# CSCI-GA 2572: Deep Learning, Fall 2025

Energy-Based Models

Aditeya Baral
N19186654

## Problem 1.1: Energy-Based Models Intuition

### Problem a

Solution

Energy-based models (EBMs) naturally handle one-to-many mappings from input $x_i$ to output $y_i$ by fundamentally changing how we represent the relationship between inputs and outputs. Instead of learning a direct mapping function $f : x \to y$ that produces a single output for each input, EBMs learn an energy function $F_W(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a scalar energy value (a measure of cost) to every possible input-output pair $(x, y)$.

For a given input $x$, the energy function evaluates all possible outputs $y \in \mathcal{Y}$. Multiple different outputs can simultaneously have low energy values, indicating that they are all compatible or valid predictions for the same input $x$. The set of valid outputs corresponds to the minima (or low-energy regions) of the energy landscape:

$$\mathcal{Y}^*(x) = \arg\min_{y \in \mathcal{Y}} F_W(x, y)$$

This set $\mathcal{Y}^*(x)$ can contain multiple elements, thus capturing the one-to-many relationship between input and output.

### Problem b

Solution

Energy-based models differ from probabilistic models in several fundamental ways:

- ***Output Representation***
  - **Probabilistic models** directly output normalized probability distributions $p(y|x)$ where $\sum_y p(y|x) = 1$ (discrete case) or $\int p(y|x)dy = 1$ (continuous case). These probabilities must satisfy non-negativity ($p(y|x) \geq 0$) and normalization constraints.
  - **Energy-based models** output unnormalized energy values $F_W(x, y) \in \mathbb{R}$ that measure the compatibility or cost of an input-output pair. Energy values can be any real number (positive, negative, or zero) and do not need to satisfy normalization constraints.
- ***Normalization Requirements***
  - **Probabilistic models** require computing the normalization constant through a partition function across all possible outputs, which can be computationally intractable for high-dimensional or continuous output spaces.

- **Energy-based models** avoid explicit normalization during training and inference. We can perform inference by simply finding the most compatible output $y^* = \arg\min_y F_W(x, y)$, which does not require evaluating or normalizing over all possible $y$.

- ***Interpretation***
  - **Probabilistic models** provide calibrated uncertainty estimates: the probability values can be interpreted as confidence levels.
  - **Energy-based models** provide relative compatibility scores: lower energy means better compatibility, but the absolute energy values of $F_W(x, y)$ don't directly represent probabilities or confidence without additional conversion (e.g., via the Boltzmann distribution).

- ***Flexibility***. Energy-based models offer more flexibility as they don't require maintaining probability constraints (normalization or non-negativity) during optimization, making them suitable for complex, multi-modal distributions where normalization is difficult.

## Problem c

Solution

To convert an energy function $F_W(x, y)$ into a probability distribution $p(y|x)$, we use the *Boltzmann distribution* as:

$$\boxed{p(y|x) = \frac{e^{-\beta F_W(x,y)}}{Z(x)}}$$

where:

- $\beta > 0$ is a temperature parameter (usually set to $\beta = 1$ for simplicity)
- $Z(x)$ is the partition function that normalizes the distribution:

$$Z(x) = \sum_{y'} e^{-\beta F_W(x,y')} \quad \text{(discrete case)}$$

or

$$Z(x) = \int_{y'} e^{-\beta F_W(x,y')} dy' \quad \text{(continuous case)}$$

The exponential $e^{-\beta F_W(x,y)}$ converts energy to an unnormalized probability: lower energy values result in higher probabilities. The partition function $Z(x)$ ensures that $\sum_y p(y|x) = 1$ or $\int p(y|x)dy = 1$, making it a valid probability distribution. The temperature parameter $\beta$ controls the sharpness of the distribution: larger $\beta$ makes the distribution more peaked around low-energy values, while smaller $\beta$ makes it more uniform.

This formulation satisfies all probability axioms: $p(y|x) \geq 0$ (since exponential is always positive) and the normalization constraint (by construction through $Z(x)$).

## Problem d

Solution

The energy function and loss function serve distinct but complementary roles in energy-based models:

***Energy Function*** $F_W(x,y)$:

- *Role:* Measures the compatibility or cost between an input $x$ and output $y$ pair. It assigns a scalar value to every possible $(x,y)$ configuration.
- *Purpose:* Defines the model's predictions by encoding which outputs are compatible (low energy) or incompatible (high energy) with a given input.
- *Use at inference:* Used directly to make predictions by finding outputs that minimize energy: $y^* = \arg\min_y F_W(x,y)$.
- *Learned object:* The energy function itself is what we want to learn - it encodes the model's knowledge about the task.

***Loss Function*** $\mathcal{L}(x,y,W)$:

- *Role:* Defines the training objective that shapes the energy function during learning. It specifies how to update the parameters $W$ based on training data.
- *Purpose:* Ensures the energy function learns the desired behavior by typically pushing down energy of correct outputs and pushing up energy of incorrect outputs.
- *Use at training:* Used to compute gradients $\frac{\partial \mathcal{L}}{\partial W}$ for parameter updates during optimization.
- *Design tool:* The loss function is designed by us to shape the energy landscape appropriately.

***Relationship***: The loss function is constructed using the energy function (e.g., $\mathcal{L}(x,y,W) = F_W(x,y) +$ regularization terms) but serves a different purpose: the energy function is the *model* we're learning, while the loss function is the *training criterion* that shapes how we learn it.

## Problem e

Solution

When training only on positive examples and thus only pushing down the energy of correct input-output pairs $(x,y)$, the following problems arise:

1. **Energy collapse**. The model can trivially minimize the loss by making the energy function output arbitrarily negative values (approaching $-\infty$) for all inputs, regardless of whether they are correct or incorrect. This results in a degenerate solution where $F_W(x,y) \to -\infty$ for all $y$.
2. **No discrimination**: Without pushing up the energy of incorrect outputs, the model cannot distinguish between correct and incorrect predictions. All outputs would have similarly low energy, making inference meaningless.
3. **Flat energy landscape:** The energy surface becomes flat or uniformly low across the output space, providing no useful gradient information for finding the optimal output during inference.
4. **Unbounded parameters:** Parameter values $W$ can grow without bound as the optimization tries to push energies lower and lower, leading to numerical instability and poor generalization.

***How to avoid these problems***.

These issues can be avoided by ensuring the energy function is properly shaped through one or more of the following methods:

1. **Use negative examples (contrastive methods):** Explicitly push up the energy of incorrect outputs while pushing down correct ones. This creates contrast in the energy landscape. Example: contrastive loss $\mathcal{L} = F_W(x, y^+) + \max(0, m - F_W(x, y^-))$ where $y^+$ is correct and $y^-$ is incorrect, with margin $m$.
2. **Regularization:** Add regularization terms to the loss function that penalize extreme energy values or large parameter magnitudes, constraining the energy function. Example: $\mathcal{L} = F_W(x, y) + \lambda \|W\|^2$.
3. **Add constraints:** Design the energy function architecture to be naturally bounded or normalized, preventing collapse (e.g., using output layers with limited range). We can also ensure that the volume of low-energy regions remains limited, preventing the entire space from having low energy.

The key principle is that training must shape the energy landscape by both pushing down energy where needed (correct outputs) and pushing up energy where needed (incorrect outputs), creating proper contrast for meaningful predictions.

## Problem f

Solution

There are three main methods to shape the energy function during training:

1. **Contrastive Methods**. These methods explicitly create contrast in the energy landscape by pushing down the energy of training samples while pushing up the energy of suitably-generated contrastive samples. Different contrastive approaches vary in which points to push up:
   - Pushing up on all possible outputs except training data (e.g., maximum likelihood)
   - Pushing up on chosen locations (e.g., select negative samples, contrastive divergence, GANs)
   - Training functions to map off-manifold points to on-manifold points (e.g., denoising auto-encoders).

   All of these are strategies to generate effective negative/contrastive samples.
2. **Regularization Methods**. These methods add regularization terms to the loss function to prevent energy collapse by limiting extreme energy values or constraining parameter magnitudes. This includes:
   - Adding regularization penalties: $\mathcal{L} = F_W(x, y) + \lambda R(W)$ where $R(W)$ could be $\|W\|^2$ (weight decay) or other penalties that penalize extreme energy values or large parameter magnitudes
   - Measuring and constraining the volume of low-energy space (e.g., sparse coding, variational auto-encoders)
3. **Architectural Methods**. These methods constrain the energy function through its architecture to prevent energy collapse by limiting the information capacity of the representation or naturally bounding the energy values. The structure of the model itself restricts the energy landscape to meaningful shapes. This includes:

- Designing architectures with bounded outputs (e.g., using activation functions with limited range)
- Building architectures so that the volume of low-energy space is naturally bounded (e.g., PCA, K-means, normalizing flows)
- Constraining the volume of low-energy regions through architectural design
- Normalizing energy values across the output space

These methods are used to shape the energy landscape so that low energy corresponds to correct outputs and high energy corresponds to incorrect outputs, enabling meaningful inference and learning.

## Problem g

Solution

An example of a loss function that uses negative examples is the **hinge loss** or margin loss:

$$\boxed{\begin{aligned}\ell_{\text{example}}(x, y, W) &= \max(0,\, m + F_W(x, y^+) - F_W(x, y^-)) \\ &= [\, m + F_W(x, y^+) - F_W(x, y^-)\,]^+\end{aligned}}$$

where:

- where $[z]^+ = \max(0, z)$
- $y^+$ is the correct (positive) output for input $x$
- $y^-$ is an incorrect (negative) output sampled from the output space
- $m > 0$ is a margin hyperparameter that controls the minimum energy gap between positive and negative examples

*Explanation*

- A penalty is applied only when $m + F_W(x, y^+) - F_W(x, y^-) > 0$, i.e., when the energy of the correct output $F_W(x, y^+)$ is not sufficiently lower than the energy of the negative output $F_W(x, y^-)$.
- Equivalently, the loss is zero when $F_W(x, y^+) < F_W(x, y^-) - m$, meaning the correct output has energy at least $m$ lower than the negative output.
- This encourages a margin of separation of at least $m$ between the energies: the correct output should have lower energy than the negative output by at least the margin $m$.
- This formulation simultaneously pushes down the energy of correct outputs and pushes up the energy of negative outputs when the margin constraint is violated.

## Problem h

Solution

***Part 1: Inference without latent variable***

Given an energy function $F(x, y)$ where $x$ is an image and $y$ is its classification, inference is performed by finding the output $y$ that minimizes the energy for a given input $x$:

$$\boxed{y^*(x) = \arg\min_y F(x, y)}$$

This finds the classification $y^*$ that has the lowest energy (highest compatibility) with the input image $x$.

***Part 2: Inference with latent variable***

When we have a latent variable $z$ and energy function $G(x, y, z)$, inference requires minimizing over both the output $y$ and the latent variable $z$:

$$\boxed{y^*(x) = \arg\min_y \min_z G(x, y, z)}$$

or equivalently:

$$\boxed{y^*(x) = \arg\min_y \left[\min_z G(x, y, z)\right] = \arg\min_y E_y(x)}$$

where $E_y(x) = \min_z G(x, y, z)$ is the effective energy of output $y$ after optimizing over the latent variable $z$. The latent variable $z$ represents hidden or unobserved factors that influence the relationship between $x$ and $y$. For each candidate output $y$, we first find the optimal latent variable $z^*(y) = \arg\min_z G(x, y, z)$ that minimizes the energy for that particular $y$. Then we select the output $y^*$ that achieves the lowest energy among all candidates after optimizing over $z$.

## Problem 1.2: Negative log-likelihood loss

### Problem i

Solution

For a given input $x$, the Gibbs distribution (also known as the Boltzmann distribution) over labels $y \in \{1, ..., n\}$ specified by the energy-based model with energy function $F_W(x, y)$ is:

$$\boxed{p(y|x) = \frac{e^{-\beta F_W(x,y)}}{\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}}}$$

where:

- $\beta > 0$ is the constant multiplier (inverse temperature parameter)
- The numerator $e^{-\beta F_W(x,y)}$ converts the energy of class $y$ to an unnormalized probability
- The denominator $\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$ is the partition function $Z(x)$ that normalizes the distribution over all $n$ classes

This distribution assigns higher probability to classes with lower energy (better compatibility with input $x$), and ensures that $\sum_{y=1}^{n} p(y|x) = 1$, making it a valid probability distribution over the $n$ classes.

### Problem ii

Solution

From the problem (i), we have:

$$p(y|x) = \frac{e^{-\beta F_W(x,y)}}{\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}}$$

where $Z_W(x) = \sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$ is the partition function (normalizer). Taking logarithm on both sides:

$$\log p(y|x) = \log \left( \frac{e^{-\beta F_W(x,y)}}{\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}} \right)$$

$$\log p(y|x) = \log \left( e^{-\beta F_W(x,y)} \right) - \log \left( \sum_{y'=1}^{n} e^{-\beta F_W(x,y')} \right)$$

$$\log p(y|x) = -\beta F_W(x,y) - \log \left( \sum_{y'=1}^{n} e^{-\beta F_W(x,y')} \right)$$

*Computing negative log-likelihood*:

The negative log-likelihood is:

$$-\log p(y|x) = \beta F_W(x,y) + \log \left( \sum_{y'=1}^{n} e^{-\beta F_W(x,y')} \right)$$

*Multiplying by* $\frac{1}{\beta}$

$$\mathcal{L}(x,y,W) = \frac{1}{\beta}\left[\beta F_W(x,y) + \log\left(\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\right)\right]$$

which simplifies to:

$$\boxed{\mathcal{L}(x,y,W) = F_W(x,y) + \frac{1}{\beta}\log\left(\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\right)}$$

This is the negative log-likelihood loss for an input $x$ and the correct label $y$, scaled by $\frac{1}{\beta}$.

## Problem iii

Solution

From part (ii), we have:

$$\mathcal{L}(x,y,W) = F_W(x,y) + \frac{1}{\beta}\log\left(\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\right)$$

Taking the gradient with respect to $W$:

$$\frac{\partial \mathcal{L}(x,y,W)}{\partial W} = \frac{\partial F_W(x,y)}{\partial W} + \frac{1}{\beta}\frac{\partial}{\partial W}\log\left(\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\right)$$

*Computing the gradient of the second term*

For the second term, we use the chain rule. Let $Z = \sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$:

$$\frac{\partial}{\partial W}\log(Z) = \frac{1}{Z}\frac{\partial Z}{\partial W}$$

*Computing the gradient of the partition function*

$$\frac{\partial Z}{\partial W} = \frac{\partial}{\partial W}\sum_{y'=1}^{n} e^{-\beta F_W(x,y')} = \sum_{y'=1}^{n}\frac{\partial}{\partial W}e^{-\beta F_W(x,y')}$$

Using the chain rule:

$$\frac{\partial}{\partial W}e^{-\beta F_W(x,y')} = e^{-\beta F_W(x,y')}\cdot(-\beta)\frac{\partial F_W(x,y')}{\partial W}$$

Therefore:

$$\frac{\partial Z}{\partial W} = -\beta\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\frac{\partial F_W(x,y')}{\partial W}$$

*Substituting back*

$$\frac{\partial}{\partial W}\log(Z) = \frac{1}{Z}\cdot\left(-\beta\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}\frac{\partial F_W(x,y')}{\partial W}\right)$$

$$= -\beta\sum_{y'=1}^{n}\frac{e^{-\beta F_W(x,y')}}{\sum_{y''=1}^{n} e^{-\beta F_W(x,y'')}}\frac{\partial F_W(x,y')}{\partial W}$$

Recognizing that $\frac{e^{-\beta F_W(x,y')}}{\sum_{y''=1}^{n} e^{-\beta F_W(x,y'')}} = p(y'|x)$:

$$= -\beta \sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W}$$

*Combining all terms*

$$\frac{\partial \mathcal{L}(x,y,W)}{\partial W} = \frac{\partial F_W(x,y)}{\partial W} + \frac{1}{\beta} \cdot \left( -\beta \sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W} \right)$$

*Simplifying*

$$\boxed{\frac{\partial \mathcal{L}(x,y,W)}{\partial W} = \frac{\partial F_W(x,y)}{\partial W} - \sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W}}$$

***Why is it intractable?***
The gradient consists of two terms:

- *Positive phase:* $\frac{\partial F_W(x,y)}{\partial W}$ - when used in gradient descent, this term pushes down the energy of the correct label $y$
- *Negative phase:* $-\sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W}$ - pushes up the energy across all classes weighted by their model probabilities, creating contrast with the correct label

The computation of $\sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W}$ requires:

1. Computing $p(y'|x)$ for all $n$ classes, which involves evaluating the partition function $Z(x) = \sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$
2. Computing the energy gradient $\frac{\partial F_W(x,y')}{\partial W}$ for all $n$ classes
3. When $n$ is very large (e.g., millions of classes), this summation becomes computationally prohibitive

***How to get around the intractability***:
We can approximate the intractable second term using sampling methods such as Monte Carlo (MC), Markov Chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC), or Contrastive Divergence (CD). Instead of computing the exact expectation over all classes, we sample $\hat{y}$ from the model distribution $p(y'|x)$ and approximate:

$$\boxed{\frac{\partial \mathcal{L}(x,y,W)}{\partial W} \approx \frac{\partial F_W(x,y)}{\partial W} - \frac{\partial F_W(x,\hat{y})}{\partial W}}$$

where $\hat{y} \sim p(y'|x)$ is a sample from the model's predicted distribution.

This replaces the expensive summation over all $n$ classes with a single sample, making the computation tractable. Multiple samples can be used for better approximation:

$$\frac{\partial \mathcal{L}(x,y,W)}{\partial W} \approx \frac{\partial F_W(x,y)}{\partial W} - \frac{1}{k} \sum_{i=1}^{k} \frac{\partial F_W(x,\hat{y}_i)}{\partial W}$$

where $\hat{y}_1, ..., \hat{y}_k$ are $k$ samples from $p(y'|x)$.

## Problem iv

Solution

The negative log-likelihood loss creates sharp edges in the energy surface due to its structure and optimization behavior. From problem (ii), the negative log-likelihood loss is:

$$\mathcal{L}(x, y, W) = F_W(x, y) + \frac{1}{\beta} \log \left( \sum_{y'=1}^{n} e^{-\beta F_W(x, y')} \right)$$

And from problem (iii), the gradient is:

$$\frac{\partial \mathcal{L}(x, y, W)}{\partial W} = \frac{\partial F_W(x, y)}{\partial W} - \sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x, y')}{\partial W}$$

where $p(y'|x) = \frac{e^{-\beta F_W(x,y')}}{\sum_{y''=1}^{n} e^{-\beta F_W(x,y'')}}$.

***Why the loss pushes energies to extremes***

1. **Pushing correct example energy to $-\infty$**

   The first term $\frac{\partial F_W(x,y)}{\partial W}$ always contributes to pushing down the energy of the correct example $y$. There is no mechanism in the loss function to stop this descent. As long as $F_W(x, y)$ can decrease, the loss $\mathcal{L}$ will also decrease (since the first term directly adds $F_W(x, y)$). Therefore, optimization will continue to push $F_W(x, y) \to -\infty$.

2. **Pushing all other examples to $+\infty$**

   The second term in the gradient, $-\sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x,y')}{\partial W}$, pushes up the expected energy of all classes weighted by their probabilities. To minimize the loss, the partition function term $\log \sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$ should be as small as possible. This is achieved when all incorrect examples $y' \neq y$ have very high energy, causing $e^{-\beta F_W(x,y')} \to 0$ for all $y' \neq y$. Thus, optimization drives $F_W(x, y') \to +\infty$ for all $y' \neq y$.

3. **No saturation mechanism**

   The NLL loss has no built-in mechanism to stop when a sufficient separation between correct and incorrect examples is achieved. Unlike margin-based losses (e.g., hinge loss with margin $m$) that stop penalizing once the margin is satisfied, NLL continues to increase the energy gap indefinitely.

***Problem for continuous*** $y$

For continuous output spaces, this behavior creates sharp edges in the energy surface. Consider two examples $y_1$ (correct) and $y_2$ (incorrect) that are very close in the output space such that $\|y_1 - y_2\| \approx 0$. Despite their proximity, NLL loss will still push $F_W(x, y_1) \to -\infty$ and $F_W(x, y_2) \to +\infty$. This creates an *infinitely sharp discontinuity in the energy function at the boundary between correct and incorrect outputs*, regardless of how close they are in the output space. The energy surface has extremely steep gradients near the data manifold, making it:

- Difficult to optimize (numerical instability)
- Poorly generalizable (no smooth interpolation)
- Unrealistic for many real-world problems where nearby outputs should have similar energies

***Why this is not an issue for discrete*** $y$

For discrete classification (where $y \in \{1, ..., n\}$), there is no notion of ”distance” or ”closeness” between different classes since they are categorical entities without a meaningful metric. Each class is treated as an independent, distinct entity. Sharp transitions between classes are acceptable and even desirable. The energy function thus only needs to *assign relative preferences among a finite set of discrete options*, not create a smooth continuous landscape.

Therefore, while NLL is problematic for continuous outputs (e.g., regression, image generation), it works well for discrete classification where sharp distinctions between classes are appropriate.

## Problem 1.3: Comparing Contrastive Loss Functions

### Problem a

Solution

We need to find

$$\frac{\partial l_{simple}(x, y, \overline{y}, W)}{\partial W}$$

where

$$l_{simple}(x, y, \overline{y}, W) = [F_W(x, y)]^+ + [m - F_W(x, \overline{y})]^+$$

We know that $[z]^+ = \max(0, z)$, so we can write:

$$l_{simple}(x, y, \overline{y}, W) = \max(0, F_W(x, y)) + \max(0, m - F_W(x, \overline{y}))$$

The derivative of $\max(0, z)$ with respect to its argument is:

$$\frac{d}{dz}\max(0, z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z < 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$

This can be written using the indicator function: $\nVdash_{z>0}$.

*Applying the chain rule*:

For the first term $[F_W(x, y)]^+$:

$$\frac{\partial}{\partial W}[F_W(x, y)]^+ = \nVdash_{F_W(x,y)>0} \cdot \frac{\partial F_W(x, y)}{\partial W}$$

For the second term $[m - F_W(x, \overline{y})]^+$:

$$\frac{\partial}{\partial W}[m - F_W(x, \overline{y})]^+ = \nVdash_{m-F_W(x,\overline{y})>0} \cdot \frac{\partial}{\partial W}(m - F_W(x, \overline{y}))$$

$$= \nVdash_{F_W(x,\overline{y})<m} \cdot \left(-\frac{\partial F_W(x, \overline{y})}{\partial W}\right)$$

*Combining the terms*:

$$\boxed{\frac{\partial l_{simple}(x, y, \overline{y}, W)}{\partial W} = \nVdash_{F_W(x,y)>0} \cdot \frac{\partial F_W(x, y)}{\partial W} - \nVdash_{F_W(x,\overline{y})<m} \cdot \frac{\partial F_W(x, \overline{y})}{\partial W}}$$

where:

- $\nVdash_{F_W(x,y)>0}$ is 1 if $F_W(x, y) > 0$ and 0 otherwise
- $\nVdash_{F_W(x,\overline{y})<m}$ is 1 if $F_W(x, \overline{y}) < m$ and 0 otherwise

## Problem b

Solution

We need to find

$$\frac{\partial l_{log}(x, y, \overline{y}, W)}{\partial W}$$

where

$$l_{log}(x, y, \overline{y}, W) = \log(1 + e^{F_W(x,y) - F_W(x,\overline{y})})$$

*Applying the chain rule*:

Let $u = F_W(x, y) - F_W(x, \overline{y})$. Then:

$$l_{log} = \log(1 + e^u)$$

The derivative is:

$$\frac{\partial l_{log}}{\partial W} = \frac{\partial}{\partial W} \log(1 + e^u)$$

Using the chain rule:

$$\frac{\partial}{\partial W} \log(1 + e^u) = \frac{1}{1 + e^u} \cdot \frac{\partial}{\partial W}(1 + e^u)$$

$$= \frac{1}{1 + e^u} \cdot e^u \cdot \frac{\partial u}{\partial W}$$

$$= \frac{e^u}{1 + e^u} \cdot \frac{\partial u}{\partial W}$$

*Computing* $\frac{\partial u}{\partial W}$:

$$\frac{\partial u}{\partial W} = \frac{\partial}{\partial W}[F_W(x, y) - F_W(x, \overline{y})]$$

$$= \frac{\partial F_W(x, y)}{\partial W} - \frac{\partial F_W(x, \overline{y})}{\partial W}$$

*Combining the results*:

$$\frac{\partial l_{log}}{\partial W} = \frac{e^u}{1 + e^u} \cdot \left( \frac{\partial F_W(x, y)}{\partial W} - \frac{\partial F_W(x, \overline{y})}{\partial W} \right)$$

Substituting back $u = F_W(x, y) - F_W(x, \overline{y})$:

$$\frac{\partial l_{log}}{\partial W} = \frac{e^{F_W(x,y) - F_W(x,\overline{y})}}{1 + e^{F_W(x,y) - F_W(x,\overline{y})}} \cdot \left( \frac{\partial F_W(x, y)}{\partial W} - \frac{\partial F_W(x, \overline{y})}{\partial W} \right)$$

This can be simplified by recognizing that $\frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} = \sigma(z)$, where $\sigma$ is the sigmoid function:

$$\boxed{\frac{\partial l_{log}(x, y, \overline{y}, W)}{\partial W} = \sigma(F_W(x, y) - F_W(x, \overline{y})) \cdot \left( \frac{\partial F_W(x, y)}{\partial W} - \frac{\partial F_W(x, \overline{y})}{\partial W} \right)}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function.

## Problem c

Solution

We need to find
$$\frac{\partial l_{square-square}(x, y, \overline{y}, W)}{\partial W}$$
where
$$l_{square-square}(x, y, \overline{y}, W) = ([F_W(x, y)]^+)^2 + ([m - F_W(x, \overline{y})]^+)^2$$
We know that $[z]^+ = \max(0, z)$, so we can write:
$$l_{square-square}(x, y, \overline{y}, W) = (\max(0, F_W(x, y)))^2 + (\max(0, m - F_W(x, \overline{y})))^2$$

*Applying the chain rule*:

For a function of the form $(\max(0, z))^2$, we use the chain rule. The derivative of $(\max(0, z))^2$ with respect to its argument is:
$$\frac{d}{dz}(\max(0, z))^2 = \begin{cases} 2z & \text{if } z > 0 \\ 0 & \text{if } z < 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$
This can be written as: $2[z]^+ \cdot \nVdash_{z>0}$ or equivalently $2\max(0, z) \cdot \nVdash_{z>0}$. Thus:
$$\frac{d}{dz}(\max(0, z))^2 = 2\max(0, z) \cdot \frac{d}{dz}\max(0, z) = 2\max(0, z) \cdot \nVdash_{z>0}$$
This simplifies to:
$$\frac{d}{dz}(\max(0, z))^2 = 2[z]^+ \cdot \nVdash_{z>0}$$
For the first term $([F_W(x, y)]^+)^2$:
$$\frac{\partial}{\partial W}([F_W(x, y)]^+)^2 = 2[F_W(x, y)]^+ \cdot \nVdash_{F_W(x,y)>0} \cdot \frac{\partial F_W(x, y)}{\partial W}$$
For the second term $([m - F_W(x, \overline{y})]^+)^2$:
$$\frac{\partial}{\partial W}([m - F_W(x, \overline{y})]^+)^2 = 2[m - F_W(x, \overline{y})]^+ \cdot \nVdash_{m-F_W(x,\overline{y})>0} \cdot \frac{\partial}{\partial W}(m - F_W(x, \overline{y}))$$
$$= 2[m - F_W(x, \overline{y})]^+ \cdot \nVdash_{F_W(x,\overline{y})<m} \cdot \left(-\frac{\partial F_W(x, \overline{y})}{\partial W}\right)$$
*Combining the terms*:
$$\boxed{\frac{\partial l_{square-square}(x, y, \overline{y}, W)}{\partial W} = 2[F_W(x, y)]^+ \cdot \nVdash_{F_W(x,y)>0} \cdot \frac{\partial F_W(x, y)}{\partial W} - 2[m - F_W(x, \overline{y})]^+ \cdot \nVdash_{F_W(x,\overline{y})<m} \cdot \frac{\partial F_W(x, \overline{y})}{\partial W}}$$
where:

- $\nVdash_{F_W(x,y)>0}$ is 1 if $F_W(x, y) > 0$ and 0 otherwise
- $\nVdash_{F_W(x,\overline{y})<m}$ is 1 if $F_W(x, \overline{y}) < m$ and 0 otherwise

## Problem d

### Problem i

Solution

The NLL (Negative Log-Likelihood) loss differs from the three losses above (simple, log, and square-square) in several ways:

1. **Requires summation/integration over all possible outputs**

   From Problem 1.2 (ii), the NLL loss is:

$$\mathcal{L}_{NLL}(x, y, W) = F_W(x, y) + \frac{1}{\beta} \log \left( \sum_{y'=1}^{n} e^{-\beta F_W(x, y')} \right)$$

   The NLL loss requires computing the partition function $\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}$, which sums over *all possible classes*. In contrast, the three contrastive losses (simple, log, square-square) only involve:

   - The correct label $y$
   - A single incorrect label $\overline{y}$

   This makes NLL *computationally intractable for large output spaces*, while the contrastive losses scale with the number of negative samples chosen.

2. **Gradient structure**

   The gradient of NLL loss (from Section 1.2(iii)) is:

$$\frac{\partial \mathcal{L}_{NLL}}{\partial W} = \frac{\partial F_W(x, y)}{\partial W} - \sum_{y'=1}^{n} p(y'|x) \frac{\partial F_W(x, y')}{\partial W}$$

   This includes an *expectation over all classes* (the negative phase). In contrast, the contrastive losses yield gradients that *depend only on the positive example* $y$ and the sampled negative $\overline{y}$, without requiring computation over the entire output space.

3. **Global vs. pairwise comparison**

   The three losses above are *explicitly pairwise-contrastive*: they directly compare the energy of the correct label $y$ with a specific incorrect label $\overline{y}$. The loss depends on the *energy difference or margin* between these two examples. However, NLL is *implicitly global-contrastive*: it compares the correct label $y$ against *all possible classes* simultaneously through the partition function.

4. **Probabilistic interpretation**

   NLL loss corresponds to maximizing the likelihood of the data under a Gibbs/Boltzmann distribution:

$$p(y|x) = \frac{e^{-\beta F_W(x,y)}}{\sum_{y'=1}^{n} e^{-\beta F_W(x,y')}}$$

   *Minimizing NLL is equivalent to maximum likelihood estimation.* The contrastive losses (simple, log, square-square) do not have this direct probabilistic interpretation.

5. **Saturation behavior**

   The contrastive losses have mechanisms to limit *how much* the correct label's energy is pushed:

   - Simple loss: $[F_W(x, y)]^+$ only penalizes when $F_W(x, y) > 0$

- Log loss: penalty diminishes as $F_W(x,y) - F_W(x,\overline{y})$ becomes more negative
- Square-square loss: $[F_W(x,y)]^+$ only penalizes when $F_W(x,y) > 0$

NLL loss has no such saturation mechanism as it will continue to push $F_W(x,y) \to -\infty$ indefinitely and all other energies up indefinitely, as discussed in Problem 1.2(iv).

**Problem ii**

Solution

***Comparison with hinge loss***
The hinge loss is:

$$l_{hinge} = [F_W(x,y) - F_W(x,\overline{y}) + m]^+ = \max(0, F_W(x,y) - F_W(x,\overline{y}) + m)$$

- **Hard cutoff**. When $F_W(x,y) - F_W(x,\overline{y}) + m \leq 0$ (i.e., the margin is satisfied), the loss becomes exactly 0 and the gradient is 0. The model stops learning from this example.
- **Sharp transition**. The hinge loss has a sharp corner at the margin boundary, it switches abruptly from penalizing to not penalizing.

*The log loss behaves similarly but "softly"*: The log loss is called a "soft-hinge" loss because it behaves similarly to the hinge loss but with a smooth, continuous gradient rather than a hard cutoff:

$$l_{log} = \log(1 + e^{F_W(x,y) - F_W(x,\overline{y})})$$

- **Soft saturation**. When $F_W(x,y) - F_W(x,\overline{y})$ becomes very negative (large margin), the loss approaches 0 asymptotically: $\log(1 + e^{F_W(x,y) - F_W(x,\overline{y})}) \approx 0$ but never exactly reaches 0.
- **Smooth gradient**. The gradient is:

$$\frac{\partial l_{log}}{\partial W} = \sigma(F_W(x,y) - F_W(x,\overline{y})) \cdot \left( \frac{\partial F_W(x,y)}{\partial W} - \frac{\partial F_W(x,\overline{y})}{\partial W} \right)$$

  where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function. As the margin increases (becomes more negative), $\sigma(z) \to 0$ smoothly, causing the gradient to decay continuously rather than dropping to 0 abruptly.
- **Continuous everywhere**. The log loss has no sharp corners, it's smooth and differentiable everywhere.

***Advantages of using soft hinge loss***

1. **Smooth optimization:** The continuous, smooth gradient makes optimization easier and more stable. There are no discontinuities in the gradient that could cause issues with gradient-based optimizers.
2. **Robustness:** The soft margin is less sensitive to outliers or mislabeled data. A single example that violates the hard margin won't dominate the gradient as much as it would with hinge loss.
3. **Better convergence properties:** Smooth loss surfaces generally lead to better convergence behavior in gradient descent, avoiding issues like getting stuck at non-smooth points.
4. **Probabilistic interpretation:** The sigmoid $\sigma(F_W(x,\overline{y}) - F_W(x,y))$ can be interpreted as the probability that $y$ is preferred over $\overline{y}$, providing a probabilistic framework for ranking examples.

## Problem iii

Solution

***Key differences***

1. **Absolute vs. relative energy targets**
   - **Simple and square-square losses** penalize the *absolute energy values*:
     - They push $F_W(x, y)$ toward 0 or below (via $[F_W(x, y)]^+$)
     - They push $F_W(x, \overline{y})$ toward $m$ or above (via $[m - F_W(x, \overline{y})]^+$)
     - Each term has an *independent target*: correct examples should have energy $\leq 0$, incorrect examples should have energy $\geq m$
   - **Hinge and log losses** penalize the *relative energy difference*:
     - They only care about the gap $F_W(x, y) - F_W(x, \overline{y})$
     - They have *no absolute target* for individual energies
2. **Independent vs. coupled gradient terms**
   - **Simple and square-square losses** have *two independent gradient terms* - one for pushing down $F_W(x, y)$ and one for pushing up $F_W(x, \overline{y})$. These terms activate independently based on absolute energy values.
   - **Hinge and log losses** have *coupled gradients* - both $F_W(x, y)$ and $F_W(x, \overline{y})$ are always updated together based on their difference. You cannot push one without affecting the gradient contribution from the other.
3. **Energy scale sensitivity**
   - Simple and square-square losses are sensitive to the absolute scale of energies
   - Hinge and log losses are invariant to adding a constant to all energies: $F_W(x, y) + c$ and $F_W(x, \overline{y}) + c$ give the same loss

***When to use simple loss***

- When you want to *explicitly control absolute energy values*, ensuring correct examples have low (near-zero or negative) energy and incorrect examples have high energy above a threshold
- When you want *uniform gradient strength* regardless of violation magnitude
- When the *energy scale matters* (e.g., when energy represents a meaningful quantity like cost)
- For *hard constraints* on energy ranges and *simpler optimization* with constant gradients

***When to use square-square loss***

- When you want to *penalize large violations more heavily*: the quadratic penalty gives stronger gradients for examples that are far from their targets
- When you want *smoother energy surfaces*: the quadratic form produces smoother gradients compared to the piecewise linear simple loss
- For tasks requiring *fine-tuning near boundaries*: the gradient magnitude decreases as energies approach their targets, allowing more careful refinement
- For *faster initial corrections* for badly misclassified examples due to stronger gradients