$ be an initial reference model, e.g., a database of natural images in which every image patch in every image is a sample. We define:

$$p_1^r(x) = \beta \frac{1}{|DB|} \sum_{x_t \in DB} \delta(x - x_t) + (1 - \beta)U(x), \quad (7)$$

where DB includes all image patches in the database, $|DB|$ is the size of the set, $U(x)$ is the uniform distribution, and

[1] In the literature, one also uses $p(x, y)$ to denote a generative model.

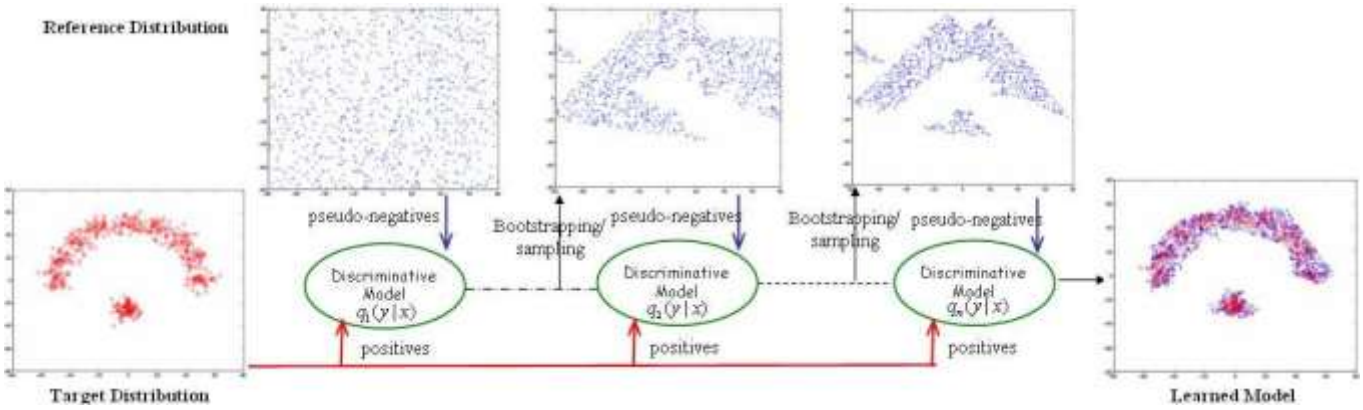


Figure 3. An illustration of the learning algorithm. The left most figure shows a target distribution from which we want to learn a generative model. The top left figure shows a reference distribution used, which is a uniform distribution. At each stage, samples are bootstrapped from the reference distribution and used as pseudo-negatives. The right most figure shows the final generative model learned. Points shown in cross are samples drawn from the final model. They are overlayed with the training set.

δ is the indicator function. In case DB is not available, we set $\theta = 0$ and $p^r_1(x) = U(x)$. (Drawing fair samples from $p^r_1(x)$ is straightforward since it is a simple mixture model. Evaluating $p^r(x)$ is more time-consuming. However, it is not necessary to compute $p^r_1(x)$ if the same reference distribution is used in 9.)

Let S^P be a set of samples from which we want to learn a generative model. We randomly draw a subset of pseudo-negatives, S^N_1, w.r.t to $p^r_1(x)$ to train a classifier q based on S^N_1 and S^P. Thus, we obtain an updated generative model $p^r_2(x)$ by

$$p^r_2(x) = \frac{1}{Z_1}\frac{q_1(y=+1|x)}{q_1(y=-1|x)} p^r_1(x), \quad (8)$$

where $Z_1 = \int \frac{q_1(y=+1|x)}{q_1(y=-1|x)} p^r_1(x)dx$. Note that $Z_1 = 1$ if $q_1(y|x) = p^r_1(y|x)$. We compute Z_1 using Monte Carlo technique [17] based on S^N_1, which is a set of fair samples. (This is an approximation and in practice, it is not critical for the overall model.)

Given $p^r_2(x)$, if we plug it back to the right side of eqn. (8) to replace $p^r_1(x)$, we can compute $p^r_3(x)$ in an identical manner. Repeating the procedure n times, we get

$$p^r_{n+1}(x) = \prod_{k=1}^{n} \frac{1}{Z_k}\frac{q_k(y=+1|x)}{q_k(y=-1|x)} p^r_1(x), \quad (9)$$

where $q_k(y=+1|x)$ is the discriminative model learned by the kth classifier. If a boosting algorithm is adopted, eqn. (9) becomes

$$p^r_{n+1}(x) = \prod_{k=1}^{n} \frac{1}{Z_k} \exp\{2\alpha^{kt} \sum_t h^{kt}(x)\} p^r_1(x). \quad (10)$$

Our goal is to have

$$p^r_{n+1}(x) \to p(x|y=+1),$$

when the set of pseudo-negatives sampled from $p^r_{n+1}(x)$ are indistinguishable from the training positive set.

Theorem 1 *$KL[p^+(x)||p^r_{n+1}(x)] \le KL[p^+(x)||p^r_n(x)]$ where KL denotes the Kullback-Leibler divergence between two distributions, and $p(x|y=+1) = p^+(x)$.*

Proof:

$$\begin{aligned} & KL[p^+(x)||p^r_n(x)] - KL[p^+(x)||p^r_{n+1}(x)] \\ = & \int p^+(x) \log \frac{1}{Z_n}\frac{q(y=+1|x)}{q(y=-1|x)} p^r_n(x)\,dx - \int p^+(x) \log [p_n(x)]dx \\ = & \int p^+(x) \log \frac{1}{Z_n} dx + \int p^+(x) \log \frac{q(y=+1|x)}{q(y=-1|x)} dx \\ = & \log \frac{1}{Z_n} + \int p^+(x) \log \frac{q(y=+1|x)}{q(y=-1|x)} dx \ge 0 \end{aligned} \quad (11)$$

It is easy to see that $Z_k = \int \frac{q_1(y=+1|x)}{q_1(y=-1|x)} p^r_1(x)dx \le 1$ and $\int p^+(x) \log \frac{q(y=+1|x)}{q(y=-1|x)} dx \ge 0$. Each classifier in average makes a better-than-random prediction. This theorem shows that $p^r_{n+1}(x)$ converges to $p(x|y=+1)$ by combining a sequence of discriminative models, and the convergence rate depends on the classification error at each step.

We make several interesting observations from eqn. (10) w.r.t. eqn. (3) and eqn. (2). Compared to eqn. (3): the discriminative power of a strong classification model, e.g. boosting, is directly used; the $p^r_1(x)$ term can be dropped if we want to compare different learned generative models, e.g. different texture patterns, since they share the same reference distribution. Compared to eqn. (2): the negative samples are not always given and our algorithm is able to gradually invent new pseudo-negative samples. Note that we assume enough positive samples are representative for the true distribution. When the number of positives is limited, our model may overfit the data.

3.2. Sampling strategies

One key problem in our learning framework is to draw **fair** samples w.r.t. $p^r_k(x)$ as pseudo-negatives in learning. Next, we discuss five sampling strategies. A general principle is to avoid sampling x from scratch since sampling is

often a time-consuming task. It is worth to mention that some sampling strategies mentioned below, e.g. ICM and constraint sampling, will not generate fair samples. However, we found that, in practice, getting difficult samples allows the algorithm converge faster than fair samples. We will study more efficient sampling methods in the future.

Bootstrapping

At early stages of the learning process, when k is small, we bootstrap pseudo-negatives directly from the existing image database. This is similar to the cascade strategies used in [22], except that we are using a soft probability here. Fig. (6) shows some pseudo-negative samples bootstrapped from a database at different stages. As we can see, pseudo-negatives become increasingly similar to the training positives. After several rounds, all the samples in the database receive a low probability. We are forced to use a sampling scheme to invent more pseudo-negatives.

Gibbs sampling

The objective of the sampling stage is to draw fair samples w.r.t. $p^r_{n+1}(x)$ in eqn. (10). In the experiments reported in this paper, each sample x is an image patch of size 23×23. To speed up the sampling process, it usually starts from pseudo-negatives used in the previous stage. For each pixel (i, j) in the image patch, we compute

$$p^r_{n+1}(x(i,j) = v, x(\Lambda/(i,j)) \mid y = +1), \forall v, \qquad (12)$$

and randomly assign value v to pixel (i, j) accordingly. Gibbs sampler [8] is used here. The potential function is based on all the weak classifiers h which make decision on both local and global information about x. Typically, several sweeps are performed to sample values for all the pixels in x.

Iterated conditional modes

We may use the Iterated Conditional Modes (ICM) [17] method to speed up the Gibbs sampling. That is, instead of sampling the value for each $x(i, j)$ according to eqn. (12), we directly choose the value which maximizes the probability. In practice, we run one sweep of Gibbs sampling followed by $4 - 5$ sweeps of ICM.

Constraints based sampling

The above two sampling schemes need to sample every pixel in x for several sweeps. On the other hand, each weak classifier h_{kt} in eqn. (10) acts as a constraint, and the combination of all the hs decide the overall probability of x. Suppose each h is a real value on a filter response of x, $h = f(F(x))$, instead of performing Gibbs sampling on the x, we can treat all the hs as random variables and run Gibbs sampler based on eqn. (10). Once the values of all the hs are obtained, we use least-square to obtain x from $\mathbf{F} \cdot x = \mathbf{f}^{-}(\mathbf{h})$, where $\mathbf{F}$ denotes the liner transformations corresponding to all the hs, and $\mathbf{f}^{-}$ are inverse functions of the weak classifiers in the boosting algorithm. Together with ridge regression, we have

$$x = (\mathbf{F}^T\mathbf{F} + \lambda I)^{-}\mathbf{F}^T(\mathbf{f}^{-}(\mathbf{h})).$$

Ridge regression is used to regularize x since all hs obtained may not always be consistent with each other. Figure (10d) shows some images sampled using this method. However, this sampling method is not yet so effective because some samples satisfying the constraints of the weak classifiers may not be obtained from the closed-form solution.

Top-down guided sampling

For some regular patterns, e.g. faces, one can use a PCA model (principle component analysis) as a reference distribution. It is very fast to draw a sample out of a PCA model, and we then use Gibbs/ICM sampler to perturb the image. We quickly locate a promising sample and use Gibbs/ICM sampler to drag it to a better state in terms of $p^r_{n+1}(x)$. This works when we want to obtain a refined model for patterns roughly following a regular distribution.

3.3. Outline of the algorithm

In this section, we give the outline of our learning framework. We use the boosting algorithm as our discriminative model in the rest of this paper.

1. Our goal is to learn a generative model for a set of trainings samples, S_P.
2. Collect an image database, DB.
3. Randomly select a sub-set of samples from the database. This is our initial pseudo-negative sample set S^N_1. If a database is not available, then draw a set of samples of white noise.
4. Train a discriminative model using a boosting algorithm.
5. Bootstrap data from the database based on eqn. (10). If all the samples receive low probability, then draw samples using one of the sampling schemes discussed in Sect. (3.2).
6. Go back to step 4 until the training error for the discriminative model reaches an upper threshold.

Figure 4. Outline of the learning algorithm.

Fig. (3) shows a toy example for the learning algorithm outlined in Fig. (4). The left most figure shows the training samples. The distribution has an irregular shape and it is hard to fit a mixture of Gaussians to it. We use a uniform distribution as our initial reference model and implement a boosting algorithm (GentalBoost [7]). Features are projections to directional lines on the plane, and there are around 500 such lines. A sequence of discriminative models gradually cut out the space for the generative models. Unlike traditional PCA or mixture of Gaussians approaches, we do not need to make any assumption about the shapes of the target distribution. The algorithm utilizes the intrinsic generalization ability in boosting to achieve accuracy and robustness.

4. Experiments

We implemented a variety of applications using the learning framework introduced in this paper, including texture modeling/synthesis, texture classification, non-photo-realistic rendering, learning natural image statistics/denoising, and face modeling. To allow the framework

to deal with both homogeneous patterns e.g. textures, and inhomogeneous patterns, e.g. faces, we use two types of features. The first set of features are Haar wavelets, which are similar to those used in [22]. These Haar filters are good at capturing common components appearing at similar locations. It has been shown that the concept of texture ties directly to the histogram of Gabor responses [21]. For each image patch, we convolve it with a bank of Gabor wavelets and obtain a histogram for each filter. Typically, each histogram has 30 bins. We use each histogram bin as a feature. The the boosting algorithm weights the importance of every bin and combines them, and eventually constrains the sampled images to have similar histograms to the training images. For an image patch of size 23×23, there are around $35,000$ features including the Haars and histogram bins. Typically, we use 40 features for each boosting strong classifier. It is critical to have real-valued weak classifier in the boosting algorithm to facilitate the sampling process. We use the GentalBoost algorithm [7] in this paper. The discrete AdaBoost algorithm [6] gives hard decision boundaries for each weak classifier, and thus, it is hard to respond to small changes in the image. For all the experiments reported below, we use nearly an identical parameter setting in training. It usually takes a couple of days on a modern PC to train.

4.1. Texture modeling

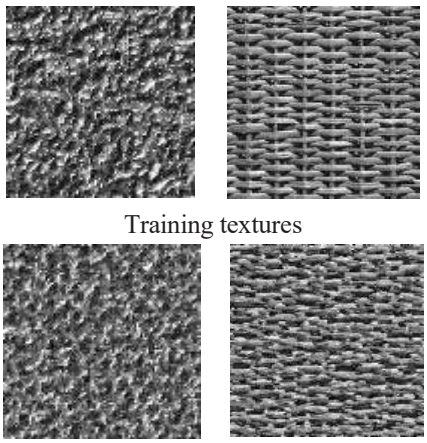


Figure 5. Examples of texture modeling. The first row shows two training images and the second row displays textures synthesized based on learned models.

An application for our framework is texture modeling. The basic learning strategy has been discussed in the beginning of this section. Fig. (6) shows some intermediate results for modeling a texture shown in Fig. (5a). There are around 25 layers of discriminative models learned and we display the pseudo-negative samples for several of them. Not surprisingly, almost all the features selected in the discriminative models are histogram features. As we can see, the pseudo-negative images look more and more like the training images after bootstrapping. The third layer shows the pseudo-negatives sampled based on eqn. (10). Interestingly, these pseudo-negatives have passed all the classification stages up to this layer, yet, they do not look like the training positive samples at all. This echoes one of the arguments made in this paper: discriminative models are focused on classifying the positives and pseudo-negatives, and they do not necessarily correspond to the underlying formation of the patterns of interest. With the pseudo-negatives gradually sampled, the model starts to converge and the sampled pseudo-negatives become increasingly faithful to the training samples. Compared to the FRAME model [21], our method is more general and flexible. It handles both homogeneous and inhomogeneous patterns. It converges faster due to the use of an image database in the early stage of the learning process and fast parameter estimation in boosting. Also, each discriminative model may combine different bins in different histograms, whereas the FRAME model has to match entire histograms one by one. In this case, our model learns a generative model for an image patch. To synthesize an image like those shown in Fig. (5), we sample patch by patch, but with an overlap of half the size to avoid boundary effect between the patches. Our applications in image analogies and image denoising below use the same strategy.

4.2. Texture classification

As stated in the paper, generative models learned separately by our framework are directly comparable if they share the same reference distribution. Also, the computing and modeling processes are directly combined, and we do not need to design additional data-driven techniques to make inference. Fig. (7) shows a classification result on two textures learned separately. We did not learn the background texture.

4.3. Image analogies

This learning framework allows us to learn very different generative models, which can be an artistic style. Fig. (8) shows an example. We use a couple of "Van Gogh" style images in [10] for training and one image is shown in Fig. (8a). We use an identical learning strategy as in texture modeling. A slight difference with the texture synthesis is that we add a likelihood term so that rendered image is slightly constrained by the original image. Fig (8c) shows a result rendered by our algorithm and Fig (8d) displays a result using the method in [10]. Unlike image analogies [10] where a pair of images are required for learning a mapping function, we directly learn a generative model (style) from a set of training images.

4.4. Learning natural image statistics

Using the same algorithm, we can learn natural image statistics. Our positive training images are from the Berkeley dataset [18]. The training process is the same as in the texture modeling and image analogies cases. However, the initial negatives samples are sampled from white noise.

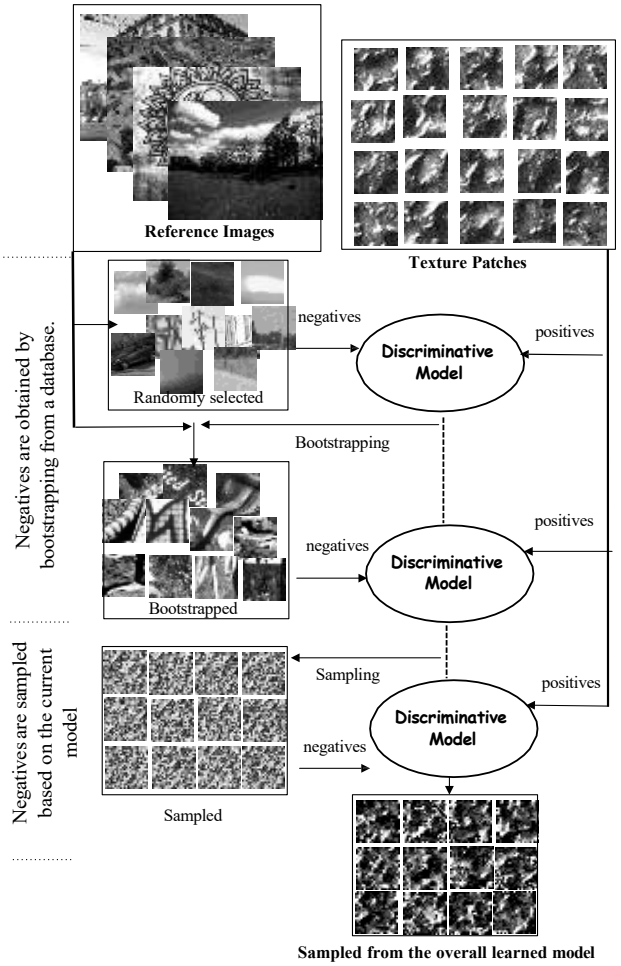
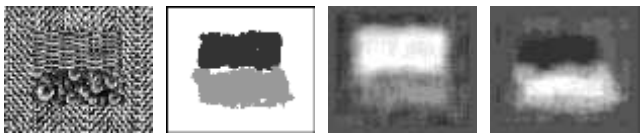


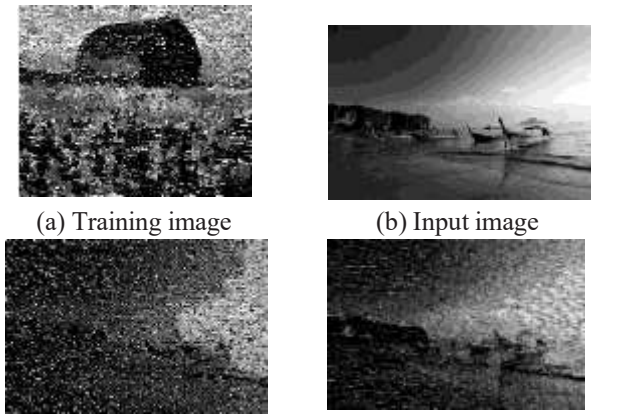
Figure 6. Illustration of the learning process for texture modeling. This is a similar figure as Fig. (3) with a real application. There are a total of 25 layers of discriminative models learned and we show several of them here. The first two set of pseudo-negatives are bootstrapped and the third and the last ones are obtained by sampling.



(a) (b) (c) (d)

Figure 7. Example of texture classification. (a) is an input image with two foreground textures and a background texture. (b) shows a classification result. (c) and (d) display the probability maps for the two foreground textures.

Fig. (1) shows patches sampled at different stages in the learning process. We can use the generative model learned from natural image statistics, as priors, to perform denoising. Fig. (9) shows an example. We also train a discriminative model based on bootstrapping procedures only (without the sampling stage). The negative images use the same images from [18] with added Gaussian noise. Fig. (9c) shows a result by discriminative model only. The result by the full model is shown in fig. (9d). This demonstrates that our model improves the robustness of discriminative models in this domain. However, our result is still a bit worse



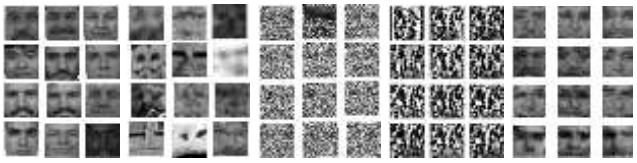
(a) Training image (b) Input image

(c) Result by our method (d) Result by image analogies

Figure 8. An example for none-photo-realist rendering. A generative model is learned based on (a). A similar style image (c) is rendered from (b). (d) shows the result in [10]

than that shown in Fig. (9e) [19], in which a generative model for learning image prior was proposed. It appears that the student-T distribution in [19] is important, and we will adopt a similar model to adapt our algorithm for image denoising in the future.

4.5. Face modeling



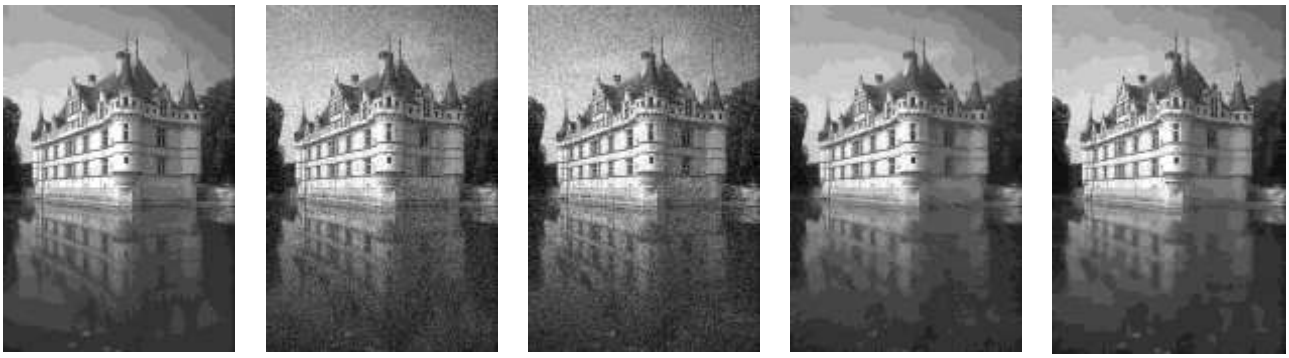
(a) (b) (c) (d) (e)

Figure 10. An example of face modeling. (a) shows some training images from the FERET dataset [5]. Some image patches bootstrapped from natural images are shown in (b). (c) displays images sampled after the bootstrapping stage. (d) shows images sampled using the constraint sampling method discussed in Sec. (3.2). Some samples drawn from the overall model are displayed in (e).

Our framework works both on homogeneous and on inhomogeneous patterns. We apply it for face modeling and fig (10) shows an example. The majority of features selected are Haars in the bootstrapping stage. Fig (10.c) shows some images sampled when all the negative samples are exhausted from an image database after 12 layers of discriminative models. These image patches, though they have passed all the discriminative models, do not look like faces. Fig (10.d) shows some images sampled using the constraint sampling method discussed in Sec. (3.2). Some face images sampled using the overall model are displayed in Fig (10.e). In face detection, this method achieves result close to the state of the art for face detection algorithms [22] on the MIT dataset. The later stage of discriminative models, however, do not further improve the detection result because they are mostly focused on capturing natural image statistics.

5. Conclusion

In this paper, we have proposed a general generative model learning framework, and it has a variety of applica-



(a) (b) (c) (d) (e)

Figure 9. An example for image denoising using learned generative model on natural image statistics. (a) is an original image. (b) shows an image by adding Gaussian nose ($\sigma = 25$) to (a). (c) is the result based on discriminative model only. (d) shows the result by our algorithm. (e) is the result by the Fields of Experts Model [19].

tions in machine learning and computer vision. Although low-level vision tasks are shown in this paper, our model can be used in high-level tasks also, e.g. shape and object modeling. For a long time there has been a debate about the use of generative and discriminative model in the literature. We provide another view to this problem in this paper and show that generative and discriminative models are not necessarily all that different.

Our method has many advantages over existing generative model methods. Most significantly, we do not need to define specific rules for different cases and the algorithm naturally works for a variety of patterns (homogeneous, inhomogeneous, and structured). Traditional discriminative model approaches do not capture the generation process of the data. Our model largely improves the modeling capability of existing discriminative methods and also improves their robustness (shown in image denoising). We will continue our research to further illustrate this point in the future.

Though we have discussed various sampling procedures in this paper, the sampling process is still slow and speed is a major bottleneck to our approach. For most the experiments shown in this paper, we use ICM with Gibbs sampling for the first sweep. Our model also assumes enough positive samples are always given. In the case with limited number of training samples, other generative models, e.g. PCA, may be better than ours. Future research is needed to study the performance of different model choices under different situations.

Acknowledgment Z.T. was funded by the NIH, Grant U54 RR021813 entitled Center for Computational Biology (CCB). We thank Yingnian Wu for many stimulating discussions, and Alan Yuille and Iasonas Kokkinos for helpful comments.

References

[1] Y. Abramson and Y. Freund, "Active Learning for Visual Object Recognition", *UCSD Report*, 2006. 1

[2] M. A. Carreira-Perpignan and G. E. Hinton, "On Contrastive Divergence Learning", *Artificial Intelligence and Statistics*, Barbados, 2005. 2

[3] S. Della-Pietra and V. Della-Pietra, and J. Lafferty, "Inducing Features of Random Fileds", *IEEE Trans. on PAMI*, vol.19, no. 4, April, 1997. 1, 2, 3

[4] R. Duda, P. Hart, and D. Stork, *Pattern Classification and Scene Analysis*, second edition, ohn Wiley & Sons, 2000. 1

[5] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms", *Image and Vision Computing J*, vol. 16, no. 5, 1998. 7

[6] Y. Freund and R. Schapire, "A Decision-theoretic Generalization of On-line Learning And An Application to Boosting", *J. of Comp. and Sys. Sci.*, 55(1), 1997. 1, 3, 6

[7] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", Dept. of Stat., Stanford U. Te. Rep. 1998. 3, 5, 6

[8] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. PAMI*, vol. 6, pp. 721-741, Nov. 1984. 5

[9] U. Grenander, Y. Chow, and D.Keenan. *HANDS: A pattern theoretic study of biological shapes*. Springer-Verlag, 1990. 1

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, D. H. Salesin, "Image Analogies", *Proc. of SIGGRAPH*, Los Angeles, Aug., 2001. 6, 7

[11] G. Hinton, "Training products of experts by minimizing contrastive divergence", *Neural Comp.*, 14(7): 1771-1800, 2002. 1, 2, 3

[12] A. Hyvarinen, J. Karhunen, E. Oja, "Independent Component Analysis", John Wiley and Sons, 2001. 1

[13] S. Kumar and M. Hebert, "Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification", *Proc. of ICCV*, 2003. 2

[14] J A Lasserre, C M Bishop, T P Minka, "Principled Hybrids of Generative and Discriminative Models",*Proc. of CVPR*, 2006. 2

[15] Y. LeCun, "A Tutorial on Energy-Based Learning",*Predicting Strutured Data*, MIT Press 2006) 2

[16] L. M. Manevitz and M. Yousef, "One-class SVMs for Document Classification", *J. of Mach. Learn. Res.*, vol.2, March, 2002. 1

[17] J.S. Liu, *Monte Carlo strategies in scientific computing*, Springer-Verlag NY INC. 2001. 2, 4, 5

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," *Proc. ICCV*, vol. 2, pp. 416-423, 2001. 6, 7

[19] S. Roth and M. J. Black, "Fields of experts: a framework for learning image priors", *Proc. of CVPR*, 2005. 2, 7, 8

[20] M. Turk and A. Pentland, "Face recognition using eigenfaces," *Proc. CVPR*, Hawaii, 1991. 1

[21] S. C. Zhu, Y.N. Wu and D.B. Mumford, "Minimax Entropy Principle and Its Applications to Texture Modelin", *Neural Computation* Vol. 9, no 8, pp 1627-1660, Nov. 1997. 1, 2, 3, 6

[22] P. Viola and M. Jones, "Fast Multi-view Face Detection", *Proc. of CVPR*, 2001. 5, 6, 7

[23] Y. N. Wu, S. C. Zhu and X. W. Liu, "Equivalence of Julesz Ensemble and FRAME models", *Int'l J. of Comp. Vis.* 38(3), 247-265, July, 2000 1

[24] V. Vapnik, "Statistical Learning Theory". Wiley-Interscience, 1998. 1

[25] M. Welling, G. Hinton, and S. Osindero, "Learning Sparse Topographic Representations with Products of Student-t Distributions", *NIPS*, 2002. 2

[26] M. Welling, R. Zemel, G. Hinton, "Self-Supervised Boosting", *NIPS*, 2002. 2