Loss Functions for Discriminative Training of Energy-Based Models.

Yann LeCun and Fu Jie Huang
The Courant Institute, New York University
{yann,jhuangfu}@cs.nyu.edu
http://yann.lecun.com

Abstract

Probabilistic graphical models associate a probability to each configuration of the relevant variables. *Energy-based models* (EBM) associate an energy to those configurations, eliminating the need for proper normalization of probability distributions. Making a decision (an inference) with an EBM consists in comparing the energies associated with various configurations of the variable to be predicted, and choosing the one with the smallest energy. Such systems must be trained discriminatively to associate low energies to the desired configurations and higher energies to undesired configurations. A wide variety of loss function can be used for this purpose. We give sufficient conditions that a loss function should satisfy so that its minimization will cause the system to approach to desired behavior. We give many specific examples of suitable loss functions, and show an application to object recognition in images.

1 Introduction

Graphical Models are overwhelmingly treated as probabilistic generative models in which inference and learning are viewed as probabilistic estimation problems. One advantage of the probabilistic approach is *compositionality*: one can build and train component models separately before assembling them into a complete system. For example, a Bayesian classifier can be built by assembling separately-trained generative models for each class. But if a model is trained discriminatively *from end to end* to make decisions, mapping raw input to ultimate outputs, there is no need for compositionality. Some applications require hard decisions rather than estimates of conditional output distributions. One example is mobile robot navigation: once trained, the robot *must* turn left or right when facing an obstacle. Computing a distribution over steering angles would be of little use in that context. The machine should be trained from end-to-end to approach the best possible decision in the largest range of situations.

Another implicit advantage of the probabilistic approach is that it provides well-justified loss functions for learning, e.g. maximum likelihood for generative models, and max conditional likelihood for discriminative models. Because of the normalization, maximizing the likelihood of the training samples will automatically decrease the likelihood of other points, thereby driving machine to approach the desired behavior. The downside is that the negative log-likelihood is *the only well-justified loss functions*. Yet, approximating a distribution over the entire space by maximizing likelihood may be an overkill when the ultimate goal is merely to produce the right decision.

We will argue that using proper probabilistic models, because they must be normalized, considerably restricts our choice of model architecture. Some desirable architectures may be difficult to normalize (the normalization may involve the computation of intractable partition functions), or may even be non-normalizable (their partition function may be an integral that does not converge).

This paper concerns a more general class of models called *Energy-Based Models* (EBM). EBMs associate an (un-normalized) energy to each configuration of the variables to be modeled. Making an inference with an EBM consists in searching for a configuration of the variables to be predicted that minimizes the energy, or comparing the energies of a small number of configurations of those variables. EBMs have considerable advantages over traditional probabilistic models: (1) There is no need to compute partition functions that may be intractable; (2) because there is no requirement for normalizability, the repertoire of possible model architectures that can be used is considerably richer.

Training an EBM consists in finding values of the trainable parameter that associate *low energies* to “desired” configurations of variables (e.g. observed on a training set), and *high energies* to “undesired” configurations. With properly normalized probabilistic models, increasing the likelihood of a “desired” configuration of variables will automatically decrease the likelihoods of other configurations. With EBMs, this is not the case: making the energy of desired configurations low may not necessarily make the energies of other configurations high. Therefore, one must be very careful when designing loss functions for EBMs [1].

[1]it is important to note that the *energy* is quantity minimized

We must make sure that the loss function we pick will effectively drive our machine to approach the desired behavior. In particular, we must ensure that the loss function has no trivial solution (e.g. where the best way to minimize the loss is to make the energy constant for all input/output pair). A particular manifestation of this is the so-called *collapse problem* that was pointed out in some early works that attempted to combined neural nets and HMMs [7, 2, 8].

This energy-based, end-to-end approach to learning has been applied with great success to sentence-level handwriting recognition in the past [10]. But there has not been a general characterization of "good" energy functions and loss functions. The main point of this paper is to give sufficient conditions that a discriminative loss function must satisfy, so that its minimization will carve out the energy landscape in input/output space in the right way, and cause the machine to approach the desired behavior. We then propose a wide family of loss functions that satisfy these conditions, independently of the architecture of the machine being trained.

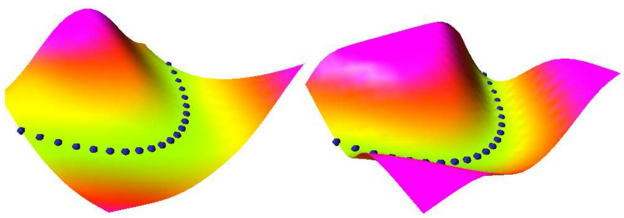

Figure 1: Two energy surfaces in X, Y space obtained by training two neural nets to compute the function $Y = X^2 - 1/2$. The blue dots represent a subset of the training samples. In the left diagram, the energy is quadratic in Y, therefore its exponential is integrable over Y. This model is equivalent to a probabilistic Gaussian model of $P(Y|X)$. The right diagram uses a non-quadratic saturated energy whose exponential is not integrable over Y. This model is not normalizable, and therefore has no probabilistic counterpart.

2 Energy-Based Models

Let us define our task as one of predicting the best configuration of a set of variables denoted collectively by Y, given a set of observed (input) variables collectively denoted by X. Given an observed configuration for X, a probabilistic model (e.g. a graphical model) will associate a (normalized) probability $P(Y|X)$ to each possible configuration of Y. When a decision must be made, the configuration of Y that maximizes $P(Y|X)$ will be picked.

An *Energy-Based Model* (EBM) associates a scalar *energy* $E(W, Y, X)$ to each configuration of X, Y. The family of possible energy functions is parameterized by a parameter vector W, which is to be learned. One can view this energy function as a measure of "compatibility" between the values of Y and X. Note that there is no requirement for normalization.

The *inference* process consists in clamping X to the observed configuration (e.g. an input image for image classification), and searching for the configuration of Y in a set $\{Y\}$ that minimizes the energy. This optimal configuration is denoted $\check{Y}$: $\check{Y} = \text{argmin}_{Y \in \{Y\}} E(W, Y, X)$. In many situations, such as classification, $\{Y\}$ will be a discrete set, but in other situations $\{Y\}$ may be a continuous set (e.g. a compact set in a vector space). This paper will not discuss how to perform this inference efficiently: the reader may use her favorite and most appropriate optimization method depending upon the form of $E(W, Y, X)$, including exhaustive search, gradient-based methods, variational methods, (loopy) belief propagation, dynamic programming, etc.

Because of the absence of normalization, EBMs should only be used for *discrimination* or *decision* tasks where only the *relative energies* of the various configurations of Y for a given X matter. However, if $\exp(-E(W, Y, X))$ is integrable over Y, for all X and W, we can turn an EBM into an equivalent probabilistic model by posing:

$$P(Y|X, W) = \frac{\exp(-\beta E(W, Y, X)}{\int_y \exp(-\beta E(W, y, X))}$$

where β is an arbitrary positive constant. The normalizing term (the denominator) is called the *partition function*. However, the EBM framework gives us more flexibility because it allows us to use energy functions whose exponential is not integrable over the domain of Y. Those models have no probabilistic equivalents.

Furthermore, we will see that training EBMs with certain loss functions circumvents the requirement for evaluating the partition function and its derivatives, which may be intractable. Solving this problem is a major issue with probabilistic models, if one judges by the considerable amount of recent publications on the subject.

Probabilistic models are generally trained with the maximum likelihood criterion (or equivalently, the negative log-likelihood loss). This criterion causes the model to approach the conditional density $P(Y|X)$ over the entire domain of Y for each X. With the EBM framework, we allow ourselves to devise loss functions that merely cause the system to make the best decisions. These loss functions are designed to place $\min_{Y \in \{Y\}} E(W, Y, X)$ near the desired Y for each X. This is a considerably less complex and less constrained problem than that of estimating the "correct" conditional density over Y for each X. To convince ourselves of this, we can note that many different energy functions may have minima at the same Y for a given X, but only one of those (or a few) maximizes the likelihood. For example, figure 1 shows two energy surfaces in X, Y space. They were obtained by training two neural nets (denoted $G(W, X)$) to approximate the function $Y = X^2 - 1/2$. In the left diagram, the energy $E(W, Y, X) = (Y - G(W, X))^2$ is quadratic in Y, there-

during inference, while the *loss* is the quantity minimized during learning

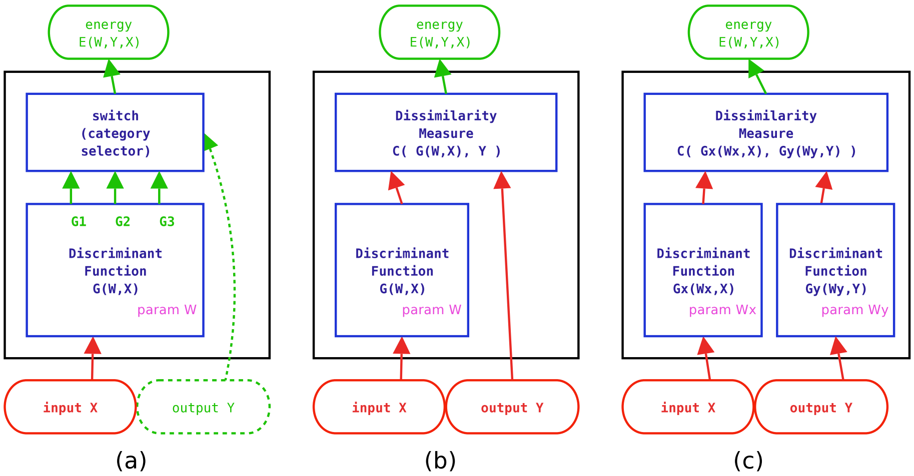

Figure 2: Examples of EBMs. (a) switch-based classifier; (b) a regressor; (c) constraint satisfaction architecture. Multidimensional variables are in red, scalars in green, and discrete variables in green dotted lines.

fore its exponential is integrable over Y. This model is equivalent to a probabilistic Gaussian model for $P(Y|X)$. The right diagram uses a non-quadratic saturated energy $E(W,Y,X) = \tanh\left((Y - G(W,X))^2\right)$ whose exponential is not integrable over Y. This model is not normalizable, and therefore has no probabilistic counterpart, yet it fulfills our desire to produce the best Y for any given X.

2.1 Previous work

Several authors have previously pointed out the shortcomings of normalized models for discriminative tasks. Bottou [4] first noted that discriminatively trained HMMs are unduly restricted in their expressive power because of the normalization requirements, a problem recently named "label bias" in [9]. To alleviate this problem, late normalization schemes for un-normalized discriminative HMMs were proposed in [6] and [10]. Recent works have revived the issue in the context of sequence labelling [5, 1, 14].

Some authors have touted the use of various non-probabilistic loss functions, such as the Perceptron loss or the maximum margin loss, for training decision-making systems [7, 10, 5, 1, 14]. However, the loss functions in these systems are intimately linked to the underlying architecture of the machine being trained. Some loss functions are incompatible with some architectures and may possess undesirable minima. The present paper gives conditions that "well-behaved" loss functions should satisfy.

Teh et al. [15] have introduced the term "Energy-Based Model" in a context similar to ours, but they only considered the log-likelihood loss function. We use the term EBM in a slightly more general sense, which include the possibility of using other loss functions. Bengio et al. [3] describe an energy-based language model, but they do not discuss the issue of loss functions.

2.2 Examples of EBMs

EBM for Classification: Traditional multi-class classifiers can be viewed as particular types of EBMs whose architecture is shown in figure 2(a). A parameterized discriminant function $G(W,X)$ produces an output vector with one component for each of the k categories $(G_0, G_1..., G_{k-1})$. Component G_i is interpreted as the energy (or "penalty") for assigning X to the i-th category. A discrete switch module selects which of the components is connected to the output energy. The position of the switch is controlled by the discrete variable Y, which is interpreted as the category. The output energy is equal to $E(W,Y,X) = \sum_{i=0}^{k-1} \delta(Y - i)G(W,X)_i$, where $\delta(Y-i)$ is equal to 1 for $Y = i$ and 0 otherwise (Kronecker function), and $G(W,X)_i$ is the i-th component of $G(W,X)$. Running the machine consists in finding the position of the switch (the value of Y) that minimizes the energy, i.e. the position of the switch that selects the smallest component of $G(W,X)$.

EBM for Regression: A regression function $G(W,X)$ with the squared error loss (e.g. a traditional neural network) is a trivial form of minimum energy machine (see figure 2(b)). The energy function of this machine is defined as $E(W,Y,X) = \frac{1}{2}||G(W,X) - Y||^2$. The value of Y that minimizes E is simply equal to $G(W,X)$. Therefore, running such a machine consists simply in computing $G(W,X)$ and copying the value into Y. The energy is then zero. This architecture can be used for classification by simply making $\{Y\}$ a discrete set (with one element for each category).

EBM for Constraint Satisfaction: sometimes, the dependency between X and Y cannot be expressed as a function that maps X's to Y's (consider for example the constraint $X^2 + Y^2 = 1$). In this case, one can resort to modeling "constraints" that X and Y must satisfy. The energy function measures the price for violating the constraints. An example architecture is shown in figure 2(c). The energy is $E(W,Y,X) = C(G_x(W_x,X), G_y(W_y,Y))$, where G_X

and G_y are functions to be learned, and $C(a, b)$ is a dissimilarity measure (e.g. a distance).

2.3 Deterministic Latent Variables

Many tasks are more conveniently modeled by architectures that use *latent variables*. Deterministic latent variables are extra variables (denoted by Z) that influence the energy, and that are not observed. During an inference, the energy is minimized over Y and Z:

$$(\check{Y}, \check{Z}) = \text{argmin}_{Y \in \{Y\},\ Z \in \{Z\}} E(W, Y, Z, X)$$

By simply redefining our energy function as:

$$\check{E}(W, Y, X) = \min_{Z \in \{Z\}} E(W, Y, Z, X)$$

We can essentially ignore the issue of latent variables.

Latent variables are very useful in situations where a hidden characteristic of the process being modeled can be inferred from observations, but cannot be predicted directly. This occurs for example in speech recognition, handwriting recognition, natural language processing, and biological sequence analysis, and other sequence labeling tasks where a *segmentation* must be performed simultaneously with the recognition. Alternative segmentations are often represented as paths in a weighted lattice. Each path may be associated with a category [10, 5, 1, 14]. The path being followed in the lattice can be viewed as a discrete latent variable. Searching for the best path using dynamic programming (Viterbi) or approximate methods (e.g. beam search) can be seen as a minimization of the energy function with respect to this discrete variable. For example, in a speech recognition context, evaluating $\min_{Z \in \{Z\}} E(W, Y^i, Z, X^i)$ is akin to "constrained segmentation": finding the best path in the lattice that produces a particular output label Y^i.

An EBM framework with which to perform graph manipulations and search, while preserving the ability to compute partial derivatives for learning is the Graph Transformer Network model described in [10]. However, that paper only mentions two loss functions (generalized perceptron and log-likelihood), without a general discussion of how to construct appropriate loss functions.

Rather than give a detailed description of latent-variable EBM for sequence processing, we will describe an application to visual object detection and recognition. The architecture is shown in figure 3. The input image is first turned into an appropriate representation (e.g. a feature vector) by a trainable front-end module (e.g. a convolutional network, as in [11]). This representation is then matched to models of each category. Each model outputs an energy that measures how well the representation matched the category (a low energy indicates a good match, a high energy a bad match). The switch selects the best-matching category. The object models take in latent variables that may be used to represent some instantiation parameters of the objects, such as the pose, illumination, or conformation. The optimal value of those parameters for a particular input is computed as part of the energy-minimizing inference process. Section 4 reports experimental results obtained with such a system.

3 Loss Functions for EBM Training.

In supervised learning, the *training set* $\mathcal{S}$ is a set of pairs $\mathcal{S} = \{(X^i, Y^i)\ ,\ i = 1..P\}$ where X^i is an input, and Y^i is a desired output to be predicted. Practically every learning methods can be described as the process of finding the parameter $W \in \{W\}$ that minimizes a judiciously chosen *loss function* $\mathcal{L}(W, \mathcal{S})$. The loss function should be a measure of the discrepancy between the machine's behavior and the desired behavior on the training set. Well-behaved loss functions for EBMs should shape the energy landscape so as to "dig holes" at (X, Y) locations near training samples, while "building hills" at un-desired locations, particularly the ones that are erroneously picked by the inference algorithm. For example, a good loss function for the regression problem depicted in figure 1 should dig holes around the blue dots (which represent a subset of the training set, while ensuring that the surrounding areas have higher energy.

In the following, we characterize the general form of loss functions whose minimization will make the machine carve out the energy landscape in the right way so as to approach the desired behavior. We define the loss on the full training set as:

$$\mathcal{L}(W, \mathcal{S}) = R\left(\frac{1}{P}\sum_{i=1}^{P} L(W, Y^i, X^i)\right) \qquad (1)$$

where $L(W, Y^i, X^i)$ is the per-sample loss function for sample (X^i, Y^i), and R is a monotonically increasing function. Loss functions that combine per-sample losses through other symmetric n-ary operations than addition (e.g. multiplication, as in the case of likelihood-like loss functions) can be trivially obtained from the above through judicious choices of R and L. With this definition, the loss is invariant under permutations of the samples, and under multiple repetitions of the same training set. In the following we will set R to the identity function. We assume that $L(W, Y^i, X^i)$ has a lower bound over W for all Y^i, X^i.

At this point, we can note that if we minimize *any* such loss on a training set over a set of functions with finite VC-dimension, appropriate VC-type upper bounds for the expected loss will apply, ensuring the convergence of the empirical loss to the expected loss as the training set size increases. Therefore, we will only discuss the conditions under which a loss function will make the machine approach the desired behavior *on the training set*.

Sometimes, the task uniquely defines a "natural" loss function (e.g. the number of mis-classified examples), but more often than not, minimizing that function is impractical. Therefore one must resort to surrogate loss functions whose choice is up to the designer of the system. One crucial, but often neglected, question must be answered before choosing a loss function: "will minimizing the loss cause the

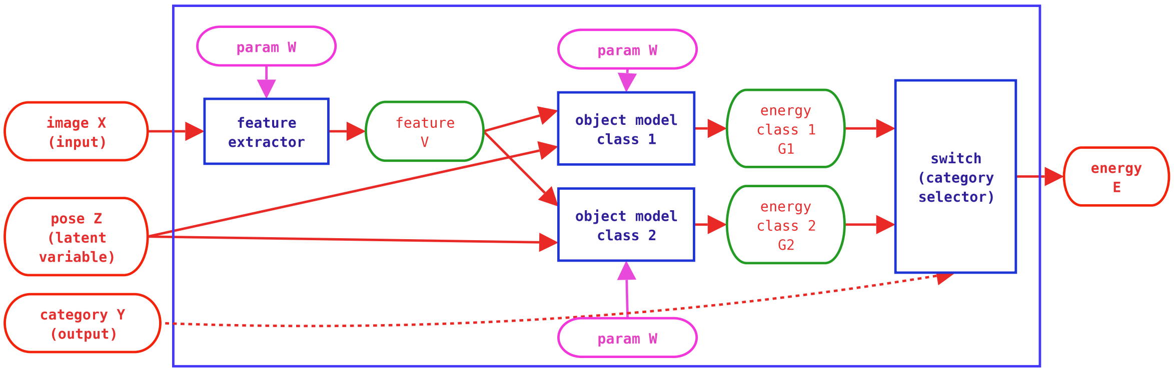

Figure 3: Example of a switch-based Energy-Based Model architecture for object recognition in images, where the pose of the object is treated as a latent variable.

learning machine to approach the desired behavior?" We will give general conditions for that.

The inference process produces the Y that minimizes $\check{E}(W, Y, X^i)$. Therefore, a well-designed loss function must drive the energy of the desired output $\check{E}(W, Y^i, X^i)$ to be lower than the energies of all the other possible outputs. Minimizing the loss function should result in "holes" at X, Y locations near the training samples, and "hills" at un-desired locations.

3.1 Conditions on the Energy

The condition for the correct classification of sample X^i is:

Condition 1 $\check{E}(W, Y^i, X^i) < \check{E}(W, Y, X^i)$, $\forall Y \in \{Y\}, Y \neq Y^i$

To ensure that the correct answer is robustly stable, we may choose to impose that the energy of the desired output be lower than the energies of the undesired outputs by a *margin* m:

Condition 2 $\check{E}(W, Y^i, X^i) < \check{E}(W, Y, X^i) - m$, $\forall Y \in \{Y\}, Y \neq Y^i$

We will now consider the case where Y is a discrete variable. Let us denote by $\bar{Y}$ the output that produces the smallest energy *while being different* from the desired output Y^i: $\bar{Y} = \text{argmin}_{Y \in \{Y\}, Y \neq Y^i} \check{E}(W, Y, X^i)$. Condition 2 can be rewritten as:

Condition 3 $\check{E}(W, Y^i, X^i) < \check{E}(W, \bar{Y}, X^i) - m$, $\bar{Y} = \text{argmin}_{Y \in \{Y\}, Y \neq Y^i} \check{E}(W, Y, X^i)$

For the continuous Y case, we can simply define $\bar{Y}$ as the lowest-energy output outside of a ball of a given radius around the desired output: $\text{argmin}_{Y \in \{Y\}, ||Y - Y^i|| > \epsilon} \check{E}(W, Y, X^i)$

3.2 Sufficient conditions on the loss function

We will now make the key assumption that L depends on X^i only indirectly through the set of energies $\{\check{E}(W, Y, X^i)$, $Y \in \{Y\}\}$. For example, if $\{Y\}$is the set of integers between 0 and $k-1$, as would be the case for the switch-based classifier with k categories shown in figure 2(a), the per-sample loss for sample (X^i, Y^i) should be of the form:

$$L(W, Y^i, X^i) = L(Y^i, \check{E}(W, 0, X^i), \ldots, \check{E}(W, k-1, X^i)) \tag{2}$$

With this assumption, we separate the choice of the loss function from the details of the internal structure of the machine, and limit the discussion to how minimizing the loss function affects the energies.

We must now characterize the form that L can take such that its minimization will eventually drive the machine to satisfy condition 3.

We must design L in such a way that minimizing it will decrease the difference $\check{E}(W, Y^i, X^i) - \check{E}(W, \bar{Y}, X^i)$, whenever $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$. In other words, whenever the difference between the energy of the incorrect answer with the lowest energy and the energy of the desired answer is less than the margin, our learning procedure should make that difference larger. We will now propose a set of sufficient conditions on the loss that guarantee this.

Since we are only concerned with how the loss influences the relative values of $\check{E}(W, Y^i, X^i)$, and $\check{E}(W, \bar{Y}, X^i)$, we will consider the shape of loss surface in the space of $\check{E}(W, Y^i, X^i)$ and $\check{E}(W, \bar{Y}, X^i)$, and view the other arguments of the loss (the energies for all the other values of Y) as parameters of that surface:

$$L(W, Y^i, X^i) = Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$$

where the parameter $[E_y]$ contains the vector of energies for all values of Y except Y^i and $\bar{Y}$.

We can now state sufficient conditions that guarantee that minimizing L will eventually satisfy condition 3. In all the sufficient conditions stated below, we assume that there exist a W such that condition 3 is satisfied for a single training example (X^i, Y^i), and that $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ is convex (convex in its 2 arguments, but not necessarily convex in W). The conditions must old for all values of $[E_y]$.

Condition 4 *The minima of $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ are in the half-plane $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.*

This condition on the loss function clearly ensures that minimizing it will drive the machine to find a solution that satisfies condition 3, if such a solution exists.

Another sufficient condition can be stated to characterize loss functions that do not have minima, or whose minimum is at infinity:

Condition 5 *the gradient of $Q_{[E_y]}(\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i))$ on the margin line $\check{E}(W, \bar{Y}, X^i) = \check{E}(W, Y^i, X^i) + m$, has a positive dot product with the direction [-1,1].*

This condition guarantees that minimizing L will drive the energies $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ toward the half-plane $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.

Yer another sufficient condition can be stated to characterize loss functions whose minima are not in the desired half-plane, but where the possible values of $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ are constrained by their dependency on W in such a way that the minimum of the loss *while satisfying the constraint* is in the desired half-plane:

Condition 6 *On the margin line $\check{E}(W, \bar{Y}, X^i) = \check{E}(W, Y^i, X^i) + m$, the following must hold:* $\left[\frac{\partial \check{E}(W,Y^i,X^i)}{\partial W} - \frac{\partial \check{E}(W,\bar{Y},X^i)}{\partial W}\right].\frac{\partial L(W,Y^i,X^i)}{\partial W} > 0$

This condition ensures that an update of the parameters W to minimize the loss will drive the energies $\check{E}(W, \bar{Y}, X^i)$ and $\check{E}(W, Y^i, X^i)$ toward the desired half-plane $\check{E}(W, \bar{Y}, X^i) < \check{E}(W, Y^i, X^i) + m$.

We must emphasize that these are only sufficient conditions. There may be legitimate loss functions (e.g. non-convex functions) that do not satisfy them, yet have the proper behavior.

3.3 Examples of Loss Functions

We can now give examples of loss functions that satisfy the above criteria, and examine whether some of the popular loss functions proposed in the literature satisfy it.

Energy Loss: The simplest and most widely used loss is the *energy loss*, which is simply of the form $L_{\text{energy}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i)$. This loss does not satisfy condition 4 or 5 in general, but there are certain forms of $\check{E}(W, Y^i, X^i)$ for which condition 6 is satisfied. For example, let us consider an energy of the form $E(W, Y, X) = \sum_{k=1}^{K} \delta(Y-k).||U^k - G(W, X)||^2$. Function $G(W, X)$ could be a neural net, on top of which are placed K radial basis functions whose centers are the vectors U^k. If the U^k are fixed and all different, then the energy loss applied to this machine fulfills condition 6. Intuitively, that is because by pulling $G(W, X)$ towards one of the RBF centers, we push it away from the others. Therefore when the energy of the desired output decreases, the other ones increase. However, if we allow the RBF centers to be learned, condition 6 is no longer fulfilled. In that case, the loss has spurious minima where all the RBF centers are equal, and the function $G(W, X)$ is constant and equal to that RBF center. The loss is zero, but the machine does not produce the desired result. Picking any combination of loss and energy that satisfy any of the conditions 4, 5, or 6 solves this collapse problem.

Generalized Perceptron Loss: We define the generalized Perceptron loss for training sample (X^i, Y^i) as:

$$L_{\text{ptron}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - \min_{Y \in \{Y\}} \check{E}(W, Y, X^i) \quad (3)$$

It is easy to see that with $E(W, Y^i, X^i) = -Y^i.W^T X^i$, and $\{Y\} = \{-1, 1\}$, the above loss reduces to the traditional linear Perceptron loss. The generalized perceptron loss satisfies condition 4 with $m = 0$. This loss was used by [10] for training a commercially deployed handwriting recognizer that combined a heuristic segmenter, a convolutional net, and a language model (where the latent variables represented paths in an interpretation lattice). A similar loss was studied by [5] for training a text parser. Because the margin is zero, this loss may not prevent collapses for certain architecture.

Generalized Margin Loss: A more robust version of the Perceptron loss is the *Margin Loss*, which directly uses the energy of most offending non-desired output $\bar{Y}$ in the contrastive term:

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m[\check{E}(W, Y^i, X^i) - \check{E}(W, \bar{Y}, X^i)] \quad (4)$$

where $Q_m(e)$ is any function that is monotonically increasing for $e > -m$. The traditional "hinge loss" used with kernel-based methods, and the loss used by the LVQ2 algorithm are special cases with $Q_m(e) = e + m$ for $e > -m$, and 0 otherwise. Special forms of that loss were used in [7] for discriminative speech recognition, and in [1] and [14] for text labeling. The exponential loss used in AdaBoost is a special form of equation (4) with $Q_m(e) = \exp(e)$. A slightly more general form of the margin loss that satisfies conditions 4 or 5 is given by:

$$L_{\text{gmargin}}(W, Y^i, X^i) = Q_{gm}[\check{E}(W, Y^i, X^i), \check{E}(W, \bar{Y}, X^i)] \quad (5)$$

with the condition that $\frac{\partial Q_{gm}(e1,e2)}{\partial e1} > \frac{\partial Q_{gm}(e1,e2)}{\partial e2}$ when $e1 + m > e2$. An example of such loss is:

$$L(W, Y^i, X^i) = Q^+(\check{E}(W, Y^i, X^i)) + Q^-(\check{E}(W, \bar{Y}, X^i)) \quad (6)$$

where $Q^+(e)$ is a convex monotonically increasing function, and $Q^-(e)$ a convex monotonically decreasing function such that if $\frac{dQ^+}{de}|e1 = 0$ and $\frac{dQ^-}{de}|e2 = 0$ then $e1 + m < e2$. When $\check{E}(W, Y^i, X^i)$ is akin to a distance (bounded below by 0), a judicious choice for Q^+ and Q^-

is:

$$L(W, Y^i, X^i) = \check{E}(W, Y^i, X^i)^2 + \kappa \exp(-\beta \check{E}(W, \bar{Y}, X^i)) \tag{7}$$

where κ and β are positive constants. A similar loss function was recently used by our group to train a pose-invariant face detection [12]. This system can simultaneously detect faces and estimate their pose using latent variables to represent the head pose.

Contrastive Free Energy Loss: While the loss functions proposed thus far involve only $\check{E}(W, \bar{Y}, X^i)$ in their contrastive part, loss functions can be devised to combine all the energies for all values of Y in their contrastive term:

$$\begin{aligned} L(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - \\ F(\check{E}(W, 0, X^i), \ldots, \check{E}(W, k-1, X^i)) \end{aligned} \tag{8}$$

F can be interpreted as a *generalized free energy* of the ensemble of systems with energies $\check{E}(W, Y, X^i)\ \forall Y \in \{Y\}$. It appears difficult to characterize the general form of F that ensures that L satisfies condition 5. An interesting special case of this loss is the familiar **negative log-likelihood loss**:

$$L_{\text{nll}}(W, Y^i, X^i) = \check{E}(W, Y^i, X^i) - F_\beta(W, X^i) \tag{9}$$

with

$$F_\beta(W, X^i) = -\frac{1}{\beta} \log \left(\int_{Y \in \{Y\}} \exp[-\beta \check{E}(W, Y, X^i)] \right) \tag{10}$$

where β is a positive constant. The second term can be interpreted as the Helmholtz free energy (log partition function) of the ensemble of systems with energies $\check{E}(W, Y, X^i)\ \forall Y \in \{Y\}$. This type of discriminative loss with $\beta = 1$ is widely used for discriminative probabilistic models in the speech, handwriting, and NLP communities [10, 2, 8]. This is also the loss function used in the *conditional random field* model of Lafferty et al [9].

We can see that loss (9) reduces to the generalized Perceptron loss when $\beta \to \infty$. Computing this loss and its derivative requires computing integrals (or sums) over $\{Y\}$ that may be intractable. It also assumes that the exponential of the energy be integrable over $\{Y\}$, which puts restrictions on the choice of $E(W, Y, X)$ and/or $\{Y\}$.

4 Illustrative Experiments

To illustrate the use of contrastive loss functions with non-probabilistic latent variables, we trained a system to recognize generic objects in images independently of the pose and the illumination. We used the NORB dataset [11] which contains 50 different uniform-colored toy objects under 18 azimuths, 9 elevations, and 6 lighting conditions. The objects are 10 instance from 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. Five instances of each category were used for training, and the other five for testing (see figure 4). A 6-layer convolutional network trained with the mean-square loss achieves 6.8% test error on this set when fed with binocular 96×96-pixel gray-scale images [11].

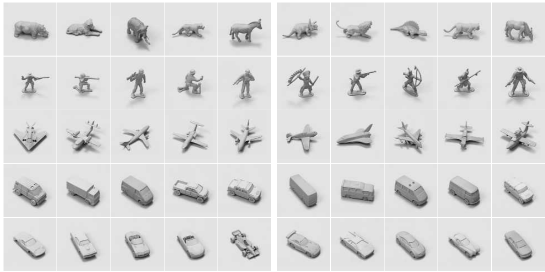

Figure 4: Invariant object recognition with NORB dataset. The left portion shows sample views of the training instances, and the right portion testing instances for the 5 categories.

We used an architecture very much like the one in figure 3, where the feature extraction module is identical to the first 5 layers of the 6-layer convolutional net used in the reference experiment. The object model functions were of the form: $E_i = ||W_i.V - F(Z)||$, $i = 1..5$, where V is the output of the 5-layer net (100 dimensions), W_i is a 9×100 (trainable) weight matrix. The latent variable Z has two dimensions that are meant to represent the azimuth and elevation of the object viewpoint. The set of possible values $\{Z\}$ contained 162 values (azimuths: 0-360 degrees every 20, elevations: 30-70 degrees every 5). The output of $F(Z)$ is a point on an azimuth/elevation half-sphere (2D surface) embedded in the 9D hypercube $[-1, +1]^9$. The minimization of the energy over Z is performed through exhaustive search (which is relatively cheap).

We used the loss function (7). This loss causes the convolutional net to produce a point as close as possible to any point on the half-sphere of the desired class, and as far as possible from the half-sphere of the best-scoring non-desired class. The system is trained "from scratch" including the convolutional net. We obtained 6.3% error on the test set, which is a moderate, but significant improvement over the 6.8% of the control experiment.

5 Discussion

Efficient End-to-End Gradient-Based Learning To perform gradient-based training of all the modules in the architecture, we must compute the gradient of the loss with respect to all the parameters. This is easily achieved with the module-based generalization of back-propagation described in [10]. A typical learning iteration would involve the following steps: (1) one forward propagation through the modules that only depend on X; (2) a run of the energy-minimizing inference algorithm on the modules that depend on Z and Y; (3) as many back-propagations through the modules that depend on Y as there are energy terms in the loss function; (4) one back-propagation through the module that depends only on X; (5) one update of the parameters.

Efficient Inference: Most loss functions described in this paper involve multiple runs of the machine (in the worst case, one run for each energy term that enters in the loss). However, the parts of the machine that solely depend on X, and not on Y or Z need not be recomputed for each run (e.g. the feature extractor in figure 3), because X does not change between runs.

If the energy function can be decomposed as a sum of functions (called *factors*) $E(W, Y, X) = \sum_j E_j(W_j, Y, Z, X)$, each of which takes subsets of the variables in Z and Y as input, we can use a form of belief propagation algorithm for factor graphs to compute the lowest energy configuration [13]. These algorithms are exact and tractable if Z and Y are discrete and the factor graph has no loop. They reduce to Viterbi-type algorithms when members of $\{Z\}$ can be represented by paths in a lattice (as is the case for sequence segmentation/labeling tasks).

Approximate Inference: We do not really need to assume that the energy-minimizing inference process always finds the global minimum of $E(W, Y, X^i)$ with respect to Y. We merely need to assume that this process finds approximate solutions (e.g. local minima) in a consistent, repeatable manner. Indeed, if there are minima of $E(W, Y, X^i)$ with respect to Y that our minimization/inference algorithm never finds, we do not need to find them and increase their energy. Their existence is irrelevant to our problem. However, it is important to note that those unreachable regions of low energy will affect the loss function of probabilistic models as well as that of EBMs if the negative log-likelihood loss is used. This is a distinct advantage of the un-normalized EBM approach: low-energy areas that are never reached by the inference algorithm are not a concern.

6 Conclusion and Outlook

Most approaches to discriminative training of graphical models in the literature use loss functions from a very small set. We show that energy-based (un-normalized) graphical models can be trained discriminatively using a very wide family of loss functions. We give a sufficient condition that the loss function must satisfy so that its minimization will make the system approach the desired behavior. We give a number of loss functions that satisfy this criterion and describe experiments in image recognition that illustrate the use of such discriminative loss functions in the presence of non-probabilistic latent variables.

Acknowledgments

The authors wish to thank Leon Bottou, Yoshua Bengio, Margarita Osadchy, and Matt Miller for useful discussions.

References

[1] Yasemin Altun, Mark Johnson, and Thomas Hofmann. Loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*, 2003.

[2] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transaction on Neural Networks*, 3(2):252–259, 1992.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, February 2003.

[4] L. Bottou. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. PhD thesis, Université de Paris XI, 91405 Orsay cedex, France, 1991.

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

[6] J. S. Denker and C. J. Burges. Image segmentation and recognition. In *The Mathematics of Induction*. Addison Wesley, 1995.

[7] X. Driancourt and L. Bottou. MLP, LVQ and DP: Comparison & cooperation. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 815–819, Seattle, 1991.

[8] P. Haffner. Connectionist speech recognition with a global MMI algorithm. In *Eurospeech'93*, Berlin, September 1993.

[9] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML)*, 2001.

[10] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[11] Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. CVPR*, 2004.

[12] M. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation. In *Proc. NIPS*, 2004.

[13] Kschischang F. R., B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Information Theory*, 47(2):498–519, February 2001.

[14] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proc. NIPS*, 2003.

[15] Y. W. Teh, M. Welling, S. Osindero, and Hinton G. E. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.