Variational Inference: Spam Detection

In this assignment, we will load the UCI SMS Spam Collection dataset. However, instead of using it directly, we will use fixed-size vector embeddings of the message. Those embeddings have already been produced and are provided here to you.

Setting the Scene

The goal of this assignment to go beyond a traditional classifier and apply variational methods. This means that we will train a model using some parameters over the distribution of which we have a prior belief. Our prior belief here is actually that each and every parameter independently follows a standard normal distribution.

During optimization, we will draw sets of parameters from our variational distribution q_ϕ. Drawing these parameters needs to be done using the reparameterization trick, so that we can add noise, i.e., $w_i = \mu_i + \sigma_i \cdot \epsilon$. The assignment also poses one or the other question (clearly marked), you need to provide your answer directly after.

The assignment has some blank spots for you to fill out (but you can customize your implementation to your liking). Training should be done using stochastic gradient descend, either using manual gradient updates or using an optimizer.

You should use autodiff-capabilities to compute gradients. It is recommended to use, for example, JAX or PyTorch. We recommend the latter, using it in a functional way (i.e., using `torch.func.jacrev`). The blanks left in this assignment and their type hints assume PyTorch. There are some cells with quick tests/sanity-checks, that you are free to remove, especially if they do not go along with how you chose to implement your solution.

At the end of the assignment, after training, you need to pick one advanced method of evaluation. We are not interested in traditional metrics here (e.g., accuracy, Kappa, F1, etc.; although you are welcome to show those). Rather, we want to exploit the variational nature of the model here and show something more interesting.

Load the Data

You're variational model shall use no more than 15 components (aim for ~5 or fewer). You'll have to apply a dimensionality reduction.

Sentence Embeddings were created in two ways:

1. (Recommended) Using ALBERT XLarge v2 (`albert-xlarge-v2`). Dim $= 2,048$.
2. Using English word vectors from `wiki-news-300d-1M` using `fasttext` . Dim $= 300$.

In either case, the embeddings were averaged along the sequence dimension to produce fixed-size vectors. In the provided dataset, the messages are retained. This is useful should you choose a qualitative evaluation.

The labels have already been converted to floats: 0.0=ham, 1.0=spam. The default example below shows a spam message.

In [1]:
```
# Modify only the file you wish to load.
# Each data file contains 3 keys: X, y, msg
import numpy as np

data = np.load(file='2048d_sms_spam_albert-xlarge-v2.npz')
# data = np.load(file='300d_sms_spam_fasttext_pca.npz')

X: np.ndarray; Y: np.ndarray
X, Y, msg = data.get('X'), data.get('y'), data.get('msg')
X.shape, Y.shape, len(msg), msg[2], Y[2]
```

Out[1]:
```
((5572, 2048),
 (5572,),
 5572,
 np.str_("Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. T
ext FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075o
ver18's"),
 np.int64(1))
```

Define the Model

Our model is to be a polynomial with degree corresponding to the number of components chosen for the PCA.

$$z = w_0 + w_1 \cdot x_1 + w_2 x_2^2 + \cdots + w_n x_n^n. \tag{1}$$

We will perform a **binary** classification problem, using the binary cross-entropy (CE). CE has a range of $[0, \infty)$. The better the predictions of our model, the lower the CE.

Note that binary CE requires our predictions to be in the range $[0, 1]$. Therefore, we will have to pass its raw outputs ("logits") through the Sigmoid function. This makes our model a **logistic** classifier. The Sigmoid function is defined as $s(x) = \frac{1}{1+e^{-z}}$.

$$p_i = s(z_i), \text{ convert our raw predictions to probabilities, then:} \tag{2}$$

$$\hat{y}_i \sim \text{Bernoulli}(p_i). \tag{3}$$

Since our predicted $\hat{y}$ will follow a Bernoulli distribution, we can directy use its likelihood function. Note that maximizing the Bernoulli likelihood is equivalent of minimizing the binary CE! The Bernoulli distribution is parameterized by a single parameter, p.

However, in our context, p is unknown. In order not to confuse the Bernoulli distribution's parameter p with anything, in the following, we have substituted it with $s(z_i)$. We will optimize for it, so that the output of our logisitic classifier becomes p. The likelihood (for the prediction $\hat{y}_i$ of a single observation and its label y_i) then becomes:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_i s(z_i)^{y_i} \cdot (1 - s(z_i))^{1-y_i}, \tag{4}$$

$$\log\left(p(\mathbf{y}|\mathbf{X}, \mathbf{w})\right) = \sum_i \left[\log\left(s(z_i)^{y_i}\right) + \log\left((1 - s(z_i))^{1-y_i}\right)\right], \tag{5}$$

$$= \sum_i \left[y_i \cdot \log\left(s(z_i)\right) + (1 - y_i) \cdot \log\left(1 - s(z_i)\right)\right]. \tag{6}$$

- Our **prior** belief is that each parameter follows a standard normal distribution, i.e., $w_j \sim \mathcal{N}(0, 1)$.
- For our variational distribution, use a mean-field approximation of standard (independent) normals, too.

Question: Conceptually, how does the log-likelihood $p(y|x, w)$ compute its result given a single n-dimensional observation under the assumption of independent dimensions?

Answer: In case of independent dimensions, the (log-)likelihood across all dimensions is multiplied (summed).

Implement the Model

Here, you are encouraged to use a library/framework like PyTorch or JAX and esp. functionality for automatic differentiation to compute gradients. You may also import functions like `sigmoid` or `vmap` for vectorized operations.

It is recommended to implement vectorized versions of your required functions (i.e., batch-processing).

It is preferable to use type-hints for your functions. Furthermore, you can write better code by inserting assertions (e.g., for dimensionality or other sanity-checks). Please use Python-style comments like in the following:

```
def func(w0: float, x: float) -> float:
    """
    Function to compute a multiple of x.
    """
    ...
```

In [2]:
```
!pip install torch torchvision torchaudio


import torch
from torch import Tensor

print(torch.__version__) # Just to check the version and that torch is loaded
print("CUDA available:", torch.cuda.is_available())


def model(w: Tensor, x: Tensor) -> Tensor:
    """
    Takes the coefficients w, the observations, computes the polynomial and
    returns probabilities.
    w: Tensor of shape (..., D+1) containing [w0, w1, ..., wD]
```

```
    x: Tensor of shape (N, D) containing D-dimensional inputs for N samples

    Returns:
        Tensor of shape (N,) with predicted probabilities in [0,1]
    """
    # Number of features
    D = x.shape[1]
    # Ensure w has D+1 coefficients
    assert w.shape[-1] == D + 1, f"Expected w[..., D+1], got {w.shape[-1]}"

    # Compute the polynomial z = w0 + w1*x1 + w2*(x2^2) + ... + wD*(xD^D)
    # Broadcast w terms against x dimensions
    z = w[..., 0]  # bias term, shape broadcastable to (N,)
    for j in range(1, D + 1):
        # x[:, j-1] raised to the j-th power, shape (N,)
        term = x[:, j-1] ** j
        # w[..., j] has shape like (...,)
        z = z + w[..., j] * term
    # Apply sigmoid to get probabilities
    return torch.sigmoid(z)
```

```
[notice] A new release of pip is available: 23.0.1 -> 25.1.1
[notice] To update, run: C:\Users\kemal\AppData\Local\Microsoft\WindowsApps\Pytho
nSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\python.exe -m pip install --upgrade
pip
```

```
Collecting torch
  Downloading torch-2.7.0-cp310-cp310-win_amd64.whl (212.5 MB)
     ---------------------------------------- 0.0/212.5 MB ? eta -:--:--
     ---------------------------------------- 0.2/212.5 MB 6.9 MB/s eta 0:00:31
     ---------------------------------------- 1.9/212.5 MB 24.4 MB/s eta 0:00:09
      ---------------------------------------- 2.7/212.5 MB 21.9 MB/s eta 0:00:10
      ---------------------------------------- 3.4/212.5 MB 19.6 MB/s eta 0:00:11
      ---------------------------------------- 4.0/212.5 MB 18.1 MB/s eta 0:00:12
      ---------------------------------------- 4.5/212.5 MB 16.8 MB/s eta 0:00:13
      ---------------------------------------- 4.5/212.5 MB 16.8 MB/s eta 0:00:13
      ---------------------------------------- 4.5/212.5 MB 16.8 MB/s eta 0:00:13
      ---------------------------------------- 5.1/212.5 MB 12.4 MB/s eta 0:00:17
      ---------------------------------------- 5.1/212.5 MB 12.4 MB/s eta 0:00:17
     - -------------------------------------- 7.4/212.5 MB 14.8 MB/s eta 0:00:14
     - -------------------------------------- 7.8/212.5 MB 14.7 MB/s eta 0:00:14
     - -------------------------------------- 7.8/212.5 MB 14.7 MB/s eta 0:00:14
     - -------------------------------------- 7.8/212.5 MB 14.7 MB/s eta 0:00:14
     - -------------------------------------- 9.7/212.5 MB 14.1 MB/s eta 0:00:15
     - -------------------------------------- 10.1/212.5 MB 14.0 MB/s eta 0:00:15
     - -------------------------------------- 10.1/212.5 MB 14.0 MB/s eta 0:00:15
     -- ------------------------------------- 11.6/212.5 MB 13.6 MB/s eta 0:00:15
     -- ------------------------------------- 12.2/212.5 MB 13.1 MB/s eta 0:00:16
     -- ------------------------------------- 12.8/212.5 MB 12.9 MB/s eta 0:00:16
     -- ------------------------------------- 13.5/212.5 MB 12.8 MB/s eta 0:00:16
     -- ------------------------------------- 14.1/212.5 MB 12.8 MB/s eta 0:00:16
     -- ------------------------------------- 14.8/212.5 MB 14.6 MB/s eta 0:00:14
     -- ------------------------------------- 15.4/212.5 MB 16.0 MB/s eta 0:00:13
     -- ------------------------------------- 16.0/212.5 MB 14.9 MB/s eta 0:00:14
     -- ------------------------------------- 16.6/212.5 MB 14.2 MB/s eta 0:00:14
     --- ------------------------------------ 17.3/212.5 MB 13.4 MB/s eta 0:00:15
     --- ------------------------------------ 17.9/212.5 MB 13.1 MB/s eta 0:00:15
     --- ------------------------------------ 18.5/212.5 MB 14.6 MB/s eta 0:00:14
     --- ------------------------------------ 19.1/212.5 MB 13.6 MB/s eta 0:00:15
     --- ------------------------------------ 19.8/212.5 MB 13.4 MB/s eta 0:00:15
     --- ------------------------------------ 20.3/212.5 MB 14.5 MB/s eta 0:00:14
     --- ------------------------------------ 21.0/212.5 MB 13.9 MB/s eta 0:00:14
     --- ------------------------------------ 21.6/212.5 MB 13.1 MB/s eta 0:00:15
     --- ------------------------------------ 22.2/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 22.8/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 23.4/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 23.5/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 23.5/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 24.4/212.5 MB 12.4 MB/s eta 0:00:16
     ---- ----------------------------------- 25.6/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 25.6/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 27.0/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 27.5/212.5 MB 13.1 MB/s eta 0:00:15
     ---- ----------------------------------- 27.8/212.5 MB 12.8 MB/s eta 0:00:15
     ----- ---------------------------------- 28.9/212.5 MB 13.1 MB/s eta 0:00:15
     ----- ---------------------------------- 29.1/212.5 MB 12.6 MB/s eta 0:00:15
     ----- ---------------------------------- 29.7/212.5 MB 12.6 MB/s eta 0:00:15
     ----- ---------------------------------- 30.4/212.5 MB 12.6 MB/s eta 0:00:15
     ----- ---------------------------------- 31.0/212.5 MB 12.6 MB/s eta 0:00:15
     ----- ---------------------------------- 31.4/212.5 MB 12.4 MB/s eta 0:00:15
     ----- ---------------------------------- 31.9/212.5 MB 12.1 MB/s eta 0:00:15
     ----- ---------------------------------- 32.3/212.5 MB 12.1 MB/s eta 0:00:15
     ----- ---------------------------------- 32.8/212.5 MB 11.9 MB/s eta 0:00:16
     ----- ---------------------------------- 33.3/212.5 MB 11.7 MB/s eta 0:00:16
     ------ --------------------------------- 33.8/212.5 MB 13.1 MB/s eta 0:00:14
     ------ --------------------------------- 34.3/212.5 MB 12.6 MB/s eta 0:00:15
```

```
------ ------------------------------ 34.8/212.5 MB 12.1 MB/s eta 0:00:15
------ ------------------------------ 35.3/212.5 MB 11.7 MB/s eta 0:00:16
------ ------------------------------ 35.8/212.5 MB 11.3 MB/s eta 0:00:16
------ ------------------------------ 36.4/212.5 MB 11.7 MB/s eta 0:00:16
------ ------------------------------ 36.9/212.5 MB 11.5 MB/s eta 0:00:16
------ ------------------------------ 37.4/212.5 MB 11.1 MB/s eta 0:00:16
------ ------------------------------ 38.0/212.5 MB 11.3 MB/s eta 0:00:16
------ ------------------------------ 38.5/212.5 MB 10.9 MB/s eta 0:00:16
------ ------------------------------ 39.1/212.5 MB 10.9 MB/s eta 0:00:16
------- ------------------------------ 39.7/212.5 MB 11.1 MB/s eta 0:00:16
------- ------------------------------ 40.2/212.5 MB 10.9 MB/s eta 0:00:16
------- ------------------------------ 40.8/212.5 MB 10.9 MB/s eta 0:00:16
------- ------------------------------ 41.4/212.5 MB 11.1 MB/s eta 0:00:16
------- ------------------------------ 42.1/212.5 MB 11.5 MB/s eta 0:00:15
------- ------------------------------ 42.7/212.5 MB 11.5 MB/s eta 0:00:15
------- ------------------------------ 43.3/212.5 MB 11.9 MB/s eta 0:00:15
------- ------------------------------ 43.8/212.5 MB 12.1 MB/s eta 0:00:14
------- ------------------------------ 44.5/212.5 MB 12.3 MB/s eta 0:00:14
-------- ------------------------------ 45.1/212.5 MB 12.4 MB/s eta 0:00:14
-------- ------------------------------ 45.8/212.5 MB 12.6 MB/s eta 0:00:14
-------- ------------------------------ 46.5/212.5 MB 12.8 MB/s eta 0:00:13
-------- ------------------------------ 47.1/212.5 MB 13.1 MB/s eta 0:00:13
-------- ------------------------------ 47.7/212.5 MB 13.1 MB/s eta 0:00:13
-------- ------------------------------ 48.3/212.5 MB 13.1 MB/s eta 0:00:13
-------- ------------------------------ 48.8/212.5 MB 13.1 MB/s eta 0:00:13
-------- ------------------------------ 49.2/212.5 MB 12.8 MB/s eta 0:00:13
-------- ------------------------------ 49.7/212.5 MB 12.9 MB/s eta 0:00:13
-------- ------------------------------ 50.2/212.5 MB 12.6 MB/s eta 0:00:13
--------- ------------------------------ 50.7/212.5 MB 12.6 MB/s eta 0:00:13
--------- ------------------------------ 51.2/212.5 MB 12.4 MB/s eta 0:00:14
--------- ------------------------------ 51.7/212.5 MB 12.4 MB/s eta 0:00:14
--------- ------------------------------ 52.2/212.5 MB 12.1 MB/s eta 0:00:14
--------- ------------------------------ 52.7/212.5 MB 12.1 MB/s eta 0:00:14
--------- ------------------------------ 53.3/212.5 MB 11.9 MB/s eta 0:00:14
--------- ------------------------------ 53.9/212.5 MB 11.7 MB/s eta 0:00:14
--------- ------------------------------ 54.2/212.5 MB 11.5 MB/s eta 0:00:14
--------- ------------------------------ 54.8/212.5 MB 11.3 MB/s eta 0:00:14
--------- ------------------------------ 55.7/212.5 MB 11.3 MB/s eta 0:00:14
---------- ------------------------------ 56.2/212.5 MB 11.1 MB/s eta 0:00:15
---------- ------------------------------ 56.8/212.5 MB 10.9 MB/s eta 0:00:15
---------- ------------------------------ 57.4/212.5 MB 10.9 MB/s eta 0:00:15
---------- ------------------------------ 58.0/212.5 MB 11.1 MB/s eta 0:00:14
---------- ------------------------------ 58.6/212.5 MB 11.1 MB/s eta 0:00:14
---------- ------------------------------ 59.2/212.5 MB 11.3 MB/s eta 0:00:14
---------- ------------------------------ 59.8/212.5 MB 11.3 MB/s eta 0:00:14
---------- ------------------------------ 60.4/212.5 MB 11.5 MB/s eta 0:00:14
---------- ------------------------------ 61.0/212.5 MB 11.7 MB/s eta 0:00:13
----------- ------------------------------ 61.7/212.5 MB 11.7 MB/s eta 0:00:13
----------- ------------------------------ 62.6/212.5 MB 12.1 MB/s eta 0:00:13
----------- ------------------------------ 63.2/212.5 MB 12.4 MB/s eta 0:00:13
----------- ------------------------------ 63.8/212.5 MB 12.4 MB/s eta 0:00:13
----------- ------------------------------ 64.4/212.5 MB 12.8 MB/s eta 0:00:12
----------- ------------------------------ 65.1/212.5 MB 13.4 MB/s eta 0:00:12
----------- ------------------------------ 65.8/212.5 MB 13.4 MB/s eta 0:00:11
----------- ------------------------------ 66.5/212.5 MB 13.6 MB/s eta 0:00:11
------------ ------------------------------ 67.1/212.5 MB 13.6 MB/s eta 0:00:11
------------ ------------------------------ 68.1/212.5 MB 13.9 MB/s eta 0:00:11
------------ ------------------------------ 68.8/212.5 MB 13.9 MB/s eta 0:00:11
------------ ------------------------------ 69.4/212.5 MB 14.2 MB/s eta 0:00:11
------------ ------------------------------ 70.2/212.5 MB 14.2 MB/s eta 0:00:10
```

```
------------ ------------------------- 71.0/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 71.7/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 72.3/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 72.9/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 73.5/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 74.1/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 74.7/212.5 MB 14.2 MB/s eta 0:00:10
------------ ------------------------- 75.3/212.5 MB 13.9 MB/s eta 0:00:10
------------ ------------------------- 76.0/212.5 MB 13.9 MB/s eta 0:00:10
------------ ------------------------- 76.6/212.5 MB 13.9 MB/s eta 0:00:10
------------ ------------------------- 77.1/212.5 MB 13.6 MB/s eta 0:00:10
------------ ------------------------- 77.9/212.5 MB 13.4 MB/s eta 0:00:11
------------- ------------------------ 78.5/212.5 MB 13.4 MB/s eta 0:00:11
------------- ------------------------ 79.3/212.5 MB 13.4 MB/s eta 0:00:10
------------- ------------------------ 80.0/212.5 MB 13.1 MB/s eta 0:00:11
------------- ------------------------ 80.5/212.5 MB 13.1 MB/s eta 0:00:11
------------- ------------------------ 81.1/212.5 MB 13.1 MB/s eta 0:00:11
------------- ------------------------ 81.7/212.5 MB 12.8 MB/s eta 0:00:11
------------- ------------------------ 82.1/212.5 MB 12.6 MB/s eta 0:00:11
------------- ------------------------ 82.5/212.5 MB 12.4 MB/s eta 0:00:11
------------- ------------------------ 82.9/212.5 MB 12.1 MB/s eta 0:00:11
------------- ------------------------ 83.3/212.5 MB 11.9 MB/s eta 0:00:11
------------- ------------------------ 83.7/212.5 MB 11.7 MB/s eta 0:00:12
-------------- ----------------------- 84.2/212.5 MB 11.7 MB/s eta 0:00:11
-------------- ----------------------- 84.7/212.5 MB 11.5 MB/s eta 0:00:12
-------------- ----------------------- 85.1/212.5 MB 11.3 MB/s eta 0:00:12
-------------- ----------------------- 85.6/212.5 MB 11.3 MB/s eta 0:00:12
-------------- ----------------------- 86.1/212.5 MB 10.9 MB/s eta 0:00:12
-------------- ----------------------- 86.5/212.5 MB 10.9 MB/s eta 0:00:12
-------------- ----------------------- 87.0/212.5 MB 10.9 MB/s eta 0:00:12
-------------- ----------------------- 87.5/212.5 MB 10.7 MB/s eta 0:00:12
-------------- ----------------------- 88.0/212.5 MB 10.7 MB/s eta 0:00:12
-------------- ----------------------- 88.5/212.5 MB 10.6 MB/s eta 0:00:12
-------------- ----------------------- 89.0/212.5 MB 10.6 MB/s eta 0:00:12
--------------- ---------------------- 89.6/212.5 MB 10.4 MB/s eta 0:00:12
--------------- ---------------------- 90.1/212.5 MB 10.4 MB/s eta 0:00:12
--------------- ---------------------- 90.6/212.5 MB 10.4 MB/s eta 0:00:12
--------------- ---------------------- 91.2/212.5 MB 10.2 MB/s eta 0:00:12
--------------- ---------------------- 91.7/212.5 MB 10.2 MB/s eta 0:00:12
--------------- ---------------------- 92.3/212.5 MB 10.4 MB/s eta 0:00:12
--------------- ---------------------- 92.9/212.5 MB 10.7 MB/s eta 0:00:12
--------------- ---------------------- 93.4/212.5 MB 10.7 MB/s eta 0:00:12
--------------- ---------------------- 94.0/212.5 MB 11.1 MB/s eta 0:00:11
--------------- ---------------------- 94.6/212.5 MB 11.1 MB/s eta 0:00:11
---------------- --------------------- 95.2/212.5 MB 11.3 MB/s eta 0:00:11
---------------- --------------------- 95.8/212.5 MB 11.5 MB/s eta 0:00:11
---------------- --------------------- 96.2/212.5 MB 11.5 MB/s eta 0:00:11
---------------- --------------------- 96.6/212.5 MB 11.5 MB/s eta 0:00:11
---------------- --------------------- 97.1/212.5 MB 11.3 MB/s eta 0:00:11
---------------- --------------------- 97.5/212.5 MB 11.3 MB/s eta 0:00:11
---------------- --------------------- 98.0/212.5 MB 11.3 MB/s eta 0:00:11
---------------- --------------------- 98.4/212.5 MB 11.1 MB/s eta 0:00:11
---------------- --------------------- 98.9/212.5 MB 11.1 MB/s eta 0:00:11
---------------- --------------------- 99.4/212.5 MB 11.1 MB/s eta 0:00:11
---------------- --------------------- 99.8/212.5 MB 10.9 MB/s eta 0:00:11
---------------- -------------------- 100.4/212.5 MB 10.9 MB/s eta 0:00:11
---------------- -------------------- 100.8/212.5 MB 10.9 MB/s eta 0:00:11
---------------- -------------------- 101.3/212.5 MB 10.9 MB/s eta 0:00:11
---------------- -------------------- 101.8/212.5 MB 10.9 MB/s eta 0:00:11
---------------- -------------------- 102.3/212.5 MB 10.7 MB/s eta 0:00:11
```

```
---------------- ------------------ 102.9/212.5 MB 10.9 MB/s eta 0:00:11
---------------- ------------------ 103.4/212.5 MB 10.7 MB/s eta 0:00:11
---------------- ------------------ 103.9/212.5 MB 10.7 MB/s eta 0:00:11
---------------- ------------------ 104.4/212.5 MB 10.7 MB/s eta 0:00:11
---------------- ------------------ 105.0/212.5 MB 10.7 MB/s eta 0:00:11
---------------- ------------------ 105.6/212.5 MB 10.7 MB/s eta 0:00:10
---------------- ------------------ 106.1/212.5 MB 10.7 MB/s eta 0:00:10
---------------- ------------------ 106.7/212.5 MB 10.9 MB/s eta 0:00:10
---------------- ------------------ 107.3/212.5 MB 11.1 MB/s eta 0:00:10
---------------- ------------------ 107.8/212.5 MB 11.1 MB/s eta 0:00:10
---------------- ------------------ 108.5/212.5 MB 11.3 MB/s eta 0:00:10
---------------- ------------------ 109.0/212.5 MB 11.5 MB/s eta 0:00:10
---------------- ------------------ 109.4/212.5 MB 11.3 MB/s eta 0:00:10
---------------- ------------------ 109.9/212.5 MB 11.3 MB/s eta 0:00:10
---------------- ------------------ 110.3/212.5 MB 11.3 MB/s eta 0:00:10
---------------- ------------------ 110.7/212.5 MB 11.3 MB/s eta 0:00:10
---------------- ------------------ 111.2/212.5 MB 11.1 MB/s eta 0:00:10
---------------- ------------------ 111.6/212.5 MB 11.1 MB/s eta 0:00:10
---------------- ------------------ 112.1/212.5 MB 11.1 MB/s eta 0:00:10
---------------- ------------------ 112.6/212.5 MB 10.9 MB/s eta 0:00:10
---------------- ------------------ 113.0/212.5 MB 10.9 MB/s eta 0:00:10
---------------- ------------------ 113.5/212.5 MB 10.9 MB/s eta 0:00:10
---------------- ------------------ 114.0/212.5 MB 10.9 MB/s eta 0:00:10
---------------- ------------------ 114.4/212.5 MB 10.7 MB/s eta 0:00:10
---------------- ------------------ 114.7/212.5 MB 10.4 MB/s eta 0:00:10
---------------- ------------------ 115.1/212.5 MB 10.2 MB/s eta 0:00:10
---------------- ------------------ 115.4/212.5 MB 10.1 MB/s eta 0:00:10
---------------- ------------------ 115.8/212.5 MB 9.9 MB/s eta 0:00:10
---------------- ------------------ 116.2/212.5 MB 9.8 MB/s eta 0:00:10
---------------- ------------------ 116.6/212.5 MB 9.6 MB/s eta 0:00:10
---------------- ------------------ 117.0/212.5 MB 9.6 MB/s eta 0:00:10
---------------- ------------------ 117.4/212.5 MB 9.5 MB/s eta 0:00:11
---------------- ------------------ 117.8/212.5 MB 9.4 MB/s eta 0:00:11
---------------- ------------------ 118.2/212.5 MB 9.2 MB/s eta 0:00:11
---------------- ------------------ 118.6/212.5 MB 9.1 MB/s eta 0:00:11
---------------- ------------------ 119.0/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 119.4/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 119.9/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 120.3/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 120.8/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 121.2/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 121.7/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 122.2/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 122.7/212.5 MB 9.0 MB/s eta 0:00:11
---------------- ------------------ 123.2/212.5 MB 9.0 MB/s eta 0:00:10
---------------- ------------------ 123.7/212.5 MB 9.1 MB/s eta 0:00:10
---------------- ------------------ 124.1/212.5 MB 9.1 MB/s eta 0:00:10
---------------- ------------------ 124.7/212.5 MB 9.2 MB/s eta 0:00:10
---------------- ------------------ 125.2/212.5 MB 9.5 MB/s eta 0:00:10
---------------- ------------------ 125.7/212.5 MB 9.8 MB/s eta 0:00:09
---------------- ------------------ 126.3/212.5 MB 9.9 MB/s eta 0:00:09
---------------- ------------------ 126.8/212.5 MB 10.1 MB/s eta 0:00:09
---------------- ------------------ 127.3/212.5 MB 10.2 MB/s eta 0:00:09
---------------- ------------------ 127.9/212.5 MB 10.4 MB/s eta 0:00:09
---------------- ------------------ 128.4/212.5 MB 10.6 MB/s eta 0:00:08
---------------- ------------------ 129.0/212.5 MB 10.7 MB/s eta 0:00:08
---------------- ------------------ 129.6/212.5 MB 10.9 MB/s eta 0:00:08
---------------- ------------------ 130.1/212.5 MB 11.1 MB/s eta 0:00:08
---------------- ------------------ 130.7/212.5 MB 11.3 MB/s eta 0:00:08
---------------- ------------------ 131.3/212.5 MB 11.5 MB/s eta 0:00:08
```

```
---------------------- -------------- 131.9/212.5 MB 11.7 MB/s eta 0:00:07
---------------------- -------------- 132.6/212.5 MB 11.7 MB/s eta 0:00:07
---------------------- -------------- 133.2/212.5 MB 11.9 MB/s eta 0:00:07
---------------------- -------------- 133.8/212.5 MB 12.1 MB/s eta 0:00:07
---------------------- -------------- 134.4/212.5 MB 12.4 MB/s eta 0:00:07
---------------------- -------------- 135.1/212.5 MB 12.4 MB/s eta 0:00:07
---------------------- -------------- 135.7/212.5 MB 12.6 MB/s eta 0:00:07
---------------------- -------------- 136.4/212.5 MB 12.8 MB/s eta 0:00:06
---------------------- -------------- 137.0/212.5 MB 13.1 MB/s eta 0:00:06
---------------------- -------------- 137.7/212.5 MB 13.1 MB/s eta 0:00:06
----------------------- ------------- 138.4/212.5 MB 13.4 MB/s eta 0:00:06
----------------------- ------------- 138.9/212.5 MB 13.6 MB/s eta 0:00:06
----------------------- ------------- 139.5/212.5 MB 13.4 MB/s eta 0:00:06
----------------------- ------------- 140.0/212.5 MB 13.1 MB/s eta 0:00:06
----------------------- ------------- 140.5/212.5 MB 13.1 MB/s eta 0:00:06
----------------------- ------------- 141.0/212.5 MB 12.8 MB/s eta 0:00:06
----------------------- ------------- 141.5/212.5 MB 12.8 MB/s eta 0:00:06
----------------------- ------------- 142.0/212.5 MB 12.6 MB/s eta 0:00:06
----------------------- ------------- 142.5/212.5 MB 12.6 MB/s eta 0:00:06
----------------------- ------------- 143.1/212.5 MB 12.3 MB/s eta 0:00:06
------------------------ ------------ 143.6/212.5 MB 12.4 MB/s eta 0:00:06
------------------------ ------------ 144.1/212.5 MB 12.4 MB/s eta 0:00:06
------------------------ ------------ 144.7/212.5 MB 12.1 MB/s eta 0:00:06
------------------------ ------------ 145.2/212.5 MB 12.1 MB/s eta 0:00:06
------------------------ ------------ 145.8/212.5 MB 11.9 MB/s eta 0:00:06
------------------------ ------------ 146.3/212.5 MB 11.9 MB/s eta 0:00:06
------------------------ ------------ 146.9/212.5 MB 11.7 MB/s eta 0:00:06
------------------------ ------------ 147.6/212.5 MB 11.9 MB/s eta 0:00:06
------------------------ ------------ 147.9/212.5 MB 11.5 MB/s eta 0:00:06
------------------------ ------------ 148.5/212.5 MB 11.5 MB/s eta 0:00:06
------------------------ ------------ 149.1/212.5 MB 11.3 MB/s eta 0:00:06
------------------------- ----------- 149.7/212.5 MB 11.5 MB/s eta 0:00:06
------------------------- ----------- 150.3/212.5 MB 11.5 MB/s eta 0:00:06
------------------------- ----------- 150.9/212.5 MB 11.7 MB/s eta 0:00:06
------------------------- ----------- 151.6/212.5 MB 11.9 MB/s eta 0:00:06
------------------------- ----------- 152.2/212.5 MB 12.1 MB/s eta 0:00:05
------------------------- ----------- 152.9/212.5 MB 12.1 MB/s eta 0:00:05
------------------------- ----------- 153.5/212.5 MB 12.6 MB/s eta 0:00:05
------------------------- ----------- 154.2/212.5 MB 12.8 MB/s eta 0:00:05
------------------------- ----------- 154.8/212.5 MB 12.9 MB/s eta 0:00:05
-------------------------- ---------- 155.3/212.5 MB 12.8 MB/s eta 0:00:05
-------------------------- ---------- 155.8/212.5 MB 12.8 MB/s eta 0:00:05
-------------------------- ---------- 156.3/212.5 MB 12.6 MB/s eta 0:00:05
-------------------------- ---------- 156.8/212.5 MB 12.6 MB/s eta 0:00:05
-------------------------- ---------- 157.3/212.5 MB 12.6 MB/s eta 0:00:05
-------------------------- ---------- 157.8/212.5 MB 12.4 MB/s eta 0:00:05
-------------------------- ---------- 158.3/212.5 MB 12.6 MB/s eta 0:00:05
-------------------------- ---------- 158.8/212.5 MB 12.4 MB/s eta 0:00:05
-------------------------- ---------- 159.3/212.5 MB 12.4 MB/s eta 0:00:05
-------------------------- ---------- 159.9/212.5 MB 12.1 MB/s eta 0:00:05
-------------------------- ---------- 160.3/212.5 MB 11.9 MB/s eta 0:00:05
--------------------------- --------- 160.8/212.5 MB 11.9 MB/s eta 0:00:05
--------------------------- --------- 161.4/212.5 MB 11.9 MB/s eta 0:00:05
--------------------------- --------- 161.9/212.5 MB 11.7 MB/s eta 0:00:05
--------------------------- --------- 162.5/212.5 MB 11.7 MB/s eta 0:00:05
--------------------------- --------- 163.1/212.5 MB 11.7 MB/s eta 0:00:05
--------------------------- --------- 163.6/212.5 MB 11.5 MB/s eta 0:00:05
--------------------------- --------- 164.2/212.5 MB 11.5 MB/s eta 0:00:05
--------------------------- --------- 164.8/212.5 MB 11.3 MB/s eta 0:00:05
--------------------------- --------- 165.4/212.5 MB 11.5 MB/s eta 0:00:05
```

```
---------------------------- -------- 166.1/212.5 MB 11.7 MB/s eta 0:00:04
---------------------------- ------- 166.7/212.5 MB 11.7 MB/s eta 0:00:04
---------------------------- ------- 167.3/212.5 MB 11.9 MB/s eta 0:00:04
---------------------------- ------- 167.9/212.5 MB 12.1 MB/s eta 0:00:04
---------------------------- ------- 168.5/212.5 MB 12.4 MB/s eta 0:00:04
---------------------------- ------- 169.2/212.5 MB 12.4 MB/s eta 0:00:04
---------------------------- ------- 169.9/212.5 MB 12.6 MB/s eta 0:00:04
---------------------------- ------- 170.5/212.5 MB 12.8 MB/s eta 0:00:04
---------------------------- ------- 171.2/212.5 MB 13.1 MB/s eta 0:00:04
---------------------------- ------- 171.7/212.5 MB 13.1 MB/s eta 0:00:04
----------------------------- ------ 172.4/212.5 MB 13.1 MB/s eta 0:00:04
----------------------------- ------ 172.8/212.5 MB 13.1 MB/s eta 0:00:04
----------------------------- ------ 173.3/212.5 MB 12.8 MB/s eta 0:00:04
----------------------------- ------ 173.8/212.5 MB 12.8 MB/s eta 0:00:04
----------------------------- ------ 174.3/212.5 MB 12.6 MB/s eta 0:00:04
----------------------------- ------ 174.8/212.5 MB 12.6 MB/s eta 0:00:03
----------------------------- ------ 175.3/212.5 MB 12.4 MB/s eta 0:00:04
----------------------------- ------ 175.9/212.5 MB 12.4 MB/s eta 0:00:03
----------------------------- ------ 176.4/212.5 MB 12.1 MB/s eta 0:00:03
----------------------------- ------ 176.9/212.5 MB 12.1 MB/s eta 0:00:03
----------------------------- ------ 177.4/212.5 MB 11.9 MB/s eta 0:00:03
----------------------------- ------ 178.0/212.5 MB 11.9 MB/s eta 0:00:03
------------------------------ ----- 178.5/212.5 MB 11.9 MB/s eta 0:00:03
------------------------------ ----- 179.1/212.5 MB 11.7 MB/s eta 0:00:03
------------------------------ ----- 179.6/212.5 MB 11.7 MB/s eta 0:00:03
------------------------------ ----- 180.2/212.5 MB 11.5 MB/s eta 0:00:03
------------------------------ ----- 180.7/212.5 MB 11.7 MB/s eta 0:00:03
------------------------------ ----- 181.2/212.5 MB 11.3 MB/s eta 0:00:03
------------------------------ ----- 181.6/212.5 MB 11.1 MB/s eta 0:00:03
------------------------------ ----- 182.0/212.5 MB 11.1 MB/s eta 0:00:03
------------------------------ ----- 182.4/212.5 MB 10.9 MB/s eta 0:00:03
------------------------------ ----- 182.9/212.5 MB 10.9 MB/s eta 0:00:03
------------------------------ ----- 183.3/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------ ----- 183.8/212.5 MB 10.7 MB/s eta 0:00:03
------------------------------- ---- 184.2/212.5 MB 10.7 MB/s eta 0:00:03
------------------------------- ---- 184.7/212.5 MB 10.7 MB/s eta 0:00:03
------------------------------- ---- 185.1/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------- ---- 185.6/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------- ---- 186.1/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------- ---- 186.6/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------- ---- 187.1/212.5 MB 10.6 MB/s eta 0:00:03
------------------------------- ---- 187.6/212.5 MB 10.4 MB/s eta 0:00:03
------------------------------- ---- 188.0/212.5 MB 10.4 MB/s eta 0:00:03
------------------------------- ---- 188.6/212.5 MB 10.4 MB/s eta 0:00:03
------------------------------- ---- 189.1/212.5 MB 10.4 MB/s eta 0:00:03
-------------------------------- --- 189.7/212.5 MB 10.4 MB/s eta 0:00:03
-------------------------------- --- 190.2/212.5 MB 10.4 MB/s eta 0:00:03
-------------------------------- --- 190.7/212.5 MB 10.2 MB/s eta 0:00:03
-------------------------------- --- 191.3/212.5 MB 10.4 MB/s eta 0:00:03
-------------------------------- --- 191.8/212.5 MB 10.6 MB/s eta 0:00:02
-------------------------------- --- 192.4/212.5 MB 10.7 MB/s eta 0:00:02
-------------------------------- --- 193.0/212.5 MB 10.9 MB/s eta 0:00:02
-------------------------------- --- 193.5/212.5 MB 10.9 MB/s eta 0:00:02
-------------------------------- --- 194.2/212.5 MB 11.1 MB/s eta 0:00:02
-------------------------------- --- 194.8/212.5 MB 11.3 MB/s eta 0:00:02
--------------------------------- -- 195.3/212.5 MB 11.5 MB/s eta 0:00:02
--------------------------------- -- 196.0/212.5 MB 11.7 MB/s eta 0:00:02
--------------------------------- -- 196.6/212.5 MB 11.9 MB/s eta 0:00:02
--------------------------------- -- 197.2/212.5 MB 12.1 MB/s eta 0:00:02
--------------------------------- -- 197.8/212.5 MB 12.1 MB/s eta 0:00:02
```

```
     ---------------------------------- -- 198.5/212.5 MB 12.4 MB/s eta 0:00:02
     ---------------------------------- -- 199.1/212.5 MB 12.6 MB/s eta 0:00:02
     ---------------------------------- -- 199.7/212.5 MB 12.8 MB/s eta 0:00:01
     ---------------------------------- -- 200.4/212.5 MB 12.8 MB/s eta 0:00:01
     ---------------------------------- - 201.1/212.5 MB 13.1 MB/s eta 0:00:01
     ---------------------------------- - 201.7/212.5 MB 13.1 MB/s eta 0:00:01
     ---------------------------------- - 202.1/212.5 MB 13.1 MB/s eta 0:00:01
     ---------------------------------- - 202.4/212.5 MB 12.6 MB/s eta 0:00:01
     ---------------------------------- - 202.8/212.5 MB 12.6 MB/s eta 0:00:01
     ---------------------------------- - 203.1/212.5 MB 12.1 MB/s eta 0:00:01
     ---------------------------------- - 203.5/212.5 MB 11.9 MB/s eta 0:00:01
     ---------------------------------- - 203.8/212.5 MB 11.7 MB/s eta 0:00:01
     ---------------------------------- - 204.2/212.5 MB 11.5 MB/s eta 0:00:01
     ---------------------------------- - 204.6/212.5 MB 11.3 MB/s eta 0:00:01
     ---------------------------------- - 204.9/212.5 MB 11.1 MB/s eta 0:00:01
     ---------------------------------- - 205.3/212.5 MB 10.9 MB/s eta 0:00:01
     ---------------------------------- - 205.8/212.5 MB 10.7 MB/s eta 0:00:01
     ---------------------------------- - 206.1/212.5 MB 10.6 MB/s eta 0:00:01
     ---------------------------------- - 206.6/212.5 MB 10.4 MB/s eta 0:00:01
     ----------------------------------  207.0/212.5 MB 10.2 MB/s eta 0:00:01
     ----------------------------------  207.4/212.5 MB 10.1 MB/s eta 0:00:01
     ----------------------------------  207.8/212.5 MB 9.9 MB/s eta 0:00:01
     ----------------------------------  208.3/212.5 MB 9.8 MB/s eta 0:00:01
     ----------------------------------  208.7/212.5 MB 9.6 MB/s eta 0:00:01
     ----------------------------------  209.2/212.5 MB 9.5 MB/s eta 0:00:01
     ----------------------------------  209.6/212.5 MB 9.4 MB/s eta 0:00:01
     ----------------------------------  210.1/212.5 MB 9.2 MB/s eta 0:00:01
     ----------------------------------  210.6/212.5 MB 9.1 MB/s eta 0:00:01
     ----------------------------------  211.1/212.5 MB 9.1 MB/s eta 0:00:01
     ----------------------------------  211.6/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.0/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ----------------------------------  212.5/212.5 MB 9.0 MB/s eta 0:00:01
     ---------------------------------- 212.5/212.5 MB 5.9 MB/s eta 0:00:00
Collecting torchvision
  Downloading torchvision-0.22.0-cp310-cp310-win_amd64.whl (1.7 MB)
     ---------------------------------------- 0.0/1.7 MB ? eta -:--:--
     ---------- ---------------------------- 0.6/1.7 MB 11.5 MB/s eta 0:00:01
     ------------------------ -------------- 1.1/1.7 MB 13.9 MB/s eta 0:00:01
     ---------------------------------------- - 1.6/1.7 MB 11.5 MB/s eta 0:00:01
     ---------------------------------------- 1.7/1.7 MB 10.9 MB/s eta 0:00:00
Collecting torchaudio
  Downloading torchaudio-2.7.0-cp310-cp310-win_amd64.whl (2.5 MB)
     ---------------------------------------- 0.0/2.5 MB ? eta -:--:--
     -------- ------------------------------- 0.5/2.5 MB 16.5 MB/s eta 0:00:01
     ---------------- ----------------------- 1.1/2.5 MB 11.3 MB/s eta 0:00:01
     ------------------------- -------------- 1.6/2.5 MB 11.5 MB/s eta 0:00:01
     ---------------------------------- ----- 2.2/2.5 MB 11.7 MB/s eta 0:00:01
     ---------------------------------------- 2.5/2.5 MB 11.3 MB/s eta 0:00:00
Collecting networkx
  Downloading networkx-3.4.2-py3-none-any.whl (1.7 MB)
     ---------------------------------------- 0.0/1.7 MB ? eta -:--:--
     ----------- ---------------------------- 0.5/1.7 MB 16.2 MB/s eta 0:00:01
     ------------------------ -------------- 1.1/1.7 MB 13.9 MB/s eta 0:00:01
```

```
     ---------------------------------------- - 1.7/1.7 MB 13.4 MB/s eta 0:00:01
     ---------------------------------------- 1.7/1.7 MB 11.0 MB/s eta 0:00:00
Requirement already satisfied: jinja2 in c:\users\kemal\appdata\local\packages\py
thonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python
310\site-packages (from torch) (3.1.6)
Collecting sympy>=1.13.3
  Downloading sympy-1.14.0-py3-none-any.whl (6.3 MB)
     ---------------------------------------- 0.0/6.3 MB ? eta -:--:--
     --- ------------------------------------ 0.6/6.3 MB 17.5 MB/s eta 0:00:01
     ------- -------------------------------- 1.1/6.3 MB 14.3 MB/s eta 0:00:01
     ---------- ----------------------------- 1.7/6.3 MB 13.8 MB/s eta 0:00:01
     ------------- -------------------------- 2.4/6.3 MB 13.6 MB/s eta 0:00:01
     ----------------- ---------------------- 3.0/6.3 MB 13.5 MB/s eta 0:00:01
     -------------------- ------------------- 3.6/6.3 MB 13.4 MB/s eta 0:00:01
     ------------------------ --------------- 4.2/6.3 MB 13.4 MB/s eta 0:00:01
     ---------------------------- ----------- 4.8/6.3 MB 13.4 MB/s eta 0:00:01
     ---------------------------------- ----- 5.5/6.3 MB 13.5 MB/s eta 0:00:01
     ---------------------------------------- - 6.1/6.3 MB 13.5 MB/s eta 0:00:01
     ---------------------------------------- 6.3/6.3 MB 13.0 MB/s eta 0:00:00
Collecting filelock
  Downloading filelock-3.18.0-py3-none-any.whl (16 kB)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\kemal\appdat
a\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\lo
cal-packages\python310\site-packages (from torch) (4.13.2)
Collecting fsspec
  Downloading fsspec-2025.5.0-py3-none-any.whl (196 kB)
     ---------------------------------------- 0.0/196.2 kB ? eta -:--:--
     ----------------------------------  194.6/196.2 kB 12.3 MB/s eta 0:00:01
     ---------------------------------------- 196.2/196.2 kB 4.0 MB/s eta 0:00:00
Requirement already satisfied: numpy in c:\users\kemal\appdata\local\packages\pyt
honsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packages\python3
10\site-packages (from torchvision) (2.2.6)
Collecting pillow!=8.3.*,>=5.3.0
  Downloading pillow-11.2.1-cp310-cp310-win_amd64.whl (2.7 MB)
     ---------------------------------------- 0.0/2.7 MB ? eta -:--:--
     ----- ---------------------------------- 0.4/2.7 MB 8.5 MB/s eta 0:00:01
     ------------ --------------------------- 0.9/2.7 MB 9.2 MB/s eta 0:00:01
     ------------------ --------------------- 1.3/2.7 MB 10.6 MB/s eta 0:00:01
     ------------------------ --------------- 1.8/2.7 MB 10.5 MB/s eta 0:00:01
     ---------------------------------- ----- 2.3/2.7 MB 10.5 MB/s eta 0:00:01
     ---------------------------------------  2.7/2.7 MB 10.0 MB/s eta 0:00:01
     ---------------------------------------- 2.7/2.7 MB 9.5 MB/s eta 0:00:00
Collecting mpmath<1.4,>=1.1.0
  Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)
     ---------------------------------------- 0.0/536.2 kB ? eta -:--:--
     ---------------------------- -------- 419.8/536.2 kB 13.2 MB/s eta 0:00:01
     ---------------------------------------- 536.2/536.2 kB 8.5 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\kemal\appdata\local\pa
ckages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-packag
es\python310\site-packages (from jinja2->torch) (3.0.2)
Installing collected packages: mpmath, sympy, pillow, networkx, fsspec, filelock,
torch, torchvision, torchaudio
Successfully installed filelock-3.18.0 fsspec-2025.5.0 mpmath-1.3.0 networkx-3.4.
2 pillow-11.2.1 sympy-1.14.0 torch-2.7.0 torchaudio-2.7.0 torchvision-0.22.0
2.7.0+cpu
CUDA available: False
```

In [3]:
```
# Sanity-check, we should get 4 outputs if the model is correctly vectorized!
temp = model(w=torch.rand(size=(1,6)), x=torch.rand(size=(4,5)))
temp
```

```
# output that was provided (double check):
#tensor([0.7839, 0.7999, 0.7693, 0.8770])
```

Out[3]: tensor([0.6672, 0.7226, 0.7351, 0.7392])

The Likelihood function

In [4]:
```
def log_lik(y_true: Tensor, y_hat: Tensor) -> Tensor:
    """
    For one or more observations, where y_true is the true label (0 or 1)
    and y_hat contains predicted probabilities [0,1], computes the (log)
    likelihood summed over all observations.

    Returns a single-element tensor.
    """
    # Avoid log(0) by clamping y_hat within (eps, 1-eps)
    eps = 1e-8
    p = y_hat.clamp(min=eps, max=1.0 - eps)
    # Log-likelihood: sum_i [y_i*log(p_i) + (1-y_i)*log(1-p_i)]
    ll = (y_true * torch.log(p) + (1 - y_true) * torch.log(1 - p)).sum()
    return ll
```

In [5]:
```
# Sanity-check, should be a single element here:
log_lik(y_true=torch.tensor(data=[1,0,1,0], dtype=torch.float), y_hat=temp)

# output that was provided (double check):
# tensor(-3.9837)
```

Out[5]: tensor(-3.3387)

The Evidence Lower BOund (ELBO)

Optimizing the ELBO (maximization) is the same as minimizing the KL divergence. Here, the students shall implement a **Monte Carlo** approximation.

According to the slides, this is how it's done:

1. Sample from approximate posterior distribution $q_\phi(\theta)$.
   - Direct sampling methods should be used (we have a well-defined mean-field approximation here).
   - Apply reparameterization trick to train the model. Note: Do **not** use amortized variational inference here.
2. Estimate the ELBO using stochastic gradient-based optimization.

Remember that the Bayesian framework tells us something about the *model*. The variational approach allows us to **empirically** estimate the overall goodness of fit of our model.

In order to do that sufficiently well, we need more than just point estimates. Recall that we do **not** attempt to find some best point estimates for our data, but rather a

distribution over them. In order for that to work well, we need to test many different parameter constellations and average over those results. In other words, we need to draw many different sets of possible variational distribution parameters and check how well these allow our model, on average, to predict the constant observations.

Recall the definition of the ELBO:

$$\text{ELBO} = \mathbb{E}_{q_\phi(\mathbf{w})}\left[\log\left(p(\mathbf{y}|\mathbf{X},\mathbf{w})\right)\right] - D_{\text{KL}}\left(q_\phi(\mathbf{w})\|p(\mathbf{w})\right). \tag{7}$$

For simplicity here, assume there will be the following "loops":

1. Outermost loop is governed by the epoch (E).
2. The next loop is over the stochastic mini-batches of observations (batches of size N).
   - For each batch, you draw $w^{(s)} \in 1 \dots S$ **new** different sets of parameter configurations from your variational distribution.
   - Here, you'll be using the reparameterization trick.
3. Loop over S:
   - (a) For each configuration s_i, you will have to multiply (sum) the (log) likelihood of each observation under the current likelihood function (as parameterized by $w^{(s)}$).
   - (b) Next, calculate the KL-divergence between our approximate posterior (the mean-field approximation of independent Gaussians) and our prior (which is a diagonal standard normal distribution). **Attention**:
     - You use either, the **analytical** or the **Monte Carlo** approximation of the KL-divergence. However, the analytical one is essentially *outside the expectation* (because it does not average over $w^{(s)}$), whereas the MC-approximation should perhaps be an addend/subtrahend to (a).
     - In effect, the analytical KL-divergence is calculated only **once** per batch, using the **current** variational parameters (i.e., as they were after the last optimizer's step or after initialization for first step).
   - Calculate (a) - (b), according to previous remark.
4. Average the results from step 3 (considering the remark about analytical/MC version of the KL-divergence). It needs to be an average because the ELBO is an expectation (a weighted mean) over all possible realizations of $s_i \in S$. For each individual batch, you have now an average idea of:
   - The *expected* data likelihood.
   - How strongly your prior and approximate posterior diverge from one another.
5. Do not accumulate results across batches at this point.
   - It is better to compute a gradient for each batch and apply parameter updates. Frequent, incremental updates work better in practice.

Notes:

- As S approaches ∞, the MC-approximation of the KL-divergence will be equal to the analytical solution.

- For the MC-approximation, a good S is perhaps $50 - 500$.

Putting it all together

Let's implement the "loops" from above. We will create stochastic mini-batches of our data to compute gradients on. For now, we will not implement a loop for epochs.

The ELBO for a list of parameters sets S and a mini-batch of length N is defined as follows:

$$\text{ELBO}(\phi) \approx \left(\frac{1}{S} \sum_{s=1}^{S} \underbrace{\left[\sum_{i=1}^{N} \log\left(p(y_i|x_i, w^{(s)})\right) \right]}_{\text{log-sum of i.i.d. observations}} \right) - \underbrace{D_{\text{KL}}(q_{\mu,\sigma}(w)||p(}_{\text{using the current } \mu}$$

Also note that we can use Monte Carlo to approximate the KL-diverger

$$D_{\text{KL}}(q_{\mu,\sigma}(w)||p(w)) = \mathbb{E}_{w \sim q_{\mu,\sigma}(w)} \left[\log \left(\frac{q_{\mu,\sigma}(w)}{p(w)} \right) \right],$$

$$= \frac{1}{S} \sum_{s=1}^{S} \left[\log \left(q_{\mu,\sigma} \left(w^{(s)} \right) \right) - \log \left(p \left(w^{(s)} \right) \right) \right].$$

Also note that the KL-divergence between two normals has an analytic

$$D_{\text{KL}}(q_{\mu,\sigma}(w)||\mathcal{N}(0, I)) = \frac{1}{2} \sum_{j=0} \left(\mu_j^2 + \sigma_j^2 - 1 - \log\left(\sigma_j^2\right) \right).$$

Also note that the ELBO including the MC-approximation of the KL-d

$$\text{ELBO}(\phi) \approx \frac{1}{S} \sum_{s=1}^{S} \underbrace{\left[\sum_{i=1}^{N} \log\left(p(y_i|x_i, w^{(s)})\right) \right]}_{\text{log-sum of i.i.d. observations}} - \underbrace{\left[\log \left(q_{\mu,\sigma} \left(w^{(s)} \right) \right) - \right.}_{\text{MC-approx. of K}}$$

Implementation

Here, implement everything you need.

Data Preparation and Dimensionality Reduction

Start by reducing the dimensionality of your data. Choose a combination of dataset and polynomial degree that is not too low and not too high. Aim for at least 30% explained variance and at most 15 components.

Report the (sum of the) explained variance before you proceed.

Remember the basics: splitting, randomness, scaling, etc.

In [3]:
```
import numpy as np
import pandas as pd
import torch
from torch import Tensor
from torch.utils.data import TensorDataset, DataLoader
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from typing import Literal

data = np.load(file='2048d_sms_spam_albert-xlarge-v2.npz')
# data = np.load(file='300d_sms_spam_fasttext_pca.npz')

embeddings = data['X']
labels = data['y']


# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    embeddings, labels, test_size=0.2, random_state=42, stratify=labels
)

# Standardize
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test  = scaler.transform(X_test)

# PCA: at least 30% variance, at most 15 components
pca = PCA(n_components=min(15, X_train.shape[1]))
pca.fit(X_train)
cumvar = np.cumsum(pca.explained_variance_ratio_)
n_components = int(np.searchsorted(cumvar, 0.30) + 1)
print(f"Using {n_components} PCA components capturing {cumvar[n_components-1]:.3

pca = PCA(n_components=n_components)
X_train = pca.fit_transform(X_train)
X_test  = pca.transform(X_test)

scaler2 = StandardScaler() # scale and standardize again after PCA
X_train = scaler2.fit_transform(X_train)
X_test = scaler2.transform(X_test)


# Convert to torch tensors
X_train_t = torch.tensor(X_train, dtype=torch.float32)
y_train_t = torch.tensor(y_train, dtype=torch.float32)
X_test_t  = torch.tensor(X_test, dtype=torch.float32)
y_test_t  = torch.tensor(y_test, dtype=torch.float32)
```

```python
# Mini-batch loader
BATCH_SIZE = 64
train_ds = TensorDataset(X_train_t, y_train_t)
train_loader = DataLoader(train_ds, batch_size=BATCH_SIZE, shuffle=True)

# --- MODEL AND HELPER FUNCTIONS ---

POLY_DEGREE = n_components  # use PCA components as degree

def model(w: Tensor, x: Tensor) -> Tensor:
    """
    Polynomial logistic model: z = w0 + Σ_{j=1}^D w_j * x_j^j, then sigmoid.
    - w: (..., D+1)
    - x: (N, D)
    Returns: Tensor of shape (N,) probabilities.
    """
    D = x.shape[1]
    assert w.shape[-1] == D + 1
    z = w[..., 0]
    for j in range(1, D+1):
        # More aggressive scaling of polynomial terms
        z = z + w[..., j] * (x[:, j-1] ** j) / (j * 100)
    return torch.sigmoid(z)


def log_lik(y_true: Tensor, y_hat: Tensor) -> Tensor:
    """Sum log-likelihood for Bernoulli outcomes."""
    eps = 1e-8
    p = y_hat.clamp(eps, 1 - eps)
    # Use log-sum-exp trick for numerical stability
    log_p = torch.log(p)
    log_1_p = torch.log(1 - p)
    # Add small epsilon to prevent -inf
    return (y_true * log_p + (1-y_true) * log_1_p + eps).sum()
```

```
Using 3 PCA components capturing 0.342 variance
```

Model

Below you'll find some prototypes (fill in the blanks). Again, this is just a suggestion, you can come up with your own implementation.

Here i had a really annoying issue where i would keep getting NaN values for my ELBO during training. I have commented out some print statements that helped me pinpoint the cuase of this but i will keep them here becuase if i want to experiment further with this in my free-time i would like to have this here. My conclusion for the weird bug is that the log function made it so that we would get extremely large decimal points for negative values. So we would get something like 0.000000...0001 and when it hit 0 it the log function would return -inf and when theese -inf sums get summed up, python will break and produce NaN and this will poison all future gradients.

So what i did is that i "clamp" the arguments to log so they never hit 0 and this makes sure that i dont get a division by zero in the KL and the NaNs dissappear.

In [ ]:
```
NUM_VARIATIONAL_SETS = 50

def ELBO_expected_data_likelihood(y_true: Tensor, x: Tensor, W: Tensor) -> Tenso
    """A function to calculate the first term of the ELBO.    Monte Carlo estima
    - W: shape (S, D+1)
    """
    device = y_true.device
    ll_sum = torch.zeros((), device=device)
    S = W.shape[0]
    for s in range(S):
        w_s = W[s]
        y_hat = model(w_s, x)
        # Add small epsilon to prevent log(0)
        y_hat = y_hat.clamp(min=1e-8, max=1-1e-8)
        ll = log_lik(y_true, y_hat)

        # Debug prints for first sample
        #if s == 0:
            #print(f"y_hat range: [{y_hat.min():.4f}, {y_hat.max():.4f}]")
            #print(f"log_lik: {ll:.4f}")

        ll_sum = ll_sum + ll

    return ll_sum / float(S)


def ELBO_KL_divergence_analytical(mu: Tensor, sigma: Tensor) -> Tensor:
    """The analytical version of the KL divergence.    KL[q||p] for q = N(mu, di
    """
    # Add small epsilon to prevent log(0) and ensure numerical stability
    sigma_sq = sigma**2 + 1e-6
    # Clip mu to prevent extreme values
    mu_clipped = torch.clamp(mu, min=-10.0, max=10.0)
    # Ensure sigma_sq is not too small or too large
    sigma_sq = torch.clamp(sigma_sq, min=1e-6, max=1e6)

    kl = 0.5 * torch.sum(mu_clipped**2 + sigma_sq - 1 - torch.log(sigma_sq))

    # Debug prints
    #print("Debug ELBO_KL_divergence_analytical:")
    #print(f"mu range: [{mu.min():.4f}, {mu.max():.4f}]")
    #print(f"sigma range: [{sigma.min():.4f}, {sigma.max():.4f}]")
    #print(f"sigma_sq range: [{sigma_sq.min():.4f}, {sigma_sq.max():.4f}]")
    #print(f"kl: {kl:.4f}")

    return kl


def ELBO_KL_divergence_Monte_Carlo(W: Tensor, mu: Tensor, sigma: Tensor) -> Tens
    """
    MC estimate of KL divergence via samples W ~ q.
    W shape: (S, D+1)
    """
    S, Dp = W.shape
    # Add small epsilon to prevent division by zero
    sigma_safe = sigma + 1e-8

    # Debug prints if NaN
    #if torch.isnan(W).any() or torch.isnan(mu).any() or torch.isnan(sigma).any(
```

```
        #print("Debug ELBO_KL_divergence_Monte_Carlo:")
        #print(f"W range: [{W.min():.4f}, {W.max():.4f}]")
        #print(f"mu range: [{mu.min():.4f}, {mu.max():.4f}]")
        #print(f"sigma range: [{sigma.min():.4f}, {sigma.max():.4f}]")

    # log q and log p
    log_q = -0.5 * (((W - mu) / sigma_safe) ** 2 + torch.log(2 * torch.pi * sigm
    log_p = -0.5 * (W**2 + torch.log(2 * torch.pi))

    # Debug prints if NaN
    #if torch.isnan(log_q).any() or torch.isnan(log_p).any():
        #print(f"log_q range: [{log_q.min():.4f}, {log_q.max():.4f}]")
        #print(f"log_p range: [{log_p.min():.4f}, {log_p.max():.4f}]")

    # sum over dims, mean over samples
    return torch.mean(torch.sum(log_q - log_p, dim=1))
```

In [ ]:
```
NUM_VARIATIONAL_SETS = 50
KL_DIV_TYPE = Literal['analytical', 'montecarlo']


def ELBO(use_mu: Tensor, use_sigma: Tensor, y_true: Tensor, obs: Tensor, variati

    # Reparameterization: W = mu + sigma * eps
    W = use_mu.unsqueeze(0) + use_sigma.unsqueeze(0) * variational_params_noise
    exp_lik = ELBO_expected_data_likelihood(y_true, obs, W)
    if kl == 'analytical':
        kl_div = ELBO_KL_divergence_analytical(use_mu, use_sigma)
    else:
        kl_div = ELBO_KL_divergence_Monte_Carlo(W, use_mu, use_sigma)
    elbo = exp_lik - kl_div
    outputs = (elbo,)
    if return_exp_data_lik: outputs += (exp_lik,)
    if return_kl_div:       outputs += (kl_div,)
    return outputs[0] if len(outputs)==1 else outputs
```

Training

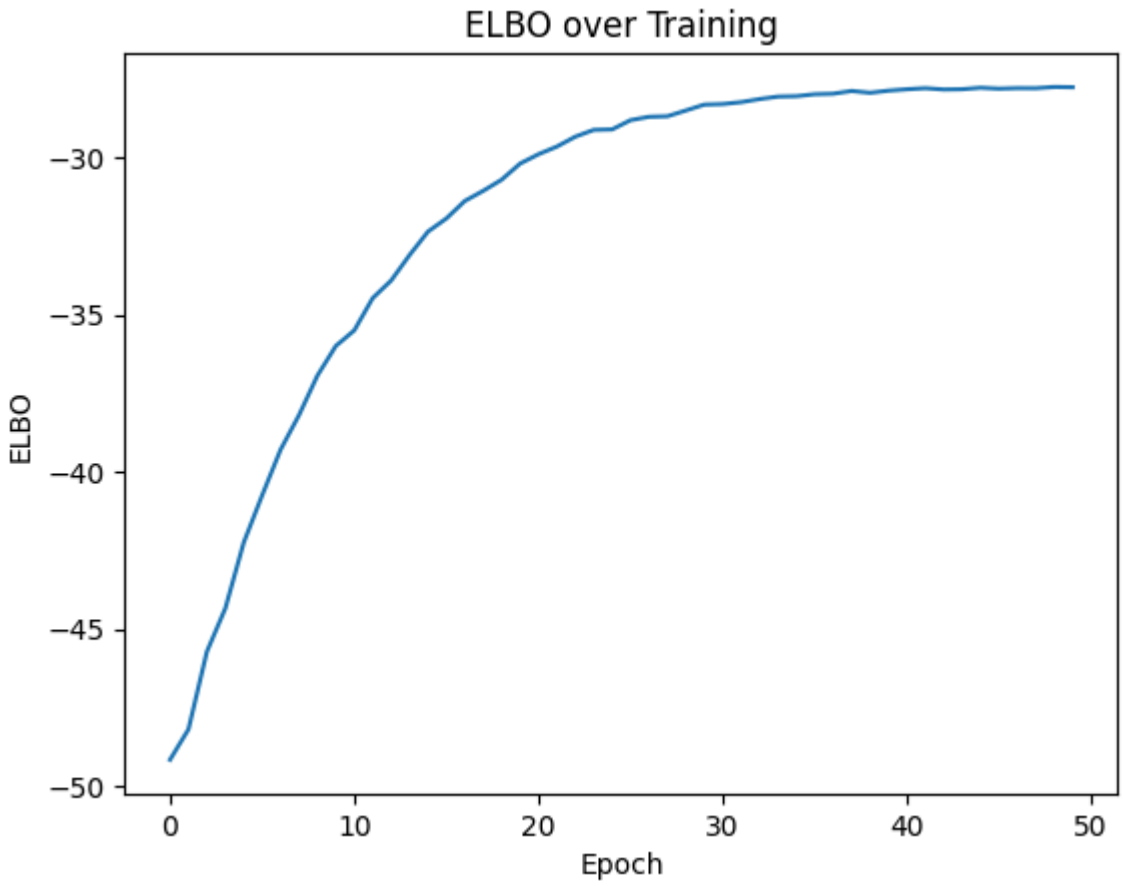
Train your model until convergence. Choose a number of iterations, batch-size, and learning rate that make sense in your scenario. Do not perform a grid search or other hyperparameter optimization. Instead, manually find some good working parameters and make your final solution just use these.

- **Plot** the training curve (i.e., plot the history for each component of the ELBO).
- **Print** the final parameters for your variational distribution after optimization.
- **Evaluate** the ELBO on the holdout dataset. Is it close? You could also do this during training.

In [ ]:
```
D_plus1 = X_train_t.shape[1] + 1
mu_param = torch.zeros(D_plus1, requires_grad=True)
log_sigma_param = torch.zeros(D_plus1, requires_grad=True)

optimizer = torch.optim.Adam([mu_param, log_sigma_param], lr=1e-3)

EPOCHS = 50
```

```
LEARNING_RATE = 0.0001
USE_KL_TYPE: KL_DIV_TYPE = 'analytical'
elbo_history = []

for epoch in range(EPOCHS):
    epoch_elbo = 0.0
    for Xb, yb in train_loader:
        optimizer.zero_grad()
        sigma = torch.exp(log_sigma_param)
        eps = torch.randn(NUM_VARIATIONAL_SETS, D_plus1)
        elbo = ELBO(mu_param, sigma, yb, Xb, eps, kl=USE_KL_TYPE)
        loss = -elbo
        loss.backward()
        optimizer.step()
        epoch_elbo += elbo.item()
    elbo_history.append(epoch_elbo / len(train_loader))
    if epoch % 10 == 0:
        print(f"Epoch {epoch:2d} ELBO: {elbo_history[-1]:.4f}")

# Final variational parameters
final_mu = mu_param.detach()
final_sigma = torch.exp(log_sigma_param).detach()
print("Final mu:", final_mu)
print("Final sigma:", final_sigma)

# Plot training ELBO
plt.plot(elbo_history)
plt.xlabel('Epoch')
plt.ylabel('ELBO')
plt.title('ELBO over Training')
plt.show()

# ELBO on test data
eps_test = torch.randn(NUM_VARIATIONAL_SETS, D_plus1)
test_elbo = ELBO(mu_param, torch.exp(log_sigma_param), y_test_t, X_test_t, eps_t
print(f"Test set ELBO: {test_elbo:.4f}")
```

```
Epoch  0 ELBO: -49.1471
Epoch 10 ELBO: -35.4965
Epoch 20 ELBO: -29.8945
Epoch 30 ELBO: -28.2986
Epoch 40 ELBO: -27.8282
Final mu: tensor([-1.6487,  0.0511,  0.1097, -0.0420])
Final sigma: tensor([0.3292, 0.9991, 1.0003, 0.9970])
```




```
Test set ELBO: -450.1004
```

In [11]:
```
test_elbo_per_sample = (test_elbo.item() / len(y_test_t))
print(f"Test set ELBO per sample: {test_elbo_per_sample:.4f}")
```

```
Test set ELBO per sample: -0.4037
```

Advanced Model Evaluation (Pick One)

In this last part of the assignment, you should select **exactly one** of the following evaluation procedures. Your task is to implement your chosen evaluation fully and clearly, produce within your notebook at least **one informative visualization**, and **provide a 100–500 word explanation** presenting:

- **Motivation:** Why you selected this particular evaluation.
- **Implementation:** How exactly your approach was implemented (with brief explanations for your visualization and choice of metrics).
- **Insights:** What interesting facts, strengths, or weaknesses were revealed from applying this evaluation.

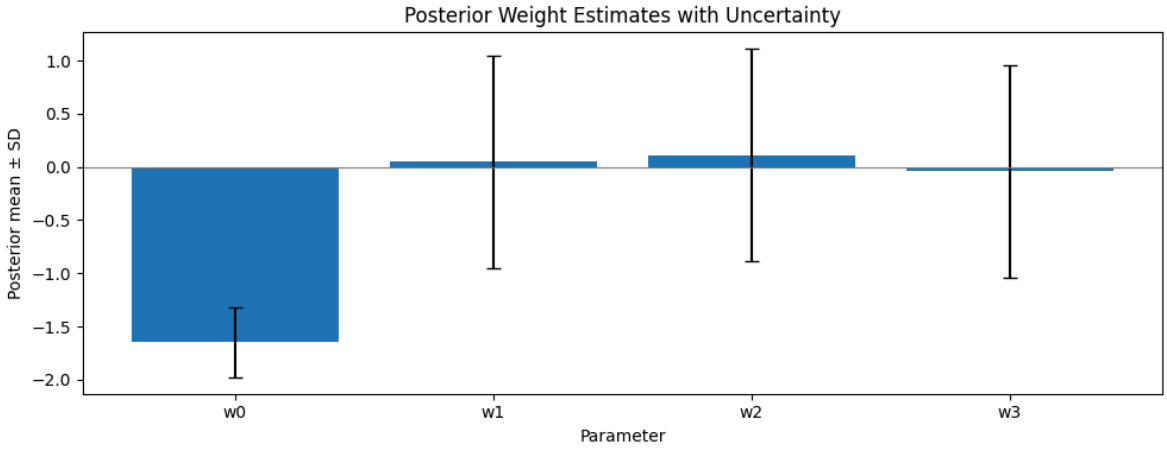
Pick one of the following:

4. **Posterior Weight Visualization**
   - Examine posterior distributions of your parameters (coefficients of your polynomial regression).

Out of the eight options, i thought it would be a good idea to evalueate the model using option 4. This is becuase i already learned a mean μ and standard deviation $´\sigma$ for all of my weights. So visualizing those as a bar plot with error bars would be pretty easy to code with a simple matplotlib figure.

In [ ]:
```
final_mu = mu_param.detach()
final_sigma = torch.exp(log_sigma_param).clamp(min=1e-3).detach()
print("Final mu:", final_mu)
print("Final sigma:", final_sigma)

# Move variational parameters to CPU numpy
weights_mu = final_mu.cpu().numpy()
weights_sigma = final_sigma.cpu().numpy()
param_names = [f"w{i}" for i in range(len(weights_mu))]

# Plot posterior means with ±1 std-error bars
plt.figure(figsize=(10, 4))
plt.bar(param_names, weights_mu, yerr=weights_sigma, capsize=4)
plt.axhline(0, color='gray', linewidth=0.8)
plt.xlabel('Parameter')
plt.ylabel('Posterior mean ± SD')
plt.title('Posterior Weight Estimates with Uncertainty')
plt.tight_layout()
plt.show()
```

```
Final mu: tensor([-1.6487,  0.0511,  0.1097, -0.0420])
Final sigma: tensor([0.3292, 0.9991, 1.0003, 0.9970])
```



After training and applying the first ELBo over epoch plot i detach the variational parameters which are `mu_param` and `log_sigma_param` and compute the sigma value. Using matplotlib i plotted a bar chart of μ for each parameter, with vertical error bars of sigma and a horizontal line. This choice of +-1 sigma error bar reflect about a 65% credible interval under the approximate gaussian posterior, giving an immediate visual cue of uncertainty magnitude.

The plot conclusively shows that only the bias term W_0 has a mu of -1.65 and sigma of 0.33 which means that the model is confident that the intercept is negative so it defaults toward the marginal spam rate. All other weights have means around 0 and sigma 1 which indicates that their posterior values collapse back to the standard normal prior. The model did NOT find enough signal in the three PCA features under the current ELBO weighting to justify moving them. This means that the model is weak because the strong

KL penalty is drowning the likelihood, so if we want to extract useful feature contributions, we will need to soften the KL term.