FICTITIOUS PREDICTION

JUSTIN CHEIGH & KATIE KEITH

1. BACKGROUND

Fictitious[1] prediction is a "prediction problem where the goal is to learn the features that enable prediction; the predictions themselves are not of interest" (Grimmer 2021 [1]). For example, Peterson and Spirling 2018 used fictitious prediction as a method to measure polarization in the Westminster Systems [2]. Specifically, they took a dataset containing the speeches and political party of various politicians over different timesteps. The idea is that times of high polarization are characterized by when it's easy to classify political party given a speech, i.e. the speeches are very polarizing. While this intuition is seemingly sound, it breaks down the minute you start looking closely. For example, the formal statement is the accuracy of predicting political party given speech is proportional to the true polarization, but what justification is there for this being a linear relationship? To begin to investigate this I provide a basic formalization of fictitious prediction below, as well as a baseline empirical pipeline that allows for synthetic evaluation.

2. BASIC PIPELINE

Here's a basic formalization of the problem of fictitious prediction, described through the context of Spirling. Let $t \in \{1, 2, \ldots, T\}$ be the timestep. We assume there exists a causal diagram of the following form:

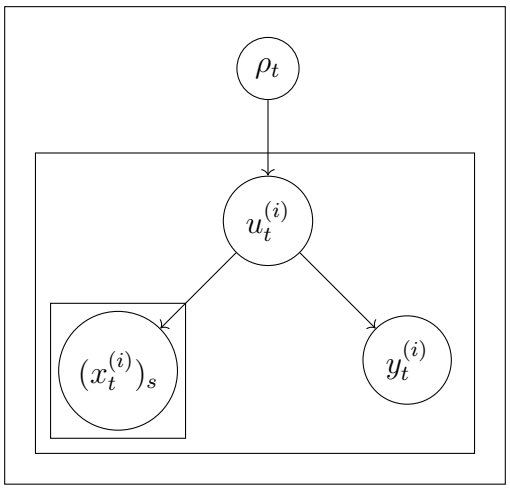

FIGURE 1. Main causal diagram of fictitious prediction.

Here, $u \in [0, 1]$ (polarization) influences $y \in \{0, 1\}$ (political party) and x (speeches). Implicit here is a subscript t, as the relationship may change over time. Our goal is to measure the true latent variable u at a specific timestep. More specifically, let I_t denote the set of individuals at timestep t and $U_t := \{u_{i,t} \mid i \in I_t\}$ to be their inherent polarization. We want to measure $\rho_t := \mathbb{E}[U_t]$. However, it's immediately obvious that we may wish to measure other descriptive statistics concerning U_t; for instance, what is $\mathrm{Var}(U_t)$?

Fictitious prediction makes the following assertion: the accuracy of a model trained to predict $\hat{y}$ given x is approximately ρ_t. Formally, let $\mathbb{P}_{\theta_t}$ be a classifier trained to predict y given x. Then the prediction can be defined by

$$\hat{y}_i = \mathbb{1}\left(\mathbb{P}_{\theta_t}\left[y_i = 1 \mid x_i\right] > .5\right),$$

May 25, 2025.

[1]All the public code for this project can be found on this repo

where x_i, y_i are the speeches/political party of individual i. From here the assumption of fictitious prediction is that

$$\rho_t \approx \frac{1}{|I_t|} \sum_{i \in I_t} \delta(\hat{y}_i, y_i). \tag{2.1}$$

Obviously, this is not true in general. We said nothing about $\mathbb{P}_{\theta_t}$! What if it just predicted randomly? There should be a notion of how viable this approximation is when $\mathbb{P}_{\theta_t}$ is a "good" vs. "bad" classifier. Should there be further assumptions? For example, maybe the sequence $\{\rho_t\}_{t=1}^T$ has certain "nice" properties. Also, why accuracy? We lose a lot of information by cutting off the probabilities: another option is cross entropy/log loss:

$$\mathrm{L}_{t,BCE} = -\frac{1}{|I_t|} \sum_{i \in I_t} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)).$$

To explore this, we begin with a fully synthetic data generating process:

3. Empirical Setup

TABLE 1. Variable Descriptions

Name	Default	Description
Population size (N)	1000	number of (x, y) pairs per timestep
Lexicon size (L)	10	number of words that are (right, left, neutral)
Vocab size (V)	$3L$	total vocabulary size (always 3L)
Speech length (S)	15	number of words per speech
Timesteps (T)	1500	number of timesteps per experiment
Alpha (α)	3	true polarizations $\sim \mathrm{Beta}(\alpha, \beta)$
Beta (β)	3	see above
Alpha mult (α_m)	2	individual polarizations sampled from Beta with alpha α_m
Epsilon (ϵ)	0.05	expected proportion of speech with neutral words

Algorithm 1 Data Generation

for $t \in \{1, 2, \ldots, T\}$ **do**
  $\rho_t \sim \mathrm{Unif}(.5, .95)$
  $\sigma_t = 0.175\rho_t^2 - .3625\rho_t + 0.1875$
  $\alpha_t = \rho_t \left(\frac{\rho_t(1-\rho_t)}{\sigma_t} - 1 \right)$
  $\beta_t = (1 - \rho_t) \left(\frac{\rho_t(1-\rho_t)}{\sigma_t} - 1 \right)$
  for $n \in \{1, 2, \ldots, N\}$ **do**
    $u_i \sim 2\left(\pi \cdot \mathrm{Beta}(\alpha_t, \beta_t) + (1 - \pi) \cdot \mathrm{Beta}(\beta_t, \alpha_t)\right) - 1$
    $y_i = \mathbb{1}(u_i \geq 0)$
    $\texttt{left} = \left[\underbrace{\frac{(1-\epsilon) \cdot \left(1 - \frac{u_i+1}{2}\right)}{L}, \cdots, \frac{(1-\epsilon) \cdot \left(1 - \frac{u_i+1}{2}\right)}{L}}_{L \text{ times}} \right]$
    $\texttt{right} = \left[\frac{(1-\epsilon) \cdot \left(\frac{u_i+1}{2}\right)}{L}, \cdots, \frac{(1-\epsilon) \cdot \left(\frac{u_i+1}{2}\right)}{L} \right]$
    $\texttt{neutral} = \left[\frac{\epsilon}{L}, \cdots, \frac{\epsilon}{L} \right]$
    $\phi_{u_i} = [\texttt{left right neutral}]^{\mathrm{T}}$
    $x_i \sim \mathrm{Multinomial}(S, \phi_{u_i})$
  end for
end for

3.1. **Generation.** Algorithm 3.1 is a basic generative model for generating synthetic data to test the central hypothesis of fictitious prediction, i.e. something of the form Equation 2.1. We begin by instantiating ρ_t to be the true polarization of the timestep. We technically require $\rho_t \in [.5, 1]$, but to avoid negative probabilities it's easier to restrict $\rho_t \in [.5, 1 - \epsilon]$.

Given ρ, we then fix a variance σ, which is directly a function of ρ; the actual function is chosen to make the resulting distributions well behaved. We wish to use a Beta distribution, so given the mean and variance we may solve the system of equations and back out what the parameters α, β must be.

The idea for the "stance" (u) of each individual is that there is a bimodal distribution, where ρ effectively parameterizes the difference in means of the two modes. Intuitively, the picture is something like Figure 3.1

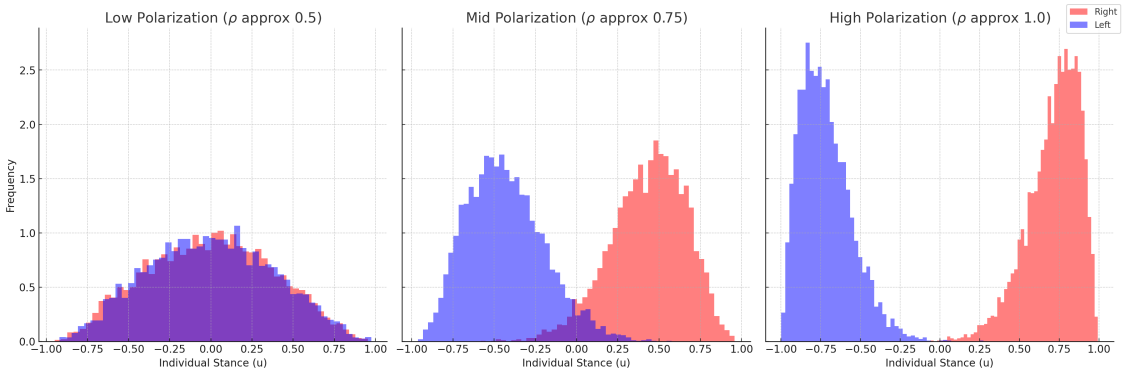

FIGURE 2. Polarization

Initially, we are making pretty strong assumptions here. We set the proportion $\pi = 0.5$. In other words, there are an equal amount of politicians from the left/right, and in particular the number of them does not rely on the true polarization. We also assume now that if the mean of the "left" distribution is m, the mean of the "right" distribution is $1 - m$; one can see this through the symmetries of the Beta distribution.

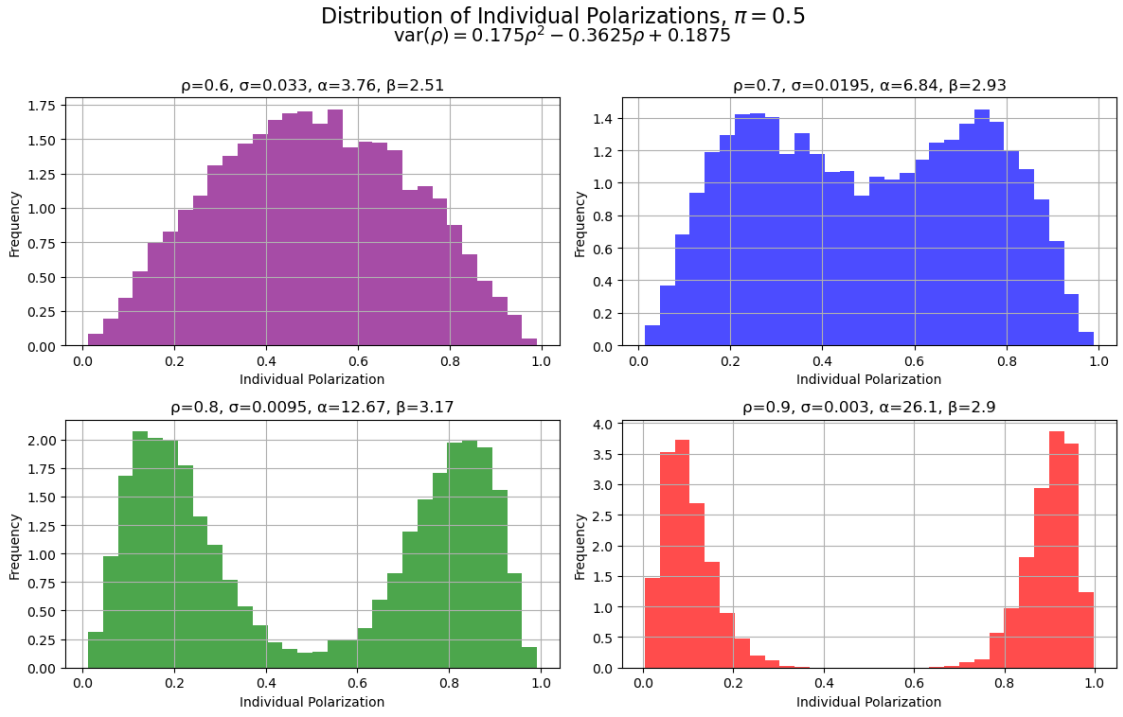

FIGURE 3. Stance Distribution

The resulting distributions look like those in Figure 3.1.

For now, we simply use an indicator to determine the political party of individual i:

$$y_i = \mathbb{1}(u_i \geq 0.5).$$

It's easy and also valid to use a Bernoulli with parameter u_i:

$$y_i = \text{Bern}(u_i).$$

We now have the problem of specifying a probability distribution over the vocabulary ϕ. Further, we would like ϕ to depend on the polarization u_i, i.e. ϕ_{u_i}. For now, we make a simplifying assumption: the vocabulary is comprised of L *right words*, L *left words*, and L neutral words.

Let's first consider the simple scenario where we have to create two distributions: $\phi_{y_i=1}$ and $\phi_{y_i=0}$. In times of high polarization ($\rho_t \approx 1$), right winged individuals are very likely to say right words, and analogous for left. In times of low polarization ($\rho_t \approx 0.5$), it's virtually indistinguishable. Here's one way to do this:

$$\phi_{y_i=0} = [\rho_t/L, \dots][(1-\rho_t-\epsilon)/L, ...][\epsilon/L, ...] \tag{3.1}$$

$$\phi_{y_i=1} = [(1-\rho_t-\epsilon)/L, ...][\rho_t/L, ...][\epsilon/L, ...]. \tag{3.2}$$

To generalize this to arbitrary $u_i \in [0, 1]$, we simply need to define ϕ in a way that

i Breaks down to 3.1 in the corresponding cases

ii Smoothly transitions otherwise.

There are many ways to do this, but the simplest is simple (linear) interpolation, which is what is used in Algorithm 3.1. Finally, we sample S points from a multinomial distribution with probability distribution ϕ_{u_i}.

Remark. To avoid vanishing gradient issues, we apply an affine transformation to the u values, such that they range form $-1, 1$ rather than $0, 1$.

Remark. In reality we implement a completely vectorized version of Algorithm 3.1, which allows us to take advantage of NumPy's fast matrix operations.

3.2. **Modeling.** Training the data is a pretty simple pipeline. We simply specify a number of folds parameter F (defaulted to 5), a model M (defaulted to GradientBoostingClassifier()), and a scoring metric S (defaulted to negative log likelihood loss). For each timestep t, we train a new instance of M to predict y_i given x_i, and we report the associated score of the best of the F folds. To introduce some notation, for a given timestep t let s_t be the associated score. For example, if S is accuracy, then Equation 2.1 simply states $s_t \approx \rho_t$.

3.3. **Results.** This setup yielded interesting preliminary results:

In Figure 3.3, one can see that polarization vs. accuracy is an even degree polynomial, whereas it's an odd degree polynomial for NLL. However, it's also obvious that neither is linear.

In Figure 3.3, one can see that the population size acts in some way as a measure of confidence.

4. Calibration

One of the ways we could get something like Figure 3.3 is if the system is *miscalibrated*. An idea we briefly looked into was calibrating the classifier, with something like *Platt Scaling*.

This is a relatively simple generation scheme, which is ideal. However, it leads to the difficult problem of inference:

5. Inference

Suppose we are given a dataset of $(x_1, y_1), \dots, (x_N, y_N)$, assumed to be generated from the DGP above. Our goal is to infer $\hat{u}_i$, or at least recover the "stance distribution" (distribution of u_i). Specifically, we are interested in the posterior of the latent variable given observed data $\vec{x}, y$:

$$\mathbb{P}(u|\vec{x}, y) = \frac{\mathbb{P}(\vec{x}, y|u)\mathbb{P}(u)}{\int \mathbb{P}(\vec{x}, y|u)\mathbb{P}(u)du}.$$

We first assume a data generating process $\mathbb{P}(x, y, u; \theta)$, where θ are unknown parameters:

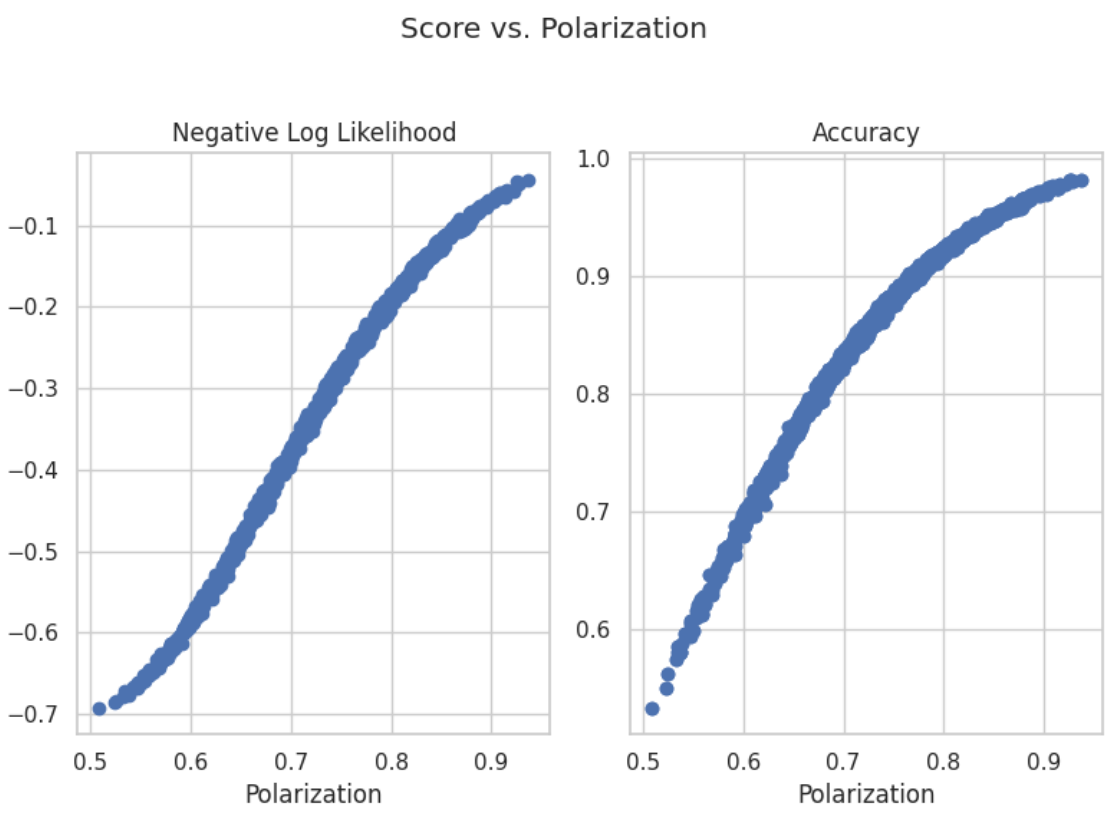

FIGURE 4. Polarization vs. Score for Accuracy/NLL

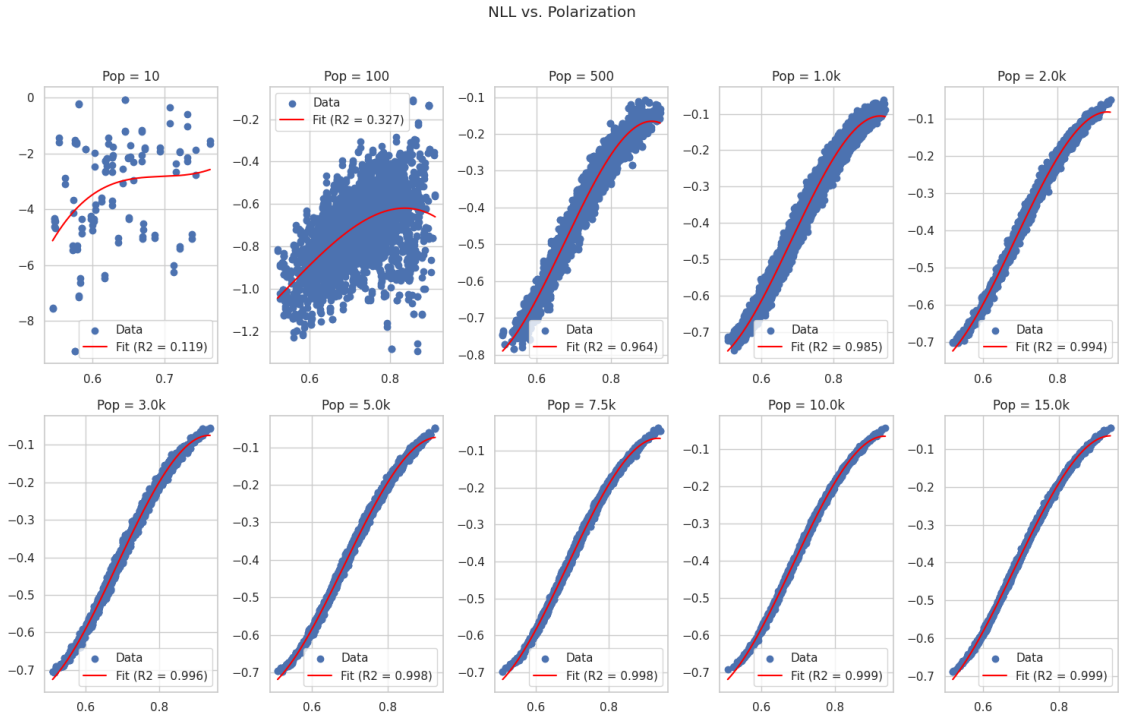

FIGURE 5. Impact of Population Size on Polarization vs. NLL.

(1) $u \sim 2 \cdot (\pi \cdot \text{Beta}(u; \alpha, \beta) + (1 - \pi) \cdot \text{Beta}(u; \beta, \alpha)) - 1$
(2) $y = 1(u \geq \lambda)$
(3) $x = \text{Softmax}(Wu), W \in \mathbb{R}^V$

- For now we assume $\lambda = 0, \pi = 0.5, V = 3 \to \theta = \{\alpha, \beta, W\}$. This gives us the following joint probability:

$$\begin{aligned} p(u, x^{(n)}, y^{(n)}; \theta) &= p(u)p(y^{(n)}|u) \prod_{s=1}^{S} p(x_s^{(n)}|u) \\ &= \left(\frac{1}{4}\text{Beta}\left(\frac{u+1}{2}; \alpha, \beta \right) + \frac{1}{4}\text{Beta}\left(\frac{u+1}{2}; \beta, \alpha \right) \right) \\ &\cdot 1\left(1(u \geq 0) = y^{(n)}\right) \prod_{s=1}^{S} \text{softmax}(Wu)_{x_s^{(n)}}. \end{aligned}$$

Here, we can get the density of u by noting u is a monotonic transform of a random variable whose density is known. Thus, applying a formula found here we get the above expression. We then need to do the following:

$$\theta^\star = \text{argmax}_\theta(\log p(x^{(1:N)}, y^{(1:N)}; \theta)) \tag{5.1}$$

$$\hat{u}_i = \text{argmax}_u(\log p(u, x^{(i)}, y^{(i)}; \theta^\star)). \tag{5.2}$$

We cannot directly maximize, so we need to use gradient information. Here's the expression for the gradient (from here):

$$\nabla_\theta \log p(x^{(1:N)}, y^{(1:N)}) = \sum_{n=1}^{N} \mathbb{E}_{p(u|x^{(n)}, y^{(n)}; \theta)}[\nabla_\theta \log p(u, x^{(n)}, y^{(n)}; \theta)]. \tag{5.3}$$

There are a number of issues here, beginning with the fact that we cannot analytically recover the posterior due to the integral being intractable. However, since $u \in [-1, 1]$, we can discretize the integral, which will be simpler than something like MCMC for sampling/using variational inference to approximate the posterior. To do this, let's write the definition of the expectation:

$$\nabla_\theta \log p(x^{(1:N)}, y^{(1:N)}) = \sum_{n=1}^{N} \int_{-1}^{1} \frac{p(x^{(n)}, y^{(n)}, u;\ \theta)}{p(x^{(n)}, y^{(n)};\ \theta)} \nabla_\theta \log p(u, x^{(n)}, y^{(n)}; \theta) \text{d}u$$

Now, we could assume u takes on G linearly spaced points in $[-1, 1]$. One way to do this is to essentially keep track of a hashmap $u \mapsto p(u, x^{(n)}, y^{(n)})$:

$$\text{Example: } \{u = -.99 : 0.03, \ldots, u = .99 : 0.007\}.$$

The sum of the values of this hashmap, by the law of total probability, is $p(x^{(n)}, y^{(n)})$. So, the posterior can be obtained by normalizing the values of the hashmap by the sum of the values. However, we can do better. Notice that when $y^{(n)} \neq 1(u \geq 0)$, $p(u, x^{(n)}, y^{(n)})$ is necessarily 0. In other words, half of the entries of our discrete distribution will be 0 in a very predictable way. Thus, we can just ignore these values and use $y^{(n)}$ as a condition.

If $y^{(n)} = 1$, have u take on G linearly spaced points in $[0, 1]$, and if $y^{(n)} = 0$ do $[-1, 0]$. This is what we do in practice. Theoretically, we're still computing:

$$\nabla_\theta \log p(x^{(1:N)}, y^{(1:N)}) = \sum_{n=1}^{N} \sum_{u} \frac{p(x^{(n)}, y^{(n)}, u;\ \theta)}{p(x^{(n)}, y^{(n)};\ \theta)} \nabla_\theta \log p(u, x^{(n)}, y^{(n)}; \theta).$$

It's convenient to move the gradient outside the sum, since then we can efficiently compute the double sum via matrices and then just sum the matrix/call .backward(). This is what I was doing initially, but it's definitely wrong, since the posterior relies on theta. To fix this, I call .detach() on the "weight" (posterior probability), which means the gradients only flow through the joint log probability (which is what we want).

6. Next Steps

Unfortunately, there are certain problems with the inference process; for example, it's not a well formalized problem, which is necessary to make this work. As a result, Katie and I have been developing new ideas (procedure for when fictitious prediction can/cannot work, ways to use fictitious prediction as a relative ordering, extend from BOW, etc.). However, none of these ideas are complete at this point.

REFERENCES

[1] Justin Grimmer, Margaret E. Roberts, and Brandon M. Stewart. Machine learning for social science: An agnostic approach. *Annual Review of Political Science*, 24(1):395–419, 2021.

[2] Andrew Peterson and Arthur Spirling. Classification accuracy as a substantive quantity of interest: Measuring polarization in westminster systems. *Political Analysis*, 26(1):120–128, 2018.