A Tutorial on Energy-Based Learning

Yann LeCun, Sumit Chopra, Raia Hadsell,
Marc'Aurelio Ranzato, and Fu Jie Huang
The Courant Institute of Mathematical Sciences,
New York University
{yann,sumit,raia,ranzato,jhuangfu}@cs.nyu.edu
http://yann.lecun.com
v1.0, August 19, 2006
To appear in "Predicting Structured Data",
G. Bakir, T. Hofman, B. Schölkopf, A. Smola, B. Taskar (eds)
MIT Press, 2006

Abstract

Energy-Based Models (EBMs) capture dependencies between variables by associating a scalar energy to each configuration of the variables. Inference consists in clamping the value of observed variables and finding configurations of the remaining variables that minimize the energy. Learning consists in finding an energy function in which observed configurations of the variables are given lower energies than unobserved ones. The EBM approach provides a common theoretical framework for many learning models, including traditional discriminative and generative approaches, as well as graph-transformer networks, conditional random fields, maximum margin Markov networks, and several manifold learning methods.

Probabilistic models must be properly normalized, which sometimes requires evaluating intractable integrals over the space of all possible variable configurations. Since EBMs have no requirement for proper normalization, this problem is naturally circumvented. EBMs can be viewed as a form of non-probabilistic factor graphs, and they provide considerably more flexibility in the design of architectures and training criteria than probabilistic approaches.

1 Introduction: Energy-Based Models

The main purpose of statistical modeling and machine learning is to encode dependencies between variables. By capturing those dependencies, a model can be used to answer questions about the values of unknown variables given the values of known variables.

Energy-Based Models (EBMs) capture dependencies by associating a scalar *energy* (a measure of compatibility) to each configuration of the variables. *Inference*, i.e., making a prediction or decision, consists in setting the value of observed variables

and finding values of the remaining variables that minimize the energy. *Learning* consists in finding an energy function that associates low energies to correct values of the remaining variables, and higher energies to incorrect values. A *loss functional*, minimized during learning, is used to measure the quality of the available energy functions. Within this common inference/learning framework, the wide choice of energy functions and loss functionals allows for the design of many types of statistical models, both probabilistic and non-probabilistic.

Energy-based learning provides a unified framework for many probabilistic and non-probabilistic approaches to learning, particularly for non-probabilistic training of graphical models and other structured models. Energy-based learning can be seen as an alternative to probabilistic estimation for prediction, classification, or decision-making tasks. Because there is no requirement for proper normalization, energy-based approaches avoid the problems associated with estimating the normalization constant in probabilistic models. Furthermore, the absence of the normalization condition allows for much more flexibility in the design of learning machines. Most probabilistic models can be viewed as special types of energy-based models in which the energy function satisfies certain normalizability conditions, and in which the loss function, optimized by learning, has a particular form.

This chapter presents a tutorial on energy-based models, with an emphasis on their use for structured output problems and sequence labeling problems. Section 1 introduces energy-based models and describes deterministic inference through energy minimization. Section 2 introduces energy-based learning and the concept of the loss function. A number of standard and non-standard loss functions are described, including the perceptron loss, several margin-based losses, and the negative log-likelihood loss. The negative log-likelihood loss can be used to train a model to produce conditional probability estimates. Section 3 shows how simple regression and classification models can be formulated in the EBM framework. Section 4 concerns models that contain latent variables. Section 5 analyzes the various loss functions in detail and gives sufficient conditions that a loss function must satisfy so that its minimization will cause the model to approach the desired behavior. A list of "good" and "bad" loss functions is given. Section 6 introduces the concept of non-probabilistic factor graphs and informally discusses efficient inference algorithms. Section 7 focuses on sequence labeling and structured output models. Linear models such as max-margin Markov networks and conditional random fields are re-formulated in the EBM framework. The literature on discriminative learning for speech and handwriting recognition, going back to the late 80's and early 90's, is reviewed. This includes globally trained systems that integrate non-linear discriminant functions, such as neural networks, and sequence alignment methods, such as dynamic time warping and hidden Markov models. Hierarchical models such as the graph transformer network architecture are also reviewed. Finally, the differences, commonalities, and relative advantages of energy-based approaches, probabilistic approaches, and sampling-based approximate methods such as contrastive divergence are discussed in Section 8.

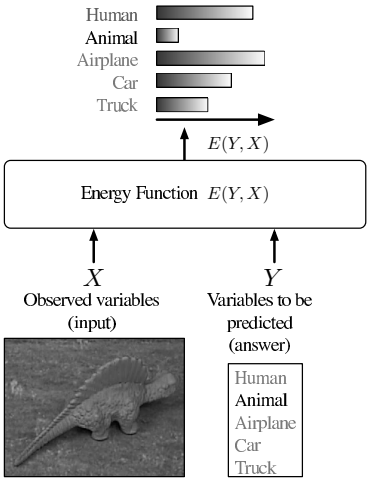

Figure 1: *A model measures the compatibility between observed variables X and variables to be predicted Y using an* energy function *$E(Y, X)$. For example, X could be the pixels of an image, and Y a discrete label describing the object in the image. Given X, the model produces the answer Y that minimizes the energy E.*

1.1 Energy-Based Inference

Let us consider a model with two sets of variables, X and Y, as represented in Figure 1. Variable X could be a vector containing the pixels from an image of an object. Variable Y could be a discrete variable that represents the possible category of the object. For example, Y could take six possible values: animal, human figure, airplane, truck, car, and "none of the above". The model is viewed as an *energy function* which measures the "goodness" (or badness) of each possible configuration of X and Y. The output number can be interpreted as the degree of *compatibility* between the values of X and Y. In the following, we use the convention that small energy values correspond to highly compatible configurations of the variables, while large energy values correspond to highly incompatible configurations of the variables. Functions of this type are given different names in different technical communities; they may be called contrast functions, value functions, or negative log-likelihood functions. In the following, we will use the term *energy function* and denote it $E(Y, X)$. A distinction should be made between the energy function, which is minimized by the inference process, and the loss functional (introduced in Section 2), which is minimized by the learning process.

In the most common use of a model, the input X is given (observed from the world), and the model produces the answer Y that is most compatible with the observed X. More precisely, the model must produce the value Y^*, chosen from a set $\mathcal{Y}$, for which $E(Y, X)$ is the smallest:

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X). \tag{1}$$

When the size of the set $\mathcal{Y}$ is small, we can simply compute $E(Y, X)$ for all possible values of $Y \in \mathcal{Y}$ and pick the smallest.

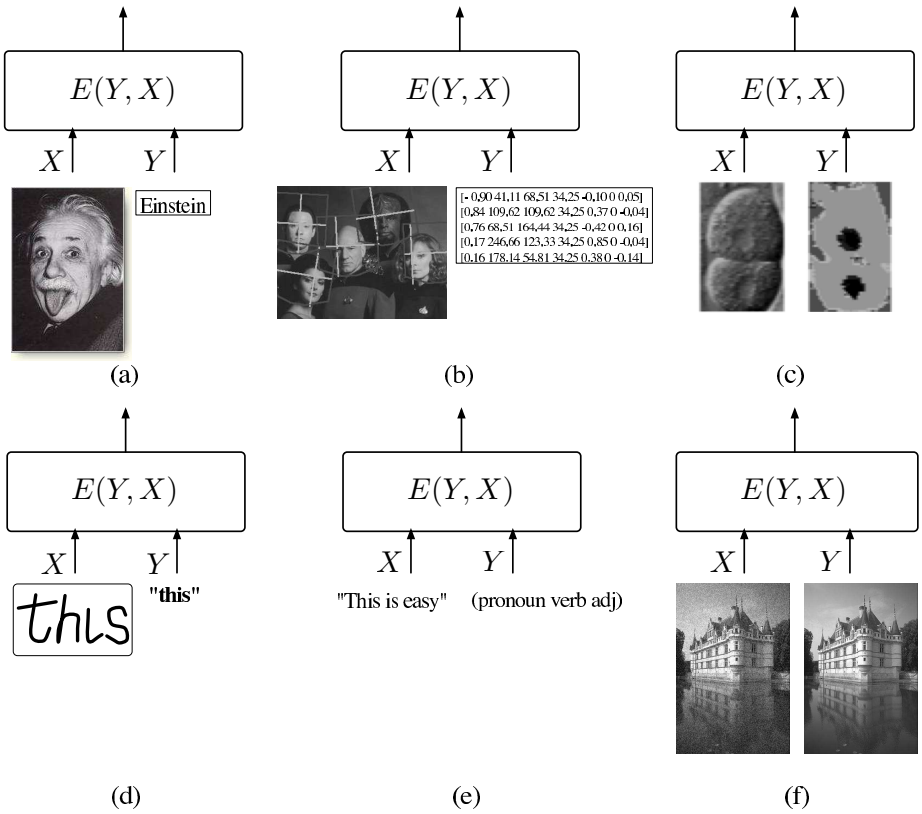

Figure 2: *Several applications of EBMs:* **(a) face recognition:** *Y is a high-cardinality discrete variable;* **(b) face detection and pose estimation:** *Y is a collection of vectors with location and pose of each possible face;* **(c) image segmentation:** *Y is an image in which each pixel is a discrete label;* **(d-e) handwriting recognition and sequence labeling:** *Y is a sequence of symbols from a highly structured but potentially infinite set (the set of English sentences). The situation is similar for many applications in natural language processing and computational biology;* **(f) image restoration:** *Y is a high-dimensional continuous variable (an image).*

In general, however, picking the best Y may not be simple. Figure 2 depicts several situations in which $\mathcal{Y}$ may be too large to make exhaustive search practical. In Figure 2(a), the model is used to recognize a face. In this case, the set $\mathcal{Y}$ is discrete and finite, but its cardinality may be tens of thousands [Chopra et al., 2005]. In Figure 2(b), the model is used to find the faces in an image and estimate their poses. The set $\mathcal{Y}$ contains a binary variable for each location indicating whether a face is present at that location, and a set of continuous variables representing the size and orientation of the face [Osadchy et al., 2005]. In Figure 2(c), the model is used to segment a biological image: each pixel must be classified into one of five categories (cell nucleus, nuclear membrane, cytoplasm, cell membrane, external medium). In this case, $\mathcal{Y}$ contains all the *consistent* label images, i.e. the ones for which the nuclear membranes are encircling the nuclei, the nuclei and cytoplasm are inside the cells walls, etc. The set is discrete, but intractably large. More importantly, members of the set must satisfy complicated consistency constraints [Ning et al., 2005]. In Figure 2(d), the model is used to recognize a handwritten sentence. Here $\mathcal{Y}$ contains all possible sentences of the English language, which is a discrete but infinite set of sequences of symbols [LeCun et al., 1998a]. In Figure 2(f), the model is used to restore an image (by cleaning the noise, enhancing the resolution, or removing scratches). The set $\mathcal{Y}$ contains all possible images (all possible pixel combinations). It is a continuous and high-dimensional set.

For each of the above situations, a specific strategy, called the *inference procedure*, must be employed to find the Y that minimizes $E(Y, X)$. In many real situations, the inference procedure will produce an approximate result, which may or may not be the global minimum of $E(Y, X)$ for a given X. In fact, there may be situations where $E(Y, X)$ has several equivalent minima. The best inference procedure to use often depends on the internal structure of the model. For example, if $\mathcal{Y}$ is continuous and $E(Y, X)$ is smooth and well-behaved with respect to Y, one may use a gradient-based optimization algorithm. If Y is a collection of discrete variables and the energy function can be expressed as a *factor graph*, i.e. a sum of energy functions (factors) that depend on different subsets of variables, efficient inference procedures for factor graphs can be used (see Section 6) [Kschischang et al., 2001, MacKay, 2003]. A popular example of such a procedure is the *min-sum* algorithm. When each element of $\mathcal{Y}$ can be represented as a path in a weighted directed acyclic graph, then the energy for a particular Y is the sum of values on the edges and nodes along a particular path. In this case, the best Y can be found efficiently using dynamic programming (e.g with the Viterbi algorithm or A^*). This situation often occurs in sequence labeling problems such as speech recognition, handwriting recognition, natural language processing, and biological sequence analysis (e.g. gene finding, protein folding prediction, etc). Different situations may call for the use of other optimization procedures, including continuous optimization methods such as linear programming, quadratic programming, non-linear optimization methods, or discrete optimization methods such as simulated annealing, graph cuts, or graph matching. In many cases, exact optimization is impractical, and one must resort to approximate methods, including methods that use surrogate energy functions (such as variational methods).

1.2 What Questions Can a Model Answer?

In the preceding discussion, we have implied that the question to be answered by the model is "What is the Y that is most compatible with this X?", a situation that occurs in *prediction, classification* or *decision-making* tasks. However, a model may be used to answer questions of several types:

1. *Prediction, classification, and decision-making*: "Which value of Y is most compatible with this X?' This situation occurs when the model is used to make hard decisions or to produce an action. For example, if the model is used to drive a robot and avoid obstacles, it must produce a single best decision such as "steer left", "steer right", or "go straight".

2. *Ranking*: "Is Y_1 or Y_2 more compatible with this X?" This is a more complex task than classification because the system must be trained to produce a complete ranking of all the answers, instead of merely producing the best one. This situation occurs in many data mining applications where the model is used to select multiple samples that best satisfy a given criterion.

3. *Detection*: "Is this value of Y compatible with X?" Typically, detection tasks, such as detecting faces in images, are performed by comparing the energy of a *face* label with a threshold. Since the threshold is generally unknown when the system is built, the system must be trained to produce energy values that increase as the image looks less like a face.

4. *Conditional density estimation*: "What is the conditional probability distribution over $\mathcal{Y}$ given X?" This case occurs when the output of the system is not used directly to produce actions, but is given to a human decision maker or is fed to the input of another, separately built system.

We often think of X as a high-dimensional variable (e.g. an image) and Y as a discrete variable (e.g. a label), but the converse case is also common. This occurs when the model is used for such applications as image restoration, computer graphics, speech and language production, etc. The most complex case is when both X and Y are high-dimensional.

1.3 Decision Making versus Probabilistic Modeling

For decision-making tasks, such as steering a robot, it is merely necessary that the system give the lowest energy to the correct answer. The energies of other answers are irrelevant, as long as they are larger. However, the output of a system must sometimes be combined with that of another system, or fed to the input of another system (or to a human decision maker). Because energies are uncalibrated (i.e. measured in arbitrary units), combining two, separately trained energy-based models is not straightforward: there is no *a priori* guarantee that their energy scales are commensurate. Calibrating energies so as to permit such combinations can be done in a number of ways. However, the only *consistent* way involves turning the collection of energies for all possible outputs into a normalized probability distribution. The simplest and most common method

for turning a collection of arbitrary energies into a collection of numbers between 0 and 1 whose sum (or integral) is 1 is through the *Gibbs distribution*:

$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}}, \tag{2}$$

where β is an arbitrary positive constant akin to an inverse temperature, and the denominator is called the *partition function* (by analogy with similar concepts in statistical physics). The choice of the Gibbs distribution may seem arbitrary, but other probability distributions can be obtained (or approximated) through a suitable re-definition of the energy function. Whether the numbers obtained this way are good probability estimates does not depend on how energies are turned into probabilities, but on how $E(Y, X)$ is estimated from data.

It should be noted that the above transformation of energies into probabilities is only possible if the integral $\int_{y \in \mathcal{Y}} e^{-\beta E(y,X)}$ converges. This somewhat restricts the energy functions and domains $\mathcal{Y}$ that can be used. More importantly, there are many practical situations where computing the partition function is intractable (e.g. when $\mathcal{Y}$ has high cardinality), or outright impossible (e.g. when $\mathcal{Y}$ is a high dimensional variable and the integral has no analytical solution). Hence probabilistic modeling comes with a high price, and should be avoided when the application does not require it.

2 Energy-Based Training: Architecture and Loss Function

Training an EBM consists in finding an energy function that produces the best Y for any X. The search for the best energy function is performed within a family of energy functions $\mathcal{E}$ indexed by a parameter W

$$\mathcal{E} = \{E(W, Y, X) : \quad W \in \mathcal{W}\}. \tag{3}$$

The *architecture* of the EBM is the internal structure of the parameterized energy function $E(W, Y, X)$. At this point, we put no particular restriction on the nature of X, Y, W, and $\mathcal{E}$. When X and Y are real vectors, $\mathcal{E}$ could be as simple as a linear combination of basis functions (as in the case of kernel methods), or a set of neural net architectures and weight values. Section gives examples of simple architectures for common applications to classification and regression. When X and Y are variable-size images, sequences of symbols or vectors, or more complex structured objects, $\mathcal{E}$ may represent a considerably richer class of functions. Sections 4, 6 and 7 discuss several examples of such architectures. One advantage of the energy-based approach is that it puts very little restrictions on the nature of $\mathcal{E}$.

To train the model for prediction, classification, or decision-making, we are given a set of training samples $\mathcal{S} = \{(X^i, Y^i) : i = 1 \ldots P\}$, where X^i is the input for the i-th training sample, and Y^i is the corresponding desired answer. In order to find the best energy function in the family $\mathcal{E}$, we need a way to assess the quality of any

particular energy function, based solely on two elements: the training set, and our prior knowledge about the task. This quality measure is called the *loss functional* (i.e. a function of function) and denoted $\mathcal{L}(E, \mathcal{S})$. For simplicity, we often denote it $\mathcal{L}(W, \mathcal{S})$ and simply call it the *loss function*. The learning problem is simply to find the W that minimizes the loss:

$$W^* = \min_{W \in \mathcal{W}} \mathcal{L}(W, \mathcal{S}). \tag{4}$$

For most cases, the loss functional is defined as follows:

$$\mathcal{L}(E, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} L(Y^i, E(W, \mathcal{Y}, X^i)) + R(W). \tag{5}$$

It is an average taken over the training set of a *per-sample loss functional*, denoted $L(Y^i, E(W, \mathcal{Y}, X^i))$, which depends on the desired answer Y^i and on the energies obtained by keeping the input sample fixed and varying the answer Y. Thus, for each sample, we evaluate a "slice" of the energy surface. The term $R(W)$ is the *regularizer*, and can be used to embed our prior knowledge about which energy functions in our family are preferable to others (in the absence of training data). With this definition, the loss is invariant under permutations of the training samples and under multiple repetitions of the training set.

Naturally, the ultimate purpose of learning is to produce a model that will give good answers for new input samples that are not seen during training. We can rely on general results from statistical learning theory which guarantee that, under simple interchangeability conditions on the samples and general conditions on the family of energy functions (finite VC dimension), the deviation between the value of the loss after minimization on the training set, and the loss on a large, separate set of test samples is bounded by a quantity that converges to zero as the size of training set increases [Vapnik, 1995].

2.1 Designing a Loss Functional

Intuitively, the per-sample loss functional should be designed in such a way that it assigns a low loss to *well-behaved* energy functions: energy functions that give the lowest energy to the correct answer and higher energy to all other (incorrect) answers. Conversely, energy functions that do not assign the lowest energy to the correct answers would have a high loss. Characterizing the appropriateness of loss functions (the ones that select the best energy functions) is further discussed in following sections.

Considering only the task of training a model to answer questions of type 1 (prediction, classification and decision-making), the main intuition of the energy-based approach is as follows. Training an EBM consists in shaping the energy function, so that for any given X, the inference algorithm will produce the desired value for Y. Since the inference algorithm selects the Y with the lowest energy, the learning procedure must shape the energy surface so that the desired value of Y has lower energy than all other (undesired) values. Figures 3 and 4 show examples of energy as a function of Y for a given input sample X^i in cases where Y is a discrete variable and a continuous scalar variable. We note three types of answers:

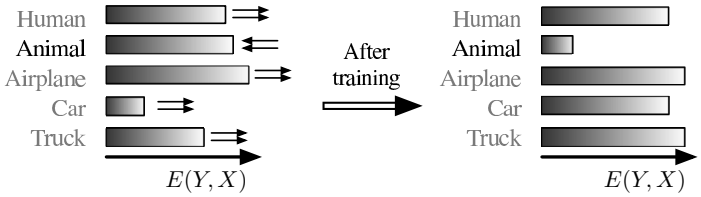

Figure 3: *How training affects the energies of the possible answers in the discrete case: the energy of the correct answer is decreased, and the energies of incorrect answers are increased, particularly if they are lower than that of the correct answer.*

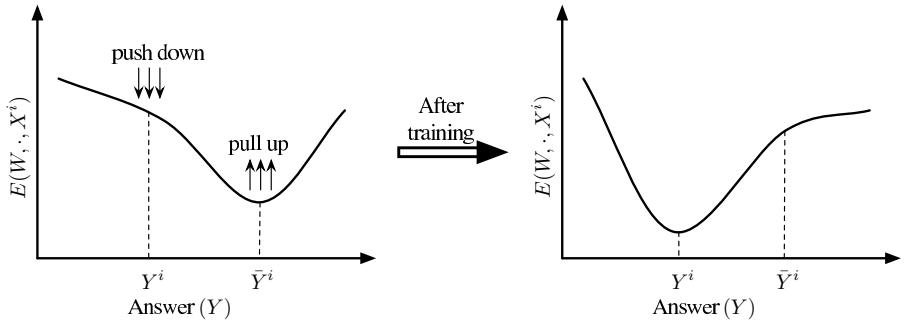

Figure 4: *The effect of training on the energy surface as a function of the answer Y in the continuous case. After training, the energy of the correct answer Y^i is lower than that of incorrect answers.*

- Y^i: the correct answer
- Y^{*i}: the answer produced by the model, i.e. the answer with the lowest energy.
- $\bar{Y}^i$: the *most offending incorrect answer*, i.e. the answer that has the lowest energy among all the incorrect answers. To define this answer in the continuous case, we can simply view all answers within a distance ϵ of Y^i as correct, and all answers beyond that distance as incorrect.

With a properly designed loss function, the learning process should have the effect of "pushing down" on $E(W, Y^i, X^i)$, and "pulling up" on the incorrect energies, particularly on $E(W, \bar{Y}^i, X^i)$. Different loss functions do this in different ways. Section 5 gives sufficient conditions that the loss function must satisfy in order to be guaranteed to shape the energy surface correctly. We show that some widely used loss functions do not satisfy the conditions, while others do.

To summarize: given a training set $\mathcal{S}$, building and training an energy-based model involves designing four components:

1. *The architecture*: the internal structure of $E(W, Y, X)$.
2. *The inference algorithm*: the method for finding a value of Y that minimizes $E(W, Y, X)$ for any given X.
3. *The loss function*: $\mathcal{L}(W, \mathcal{S})$ measures the quality of an energy function using the training set.
4. *The learning algorithm*: the method for finding a W that minimizes the loss functional over the family of energy functions $\mathcal{E}$, given the training set.

Properly designing the architecture and the loss function is critical. Any prior knowledge we may have about the task at hand is embedded into the architecture and into the loss function (particularly the regularizer). Unfortunately, not all combinations of architectures and loss functions are allowed. With some combinations, minimizing the loss will not make the model produce the best answers. Choosing the combinations of architecture and loss functions that can learn effectively and efficiently is critical to the energy-based approach, and thus is a central theme of this tutorial.

2.2 Examples of Loss Functions

We now describe a number of standard loss functions that have been proposed and used in the machine learning literature. We shall discuss them and classify them as "good" or "bad" in an energy-based setting. For the time being, we set aside the regularization term, and concentrate on the data-dependent part of the loss function.

2.2.1 Energy Loss

The simplest and the most straightforward of all the loss functions is the energy loss. For a training sample (X^i, Y^i), the per-sample loss is defined simply as:

$$L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i). \tag{6}$$

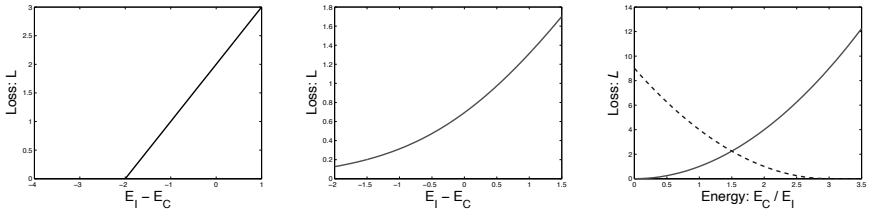

Figure 5: *The hinge loss (left) and log loss (center) penalize $E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)$ linearly and logarithmically, respectively. The square-square loss (right) separately penalizes large values of $E(W, Y^i, X^i)$ (solid line) and small values of $E(W, \bar{Y}^i, X^i)$ (dashed line) quadratically.*

This loss function, although very popular for things like regression and neural network training, cannot be used to train most architectures: while this loss will push down on the energy of the desired answer, it will not pull up on any other energy. With some architectures, this can lead to a *collapsed solution* in which the energy is constant and equal to zero. The energy loss will only work with architectures that are designed in such a way that pushing down on $E(W, Y^i, X^i)$ will automatically make the energies of the other answers larger. A simple example of such an architecture is $E(W, Y^i, X^i) = ||Y^i - G(W, X^i)||^2$, which corresponds to regression with mean-squared error with G being the regression function.

2.2.2 Generalized Perceptron Loss

The generalized perceptron loss for a training sample (X^i, Y^i) is defined as

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i). \qquad (7)$$

This loss is always positive, since the second term is a lower bound on the first term. Minimizing this loss has the effect of pushing down on $E(W, Y^i, X^i)$, while pulling up on the energy of the answer produced by the model.

While the perceptron loss has been widely used in many settings, including for models with structured outputs such as handwriting recognition [LeCun et al., 1998a] and parts of speech tagging [Collins, 2002], it has a major deficiency: there is no mechanism for creating an energy gap between the correct answer and the incorrect ones. Hence, as with the energy loss, the perceptron loss may produce flat (or almost flat) energy surfaces if the architecture allows it. Consequently, a meaningful, uncollapsed result is only guaranteed with this loss if a model is used that cannot produce a flat energy surface. For other models, one cannot guarantee anything.

2.2.3 Generalized Margin Losses

Several loss functions can be described as *margin* losses; the hinge loss, log loss, LVQ2 loss, minimum classification error loss, square-square loss, and square-exponential loss all use some form of margin to create an energy gap between the correct answer and the

incorrect answers. Before discussing the generalized margin loss we give the following definitions.

Definition 1 *Let Y be a discrete variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are incorrect:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y} and Y \neq Y^i} E(W, Y, X^i). \tag{8}$$

If Y is a continuous variable then the definition of the most offending incorrect answer can be defined in a number of ways. The simplest definition is as follows.

Definition 2 *Let Y be a continuous variable. Then for a training sample (X^i, Y^i), the* ***most offending incorrect answer*** *$\bar{Y}^i$ is the answer that has the lowest energy among all answers that are at least ϵ away from the correct answer:*

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \tag{9}$$

The generalized margin loss is a more robust version of the generalized perceptron loss. It directly uses the energy of the most offending incorrect answer in the contrastive term:

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m\left(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)\right). \tag{10}$$

Here m is a positive parameter called the *margin* and $Q_m(e_1, e_2)$ is a convex function whose gradient has a positive dot product with the vector $[1, -1]$ in the region where $E(W, Y^i, X^i) + m > E(W, \bar{Y}^i, X^i)$. In other words, the loss surface is slanted toward low values of $E(W, Y^i, X^i)$ and high values of $E(W, \bar{Y}^i, X^i)$ wherever $E(W, Y^i, X^i)$ is not smaller than $E(W, \bar{Y}^i, X^i)$ by at least m. Two special cases of the generalized margin loss are given below:

Hinge Loss: A particularly popular example of generalized margin loss is the *hinge loss*, which is used in combination with linearly parameterized energies and a quadratic regularizer in support vector machines, support vector Markov models [Altun and Hofmann, 2003], and maximum-margin Markov networks [Taskar et al., 2003]:

$$L_{\text{hinge}}(W, Y^i, X^i) = \max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right), \tag{11}$$

where m is the positive margin. The shape of this loss function is given in Figure 5. The difference between the energies of the correct answer and the most offending incorrect answer is penalized linearly when larger than $-m$. The hinge loss only depends on energy differences, hence individual energies are not constrained to take any particular value.

Log Loss: a common variation of the hinge loss is the *log loss*, which can be seen as a "soft" version of the hinge loss with an infinite margin (see Figure 5, center):

$$L_{\log}(W, Y^i, X^i) = \log\left(1 + e^{E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)}\right). \tag{12}$$

LVQ2 Loss: One of the very first proposals for discriminatively training sequence labeling systems (particularly speech recognition systems)

is a version of Kohonen's LVQ2 loss. This loss has been advocated by Driancourt and Bottou since the early 90's [Driancourt et al., 1991a, Driancourt and Gallinari, 1992b, Driancourt and Gallinari, 1992a, Driancourt, 1994, McDermott, 1997, McDermott and Katagiri, 1992]:

$$L_{\text{lvq2}}(W, Y^i, X^i) = \min\left(1, \max\left(0, \frac{E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)}{\delta E(W, \bar{Y}^i, X^i)}\right)\right), \quad (13)$$

where δ is a positive parameter. LVQ2 is a zero-margin loss, but it has the peculiarity of saturating the ratio between $E(W, Y^i, X^i)$ and $E(W, \bar{Y}^i, X^i)$ to $1 + \delta$. This mitigates the effect of outliers by making them contribute a nominal cost M to the total loss. This loss function is a continuous approximation of the number of classification errors. Unlike generalized margin losses, the LVQ2 loss is non-convex in $E(W, Y^i, X^i)$ and $E(W, \bar{Y}^i, X^i)$.

MCE Loss: The Minimum Classification Error loss was originally proposed by Juang et al. in the context of discriminative training for speech recognition systems [Juang et al., 1997]. The motivation was to build a loss function that also approximately counts the number of classification errors, while being smooth and differentiable. The number of classification errors can be written as:

$$\theta\left(E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right), \quad (14)$$

where θ is the step function (equal to zero for negative arguments, and 1 for positive arguments). However, this function is not differentiable, and therefore very difficult to optimize. The MCE Loss "softens" it with a sigmoid:

$$L_{\text{mce}}(W, Y^i, X^i) = \sigma\left(E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right), \quad (15)$$

where σ is the logistic function $\sigma(x) = (1 + e^{-x})^{-1}$. As with the LVQ2 loss, the saturation ensures that mistakes contribute a nominal cost to the overall loss. Although the MCE loss does not have an explicit margin, it does create a gap between $E(W, Y^i, X^i)$ and $E(W, \bar{Y}^i, X^i)$. The MCE loss is non-convex.

Square-Square Loss: Unlike the hinge loss, the square-square loss treats the energy of the correct answer and the most offending answer separately [LeCun and Huang, 2005, Hadsell et al., 2006]:

$$L_{\text{sq-sq}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2. \quad (16)$$

Large values of $E(W, Y^i, X^i)$ and small values of $E(W, \bar{Y}^i, X^i)$ below the margin m are both penalized quadratically (see Figure 5). Unlike the margin loss, the square-square loss "pins down" the correct answer energy at zero and "pins down" the incorrect answer energies above m. Therefore, it is only suitable for energy functions that are bounded below by zero, notably in architectures whose output module measures some sort of distance.

Square-Exponential [LeCun and Huang, 2005, Chopra et al., 2005, Osadchy et al., 2005]: The *square-exponential* loss is similar to the *square-square* loss. It only differs in the contrastive term: instead of a quadratic term it has the exponential of the negative energy of the most offending incorrect answer:

$$L_{\text{sq-exp}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \gamma e^{-E(W, \bar{Y}^i, X^i)}, \quad (17)$$

where γ is a positive constant. Unlike the square-square loss, this loss has an infinite margin and pushes the energy of the incorrect answers to infinity with exponentially decreasing force.

2.2.4 Negative Log-Likelihood Loss

The motivation for the negative log-likelihood loss comes from probabilistic modeling. It is defined as:

$$L_{\text{nll}}(W, Y^i, X^i) = E(W, Y^i, X^i) + \mathcal{F}_\beta(W, \mathcal{Y}, X^i). \tag{18}$$

Where $\mathcal{F}$ is the *free energy* of the ensemble $\{E(W, y, X^i),\ y \in \mathcal{Y}\}$:

$$\mathcal{F}_\beta(W, \mathcal{Y}, X^i) = \frac{1}{\beta} \log \left(\int_{y \in \mathcal{Y}} \exp\left(-\beta E(W, y, X^i)\right) \right). \tag{19}$$

where β is a positive constant akin to an inverse temperature. This loss can only be used if the exponential of the negative energy is integrable over $\mathcal{Y}$, which may not be the case for some choices of energy function or $\mathcal{Y}$.

The form of the negative log-likelihood loss stems from a probabilistic formulation of the learning problem in terms of the maximum conditional probability principle. Given the training set $\mathcal{S}$, we must find the value of the parameter that maximizes the conditional probability of all the answers given all the inputs in the training set. Assuming that the samples are independent, and denoting by $P(Y^i|X^i, W)$ the conditional probability of Y^i given X^i that is produced by our model with parameter W, the conditional probability of the training set under the model is a simple product over samples:

$$P(Y^1, \ldots, Y^P | X^1, \ldots, X^P, W) = \prod_{i=1}^{P} P(Y^i | X^i, W). \tag{20}$$

Applying the maximum likelihood estimation principle, we seek the value of W that maximizes the above product, or the one that minimizes the negative $\log$ of the above product:

$$-\log \prod_{i=1}^{P} P(Y^i | X^i, W) = \sum_{i=1}^{P} -\log P(Y^i | X^i, W). \tag{21}$$

Using the Gibbs distribution (Equation 2), we get:

$$-\log \prod_{i=1}^{P} P(Y^i | X^i, W) = \sum_{i=1}^{P} \beta E(W, Y^i, X^i) + \log \int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)}. \tag{22}$$

The final form of the negative log-likelihood loss is obtained by dividing the above expression by P and β (which has no effect on the position of the minimum):

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left(E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)} \right). \tag{23}$$

While many of the previous loss functions involved only $E(W, \bar{Y}^i, X^i)$ in their contrastive term, the negative log-likelihood loss combines all the energies for all values of Y in its contrastive term $\mathcal{F}_\beta(W, \mathcal{Y}, X^i)$. This term can be interpreted as the Helmholtz free energy (log partition function) of the ensemble of systems with energies $E(W, Y, X^i),\ Y \in \mathcal{Y}$. This contrastive term causes the energies of all the answers to be pulled up. The energy of the correct answer is also pulled up, but not as hard as it is pushed down by the first term. This can be seen in the expression of the gradient for a single sample:

$$\frac{\partial L_{\text{nll}}(W, Y^i, X^i)}{\partial W} = \frac{\partial E(W, Y^i, X^i)}{\partial W} - \int_{Y \in \mathcal{Y}} \frac{\partial E(W, Y, X^i)}{\partial W} P(Y|X^i, W), \quad (24)$$

where $P(Y|X^i, W)$ is obtained through the Gibbs distribution:

$$P(Y|X^i, W) = \frac{e^{-\beta E(W, Y, X^i)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)}}. \quad (25)$$

Hence, the contrastive term pulls up on the energy of each answer with a force proportional to the likelihood of that answer under the model. Unfortunately, there are many interesting models for which computing the integral over $\mathcal{Y}$ is intractable. Evaluating this integral is a major topic of research. Considerable efforts have been devoted to approximation methods, including clever organization of the calculations, Monte-Carlo sampling methods, and variational methods. While these methods have been devised as approximate ways of minimizing the NLL loss, they can be viewed in the energy-based framework as different strategies for choosing the Y's whose energies will be pulled up.

Interestingly, the NLL loss reduces to the generalized perceptron loss when $\beta \to \infty$ (zero temperature), and reduces to the log loss (Eq. 12) when $\mathcal{Y}$ has two elements (e.g. binary classification).

The NLL loss has been used extensively by many authors under various names. In the neural network classification literature, it is known as the *cross-entropy loss* [Solla et al., 1988]. It was also used by Bengio et al. to train an energy-based language model [Bengio et al., 2003]. It has been widely used under the name *maximum mutual information estimation* for discriminatively training speech recognition systems since the late 80's, including hidden Markov models with mixtures of Gaussians [Bahl et al., 1986], and HMM-neural net hybrids [Bengio et al., 1990, Bengio et al., 1992, Haffner, 1993, Bengio, 1996]. It has also been used extensively for global discriminative training of handwriting recognition systems that integrate neural nets and hidden Markov models under the names *maximum mutual information* [Bengio et al., 1993, LeCun and Bengio, 1994, Bengio et al., 1995, LeCun et al., 1997, Bottou et al., 1997] and *discriminative forward training* [LeCun et al., 1998a]. Finally, it is the loss function of choice for training other probabilistic discriminative sequence labeling models such as input/output HMM [Bengio and Frasconi, 1996], conditional random fields [Lafferty et al., 2001], and discriminative random fields [Kumar and Hebert, 2004].

Minimum Empirical Error Loss: Some authors have argued that the negative log likelihood loss puts too much emphasis on mistakes: Eq. 20 is a product whose value

is dominated by its smallest term. Hence, Ljolje et al. [Ljolje et al., 1990] proposed the *minimum empirical error loss*, which combines the conditional probabilities of the samples additively instead of multiplicatively:

$$L_{\text{mee}}(W, Y^i, X^i) = 1 - P(Y^i|X^i, W). \tag{26}$$

Substituting Equation 2 we get:

$$L_{\text{mee}}(W, Y^i, X^i) = 1 - \frac{e^{-\beta E(W,Y^i,X^i)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}}. \tag{27}$$

As with the MCE loss and the LVQ2 loss, the MEE loss saturates the contribution of any single error. This makes the system more robust to label noise and outliers, which is of particular importance to such applications such as speech recognition, but it makes the loss non-convex. As with the NLL loss, MEE requires evaluating the partition function.

3 Simple Architectures

To substantiate the ideas presented thus far, this section demonstrates how simple models of classification and regression can be formulated as energy-based models. This sets the stage for the discussion of good and bad loss functions, as well as for the discussion of advanced architectures for structured prediction.

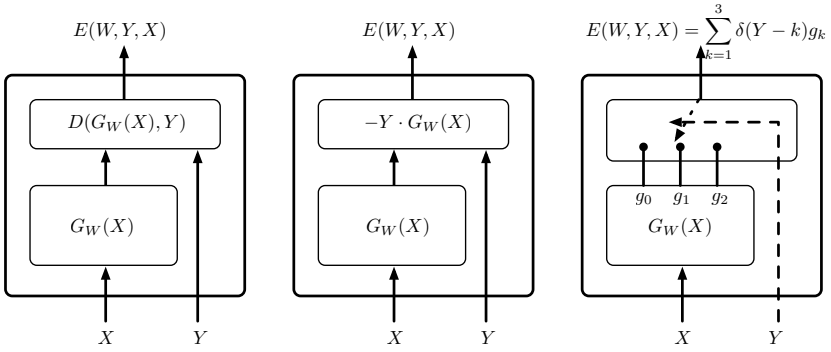

Figure 6: *Simple learning models viewed as EBMs:* **(a) a regressor:** *The energy is the discrepancy between the output of the regression function $G_W(X)$ and the answer Y. The best inference is simply $Y^* = G_W(X)$;* **(b) a simple two-class classifier:** *The set of possible answers is $\{-1, +1\}$. The best inference is $Y^* = \text{sign}(G_W(X))$;* **(c) a multiclass classifier:** *The discriminant function produces one value for each of the three categories. The answer, which can take three values, controls the position of a "switch", which connects one output of the discriminant function to the energy function. The best inference is the index of the smallest output component of $G_W(X)$.*

- Graph transformer networks are hierarchical sequence modeling systems in which the objects that are manipulated are trellises containing all the alternative interpretations at a given level. Global training can be performed using stochastic gradient by using a form of back-propagation algorithm to compute the gradients of the loss with respect to all the parameters in the system.

Acknowledgments

The authors wish to thank Geoffrey Hinton, Leon Bottou, Yoshua Bengio, Sebastian Seung, and Brendan Frey for helpful discussions.

This work was supported in part by NSF ITR grant 0325463 "New directions in predictive learning".

References

[Altun and Hofmann, 2003] Altun, Y. and Hofmann, T. (2003). Large margin methods for label sequence learning. In *Proc. of 8th European Conference on Speech Communication and Technology (EuroSpeech)*.

[Altun et al., 2003] Altun, Y., Johnson, M., and Hofmann, T. (2003). Loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*.

[Bahl et al., 1986] Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proceedings of Acoustics, Speech, and Signal Processing Conference*, pages 49–52.

[Becker and LeCun, 1989] Becker, S. and LeCun, Y. (1989). Improving the convergence of back-propagation learning with second-order methods. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proc. of the 1988 Connectionist Models Summer School*, pages 29–37, San Mateo. Morgan Kaufman.

[Bengio, 1996] Bengio, Y. (1996). *Neural Networks for Speech and Sequence Recognition*. International Thompson Computer Press, London, UK.

[Bengio et al., 1990] Bengio, Y., Cardin, R., De Mori, R., and Normandin, Y. (1990). A hybrid coder for hidden markov models using a recurrent network. In *Proceeding of ICASSP*, pages 537–540.

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

[Bengio et al., 1991] Bengio, Y., DeMori, R., Flammia, G., and Kompe, R. (1991). Global optimization of a neural network - hidden markov model hybrid. In *Proceedings of EuroSpeech'91*.

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

[Bengio and Frasconi, 1996] Bengio, Y. and Frasconi, P. (1996). An input/output HMM architecture. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 427–434. MIT Press, Cambridge, MA.

[Bengio et al., 1993] Bengio, Y., LeCun, Y., and Henderson, D. (1993). Globally trained handwritten word recognizer using spatial representation, space displacement neural networks and hidden markov models. In Cowan, J. and Tesauro, G., editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann.

[Bengio et al., 1995] Bengio, Y., LeCun, Y., Nohl, C., and Burges, C. (1995). Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303.

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

[Bottou, 2004] Bottou, L. (2004). Stochastic learning. In Bousquet, O. and von Luxburg, U., editors, *Advanced Lectures on Machine Learning*, number LNAI 3176 in Lecture Notes in Artificial Intelligence, pages 146–168. Springer Verlag, Berlin.

[Bottou and LeCun, 2004] Bottou, L. and LeCun, Y. (2004). Large-scale on-line learning. In *Advances in Neural Information Processing Systems 15*. MIT Press.

[Bottou et al., 1997] Bottou, L., LeCun, Y., and Bengio, Y. (1997). Global training of document processing systems using graph transformer networks. In *Proc. of Computer Vision and Pattern Recognition*, pages 490–494, Puerto-Rico. IEEE.

[Bourlard and Morgan, 1990] Bourlard, H. and Morgan, N. (1990). A continuous speech recognition system embedding mlp into hmm. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 186–193. Morgan Kaufmann.

[Bromley et al., 1993] Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. (1993). Signature verification using a siamese time delay neural network. In Cowan, J. and Tesauro, G., editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proc. of Computer Vision and Pattern Recognition Conference*. IEEE Press.

[Collins, 2000] Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of ICML 2000*.

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

[Collobert et al., 2006] Collobert, R., Weston, J., and Bottou, L. (2006). Trading convexity for scalability. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*. IMLS/ICML. ACM Digital Library.

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

[Driancourt, 1994] Driancourt, X. (1994). *Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique. Application à la reconnaissance de la parole. (optimization through stochastic gradient of modular systems that combine neural networks and dynamic programming, with applications to speech recognition)*. PhD thesis, Université de Paris XI, 91405 Orsay cedex, France.

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

[Driancourt et al., 1991b] Driancourt, X., Bottou, L., and P., G. (1991b). Comparison and cooperation of several classifiers. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*.

[Driancourt and Gallinari, 1992a] Driancourt, X. and Gallinari, P. (1992a). Empirical risk optimisation: neural networks and dynamic programming. In *Proceedings of Neural Networks for Signal Processing (NNSP)*.

[Driancourt and Gallinari, 1992b] Driancourt, X. and Gallinari, P. (1992b). A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron. In *Proceedings of ICASSP*.

[Franzini et al., 1990] Franzini, M., Lee, K. F., and Waibel, A. (1990). Connectionnist viterbi training: A new hybrid method for continuous speech recognition. In *Proceedings of ICASSP*, page 245.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press.

[Haffner, 1993] Haffner, P. (1993). Connectionist speech recognition with a global MMI algorithm. In *EUROSPEECH'93, 3rd European Conference on Speech Communication and Technology*, Berlin.

[Haffner et al., 1991] Haffner, P., Franzini, M., and Waibel, A. H. (1991). Integrating time-alignment and neural networks for high performance continuous speech recognition. In *Proceeding of ICASSP*, pages 105–108. IEEE.

[Haffner and Waibel, 1991] Haffner, P. and Waibel, A. H. (1991). Time-delay neural networks embedding time alignment: a performance analysis. In *EUROSPEECH'91, 2nd European Conference on Speech Communication and Technology*, Genova, Italy.

[Haffner and Waibel, 1992] Haffner, P. and Waibel, A. H. (1992). Multi-state time-delay neural networks for continuous speech recognition. In *Advances in Neural Information Processing Systems*, volume 4, pages 579–588. Morgan Kaufmann, San Mateo.

[Hinton, 2002] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800.

[Huang and LeCun, 2006] Huang, F.-J. and LeCun, Y. (2006). Large-scale learning with svm and convolutional nets for generic object categorization. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press.

[Juang et al., 1997] Juang, B.-H., Chou, W., and Lee, C.-H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265.

[Konig et al., 1996] Konig, Y., Bourlard, H., and Morgan, N. (1996). REMAP: Recursive estimation and maximization of A posteriori probabilities — application to transition-based connectionist speech recognition. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 388–394. The MIT Press.

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

[Kumar and Hebert, 2004] Kumar, S. and Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML)*.

[LeCun and Bengio, 1994] LeCun, Y. and Bengio, Y. (1994). word-level training of a handwritten word recognizer based on convolutional neural networks. In IAPR, editor, *Proc. of the International Conference on Pattern Recognition*, volume II, pages 88–92, Jerusalem. IEEE.

[LeCun et al., 1997] LeCun, Y., Bottou, L., and Bengio, Y. (1997). Reading checks with graph transformer networks. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 151–154, Munich. IEEE.

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

[LeCun et al., 1998b] LeCun, Y., Bottou, L., Orr, G., and Muller, K. (1998b). Efficient backprop. In Orr, G. and K., M., editors, *Neural Networks: Tricks of the trade*. Springer.

[LeCun and Huang, 2005] LeCun, Y. and Huang, F. (2005). Loss functions for discriminative training of energy-based models. In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*.

[Ljolje et al., 1990] Ljolje, A., Ephraim, Y., and Rabiner, L. R. (1990). Estimation of hidden markov model parameters by minimizing empirical error rate. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing*, pages 709–712.

[MacKay, 2003] MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. Available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

[McCallum et al., 2000] McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum entropy markov models for information extraction and segmetnation. In *Proc. International Conference on Machine Learning (ICML)*, pages 591–598.

[McDermott, 1997] McDermott, E. (1997). *Discriminative Training for Speech Recognition*. PhD thesis, Waseda University.

[McDermott and Katagiri, 1992] McDermott, E. and Katagiri, S. (1992). Prototype-based discriminative training for various speech units. In *Proceedings of ICASSP-92, San Francisco, CA, USA*, pages 417–420.

[Mohri, 1997] Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

[Morgan and Bourlard, 1995] Morgan, N. and Bourlard, H. (1995). Continuous speech recognition: An introduction to the hybrid hmm/connectionist approach. *IEEE Signal Processing Magazine*, 12(3):25–42.

[Ning et al., 2005] Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. (2005). Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371. Special issue on Molecular and Cellular Bioimaging, to appear.

[Osadchy et al., 2005] Osadchy, R., Miller, M., and LeCun, Y. (2005). Synergistic face detection and pose estimation with energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press.

[Sakoe et al., 1988] Sakoe, H., Isotani, R., Yoshida, K., Iso, K., and Watanabe, T. (1988). Speaker-independant word recognition using dynamic programming neural networks. In *Proceedings of ICASSP-88, New York*, pages 107–110.

[Solla et al., 1988] Solla, S., Levin, E., and Fleisher, M. (1988). Accelerated learning in layered neural networks. *Complex Systems*, 2(6):625–639.

[Taskar et al., 2003] Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *Proc. NIPS*.

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

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag.

[Vishwanathan et al., 2006] Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*. IMLS/ICML.

[Yedidia et al., 2005] Yedidia, J., Freeman, W., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312.