Homework 3: Energy-Based Models

CSCI-GA 2572 Deep Learning
Fall 2025

The goal of homework 3 is to test your understanding of Energy-Based Models, and to show you one application in structured prediction.

In the theoretical part, we'll mostly test your intuition. You'll need to write brief answers to questions about how EBMs work. In part 2, we will implement a simple optical character recognition system.

In part 1, you should submit all your answers in a pdf file. As before, we recom- mend using LaTeX.

For part 2, you will implement some neural networks by adding your code to the provided `ipynb` file.

As before, please use numerator layout.

The due date of homework 3 is at **10/19**. Submit the following files in a zip file `your_net_id.zip` through NYU Classes:

- `hw3_theory.pdf`
- `hw3_impl.ipynb`

The following behaviors will result in penalty of your final score:

1. 10% penalty for submitting your file without using the correct naming for- mat (including naming the zip file, PDF file or python file wrong, adding extra files in the zip folder, like the testing scripts in your zip file).
2. 20% penalty for late submission within the first 24 hours after the deadline. We will not accept any late submission after the first 24 hours.
3. 20% penalty for code submission that cannot be executed following the steps we mentioned.

1 Theory (50pt)

1.1 Energy-Based Models Intuition (15pts)

This question tests your intuitive understanding of Energy-based models and their properties.

(a) (1pts) How do energy-based models allow for modeling situations where the mapping from input x_i to output y_i is not 1 to 1, but 1 to many?

Energy-based models assign an energy to each pair (x, y). For a fixed x, several y values can simultaneously have low energies (multiple minima), allowing the model to represent one-to-many mappings naturally without forcing a single deterministic output.

(b) (2pts) How do energy-based models differ from models that output probabilities?

An EBM learns an unnormalized *energy function* $F_W(x, y)$, where lower energy means higher compatibility. In contrast, probabilistic models directly output normalized probabilities (e.g., via softmax). EBMs therefore avoid explicit normalization but require estimating or approximating the partition function to obtain probabilities.

(c) (2pts) How can you use energy function $F_W(x, y)$ to calculate a probability $p(y \mid x)$?

A conditional probability can be defined as:

$$p_W(y \mid x) = \frac{\exp[-F_W(x, y)]}{Z_W(x)}, \quad \text{where} \quad Z_W(x) = \sum_{y'} \exp[-F_W(x, y')].$$

For continuous y, the sum becomes an integral. Computing $Z_W(x)$ exactly is usually intractable.

(d) (2pts) What are the roles of the loss function and energy function?

The **energy function** $F_W(x, y)$ defines how compatible an input–output pair is. The **loss function** $\ell(x, y, W)$ shapes this energy landscape during training—it pushes down the energy of correct (positive) pairs and pushes up the energy of incorrect (negative) pairs, often with regularization for smoothness.

(e) (2pts) What problems can be caused by using only positive examples for energy (pushing down energy of correct inputs only)? How can it be avoided?

Training with only positive examples can cause *energy collapse*, where all configurations have low energy and the model loses discriminative power. This is avoided by including negative samples (contrastive learning, hard negative mining) or by adding regularization that penalizes uniformly low energies.

(f) (2pts) Briefly explain the three methods that can be used to shape the energy function.

(i) **Positive shaping:** Decrease $F_W(x, y)$ for real data pairs.

(ii) **Negative shaping:** Increase $F_W(x, \tilde{y})$ or $F_W(\tilde{x}, y)$ for negative examples (contrastive or margin-based losses).

(iii) **Regularization:** Smooth or constrain the energy landscape (e.g., weight decay, gradient penalties, score matching) to improve stability and sampling.

(g) (2pts) Provide an example of a loss function that uses negative examples. The format should be as follows $\ell_{\text{example}}(x, y, W) = F_W(x, y)$.

Margin-based loss:

$$\ell_{\text{hinge}}(x, y, W) = [\, m + F_W(x, y) - F_W(x, \tilde{y}) \,]_+, \quad m > 0.$$

InfoNCE (softmax contrastive) loss:

$$\ell_{\text{NCE}}(x, y, W) = -\log \frac{\exp[-F_W(x, y)]}{\sum_{y' \in \{y\} \cup \mathcal{N}(x)} \exp[-F_W(x, y')]}.$$

(h) (2pts) Say we have an energy function $F(x, y)$ with images x, classification for this image y. Write down the mathematical expression for doing inference given an input x. Now say we have a latent variable z, and our energy is $G(x, y, z)$. What is the expression for doing inference then?

For a given input x, inference corresponds to minimizing the energy:

$$y^*(x) = \arg\min_y F(x, y).$$

When a latent variable z is introduced, the inference can be written as:

$$(y^*, z^*) = \arg\min_{y,z} G(x, y, z) \quad \text{or} \quad y^*(x) = \arg\min_y \min_z G(x, y, z).$$

If y or z are continuous, the minimization is typically performed via gradient descent:

$$z_{t+1} = z_t - \eta \nabla_z G(x, y, z_t),$$

where the optimization searches for the lowest-energy configuration (most compatible with x).

1.2 Negative Log-Likelihood Loss (20 pts)

Let's consider an energy-based model we are training to do classification of input between n classes. $F_W(x, y)$ is the energy of input x and class y. We consider n classes: $y \in \{1, \ldots, n\}$.

(i) (2pts) For a given input x, write down an expression for a Gibbs distribution over labels y that this energy-based model specifies. Use β for the constant multiplier.

For an energy function $F_W(x, y)$ and inverse temperature $\beta > 0$,

$$p_W(y \mid x) = \frac{\exp\big(-\beta\, F_W(x, y)\big)}{Z_W(x)}, \qquad Z_W(x) = \sum_{y'=1}^{n} \exp\big(-\beta\, F_W(x, y')\big).$$

(ii) (5pts) Let's say for a particular data sample x, we have the label y. Give the expression for the negative log likelihood loss, i.e. negative log likelihood of the correct label (show step-by-step derivation of the loss function from the expression of the previous subproblem). For easier calculations in the following subproblem, multiply the loss by $\frac{1}{\beta}$.

For a data pair (x, y), the NLL is

$$\mathcal{L}_{\text{NLL}}(x, y; W) = -\log p_W(y \mid x) = \beta\, F_W(x, y) \;+\; \log Z_W(x).$$

Multiplying by $\frac{1}{\beta}$ (as requested):

$$\tilde{\mathcal{L}}(x, y; W) \;=\; \frac{1}{\beta}\,\mathcal{L}_{\text{NLL}} \;=\; F_W(x, y) \;+\; \frac{1}{\beta}\log\sum_{y'=1}^{n} \exp\big(-\beta\, F_W(x, y')\big)\;.$$

Derivation sketch:

$$-\log p_W(y \mid x) = -\big(-\beta F_W(x, y) - \log Z_W(x)\big) = \beta F_W(x, y) + \log Z_W(x),$$

then divide by β.

(iii) (8pts) Now, derive the gradient of that expression with respect to W (just providing the final expression is not enough). Why can it be intractable to compute it, and how can we get around the intractability?

Write

$$\tilde{\mathcal{L}}(x, y; W) = F_W(x, y) + \frac{1}{\beta} \log Z_W(x), \quad Z_W(x) = \sum_{y'} e^{-\beta F_W(x, y')}.$$

Differentiate term by term:

$$\nabla_W \tilde{\mathcal{L}} = \underbrace{\nabla_W F_W(x, y)}_{\text{“pull down” the true pair}} + \frac{1}{\beta} \frac{1}{Z_W(x)} \nabla_W Z_W(x).$$

Compute $\nabla_W Z_W(x)$:

$$\begin{aligned} \nabla_W Z_W(x) &= \sum_{y'} \nabla_W e^{-\beta F_W(x, y')} && (1) \\ &= \sum_{y'} e^{-\beta F_W(x, y')} \left(-\beta \nabla_W F_W(x, y')\right) && (2) \\ &= -\beta Z_W(x) \, \mathbb{E}_{y' \sim p_W(\cdot|x)} \left[\nabla_W F_W(x, y')\right]. && (3) \end{aligned}$$

Plugging back:

$$\nabla_W \tilde{\mathcal{L}}(x, y; W) = \nabla_W F_W(x, y) \; - \; \mathbb{E}_{y' \sim p_W(\cdot|x)}\big[\nabla_W F_W(x, y')\big].$$

Interpretation: decrease energy of the true pair and increase (on average) the energies of alternatives under the model distribution.

Intractability: computing $p_W(y' \mid x)$ requires $Z_W(x)$, which can be expensive when y is large (structured outputs) or continuous (sum $\rightarrow$ integral). Even evaluating the expectation can require sampling from $p_W(\cdot \mid x)$.

Workarounds: (i) use a small discrete label set (standard classification: exact softmax), (ii) approximate with *negative sampling*/NCE, (iii) MCMC or Langevin dynamics to sample negatives, (iv) importance sampling, (v) surrogate training like score/denoising matching or contrastive losses that avoid exact Z_W.

(iv) (5pts) Explain why negative log-likelihood loss pushes the energy of the correct example to negative infinity, and all others to positive infinity, no matter how close the two examples are, resulting in an energy surface with really sharp edges in case of continuous y (this is usually not an issue for discrete y because there's no distance measure between different classes).

From the unscaled NLL $\mathcal{L} = \beta F(x, y) + \log \sum_k e^{-\beta F(x,k)}$, the partial derivatives w.r.t. energies are

$$\frac{\partial \tilde{\mathcal{L}}}{\partial F(x, y)} = 1 - p_W(y \mid x), \qquad \frac{\partial \tilde{\mathcal{L}}}{\partial F(x, k)} = -\, p_W(k \mid x) \;\text{ for } k \neq y.$$

Thus gradient descent reduces $F(x, y)$ (since $1 - p_W(y \mid x) > 0$ unless $p_W = 1$) and increases other $F(x, k)$ (since $-p_W(k \mid x) < 0$). With no explicit regularization or margin control, the loss keeps improving by pushing $F(x, y) \rightarrow -\infty$ and $F(x, k) \rightarrow +\infty$ to make $p_W(y \mid x) \rightarrow 1$.

In continuous y, this creates extremely sharp cliffs in the energy landscape (large gradients near the decision set). In discrete y this is less problematic because there is no notion of distance between classes, but the same tendency (parameter norms $\to \infty$ under separability) remains unless tempered by regularization, finite temperature (smaller β), early stopping, or label smoothing.

1.3 Comparing Contrastive Loss Functions (15pts)

In this problem, we're going to compare a few contrastive loss functions. We are going to look at the behavior of the gradients, and understand what uses each loss function has. In the following subproblems, m is a margin, $m \in \mathbb{R}$, x is input, y is the correct label, $\bar{y}$ is the incorrect label. Define the loss in the following format: $\ell_{\text{example}}(x, y, \bar{y}, W) = F_W(x, y)$.

(a) (3pts) Simple loss function is defined as follows:

$$\ell_{\text{simple}}(x, y, \bar{y}, W) = [F_W(x, y)]_+ + [m - F_W(x, \bar{y})]_+$$

Assuming we know the derivative $\frac{\partial F_W(x,y)}{\partial W}$ for any x, y, give an expression for the partial derivative of the ℓ_{simple} with respect to W.

$$\ell_{\text{simple}}(x, y, \bar{y}, W) = [F_y]_+ + [\, m - F_{\bar{y}} \,]_+.$$

Using $\frac{d}{dW}[u]_+ = \mathbf{1}\{u > 0\}\frac{du}{dW}$,

$$\frac{\partial \ell_{\text{simple}}}{\partial W} = \mathbf{1}\{F_y > 0\}\,\frac{\partial F_y}{\partial W} \;-\; \mathbf{1}\{m - F_{\bar{y}} > 0\}\,\frac{\partial F_{\bar{y}}}{\partial W}.$$

(b) (3pts) Log loss is defined as follows:

$$\ell_{\log}(x, y, \bar{y}, W) = \log\left(1 + e^{F_W(x,y) - F_W(x,\bar{y})}\right)$$

Assuming we know the derivative $\frac{\partial F_W(x,y)}{\partial W}$ for any x, y, give an expression for the partial derivative of the $\ell_{\log}$ with respect to W.

$$\ell_{\log}(x, y, \bar{y}, W) = \log\big(1 + e^{\,F_y - F_{\bar{y}}}\big).$$

Let $\Delta = F_y - F_{\bar{y}}$. Then

$$\frac{\partial \ell_{\log}}{\partial \Delta} = \frac{e^{\Delta}}{1 + e^{\Delta}} = \sigma(\Delta), \quad \text{where } \sigma(t) = \frac{1}{1 + e^{-t}}.$$

By chain rule,

$$\frac{\partial \ell_{\log}}{\partial W} = \sigma(F_y - F_{\bar{y}})\left(\frac{\partial F_y}{\partial W} - \frac{\partial F_{\bar{y}}}{\partial W}\right).$$

(c) (3pts) Square-Square loss is defined as follows:

$$\ell_{\text{square-square}}(x, y, \bar{y}, W) = \big([F_W(x, y)]_+\big)^2 + \big([m - F_W(x, \bar{y})]_+\big)^2$$

Assuming we know the derivative $\frac{\partial F_W(x,y)}{\partial W}$ for any x, y, give an expression for the partial derivative of the $\ell_{\text{square-square}}$ with respect to W.

$$\ell_{\text{square-square}}(x, y, \bar{y}, W) = \big([F_y]_+\big)^2 + \big([\, m - F_{\bar{y}}\,]_+\big)^2.$$

Since $\frac{d}{dW}([u]_+)^2 = 2[u]_+ \, \mathbf{1}\{u > 0\} \frac{du}{dW} = 2u \, \mathbf{1}\{u > 0\} \frac{du}{dW}$,

$$\frac{\partial \ell_{\text{square-square}}}{\partial W} = 2F_y \, \mathbf{1}\{F_y > 0\} \, \frac{\partial F_y}{\partial W} \; - \; 2(m - F_{\bar{y}}) \, \mathbf{1}\{m - F_{\bar{y}} > 0\} \, \frac{\partial F_{\bar{y}}}{\partial W}.$$

(d) (6pts) Comparison.

(i) (2pts) Explain how NLL loss is different from the three losses above.

NLL arises from a normalized Gibbs model and couples *all* labels via the partition function; its gradient is $\nabla F_y - \mathbb{E}_{y' \sim p(\cdot|x)}[\nabla F_{y'}]$. The three losses above are *pairwise contrastive*: they compare a positive to sampled negatives (no global normalization), trading probabilistic calibration for simpler, local objectives.

(ii) (2pts) The hinge loss $[F_W(x, y) - F_W(x, \bar{y}) + m]_+$ has a margin parameter m, which gives 0 loss when the positive and negative examples have energy that are m apart. The log loss is sometimes called a "soft-hinge" loss. Why? What is the advantage of using a soft hinge loss?

$\log(1 + e^{\Delta}) = \text{softplus}(\Delta)$ is a smooth upper bound/approximation to $[\Delta]_+$ (hinge with m=0). Its advantage is smooth, nonzero gradients everywhere (via $\sigma(\Delta)$), which improves optimization stability and avoids the zero-gradient plateaus of the hard hinge near the margin.

(iii) (2pts) How are the simple loss and square-square loss different from the hinge/log loss? In what situations would you use the simple loss, and in what situations would you use the square-square loss?

Simple and Square–Square penalize *absolute* violations for the positive ($F_y \leq 0$) and the negative ($F_{\bar{y}} \geq m$) *separately*. Hinge/Log act on the *difference* $F_y - F_{\bar{y}}$ (relative separation). Use **Simple** when you want sparse, margin-style updates with constant gradients in the violating regions. Use **Square–Square** when large violations should be penalized more strongly and you want smoother gradients that grow with the violation magnitude (useful for stability and faster correction of big mistakes).

2 Implementation (50pt)

Please add your solutions to this notebook `hw3_impl.ipynb`. Plase use your NYU account to access the notebook. The notebook contains parts marked as **TODO**, where you should put your code or explanations. The notebook is a Google Colab notebook; you should copy it to your drive, add your solutions, and then download and submit it to NYU Classes. You're also free to run it on any other machine, as long as the version you send us can be run on Google Colab.

Link: Google Drive Notebook