A. Negative Sampling and its connection to `word2vec`

We present the case of `word2vec` for negative sampling where the number of words and contexts is such that picking a random pair of (word, context) is with high probability not related. To make the resemblance, let us describe the intuition behind `word2vec`. Here, the task is to relate words –represented as w– with contexts –represented as c. We can theoretically conceptualize words w being related with x, and contexts being related to labels y. The negative sampling by Mikolov et al., considers the following objective function: consider a pair (w, c) of a word and a context. If this pair comes from valid data that correctly connects these two, then we can say that the data pair (w, c) came from the true data distribution; if this pair does otherwise, then we claim that (w, c) does not come from the true distribution.

In math, we will denote by $\mathbb{P}[D = 1 \mid w, c]$ as the probability that (w, c) satisfies the first case, and $\mathbb{P}[D = 0 \mid w, c]$ otherwise. The paper models these probabilities as:

$$\mathbb{P}[D = 1 \mid w, c] = \frac{1}{1 + e^{-v_c^\top v_w}},$$

where v_c, v_w correspond to the vector representation of the context and word, respectively.

Now, in order to find good vector representations $\theta := \{v_c, v_w\}$ (we naively group all variables into θ), given the data, we perform maximum log-likelihood as follows:

$$\begin{aligned}
\theta &= \arg\max_\theta \left(\prod_{(w,c)\in\mathcal{D}} \mathbb{P}[D = 1 \mid w, c, \theta] \right) \\
&\quad \cdot \left(\prod_{(w,c)\notin\mathcal{D}} \mathbb{P}[D = 0 \mid w, c, \theta] \right) \\
&= \arg\max_\theta \left(\prod_{(w,c)\in\mathcal{D}} \mathbb{P}[D = 1 \mid w, c, \theta] \right) \\
&\quad \cdot \left(\prod_{(w,c)\notin\mathcal{D}} (1 - \mathbb{P}[D = 1 \mid w, c, \theta]) \right) \\
&= \arg\max_\theta \left(\sum_{(w,c)\in\mathcal{D}} \log\left(\mathbb{P}[D = 1 \mid w, c, \theta]\right) \right) \\
&\quad + \left(\sum_{(w,c)\notin\mathcal{D}} \log\left(1 - \mathbb{P}[D = 1 \mid w, c, \theta]\right) \right) \\
&= \arg\max_\theta \left(\sum_{(w,c)\in\mathcal{D}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}}\right) \right) \\
&\quad + \left(\sum_{(w,c)\notin\mathcal{D}} \log\left(\frac{1}{1 + e^{v_c^\top v_w}}\right) \right)
\end{aligned}$$

Of course, we never take the whole dataset (whole corpus $\mathcal{D}$) and do gradient descent; rather we perform SGD by considering only a subset of the data for the first term:

$$\sum_{(w,c)\in\mathcal{D}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}}\right) \approx \sum_{(w,c)\in\text{mini-batch}} \log\left(\frac{1}{1 + e^{-v_c^\top v_w}}\right);$$

Also, we cannot consider *every* data point not in the dataset; rather, we perform *negative sampling* by selecting random pairs (according to some probability - this is important)—say P pairs:

$$\left(\sum_{(w,c)\notin\mathcal{D}} \log\left(\frac{1}{1 + e^{v_c^\top v_w}}\right) \right) \approx \sum_{p=1}^{P} \log\left(\frac{1}{1 + e^{-\tilde{v}_c^\top \tilde{v}_w}}\right),$$

where the tildes represent the "non-valid" data.

B. Alternative NS^{3}L methods

With computational efficiency in mind, we compare several methods of implementing NS^{3}L in Table 6 on the F-MNIST dataset with a small Convolutional Neural Network. We split the F-MNIST dataset into a 2,000/58,000 labeled/unlabeled split and report validation error at the end of training. Specifically, we compare:

- `Supervised`: trained only on the 2,000 labeled samples.
- `Uniform`: negative labels are selected uniformly over all classes.
- `NN`: We use the Nearest Neighbor (NN) method to the exclude the class of the NN, exclude four classes with the NNs, or to label with the class with the furthest NN.
- `Threshold`: refers to the method of section 4.3
- `Oracle`: negative labels are selected uniformly over all wrong classes.

Selecting negative labels uniformly over all classes appears to hurt performance, suggesting that negative labels must be selected more carefully in the classification setting. `NN` methods appear to improve over purely supervised training, however the effectiveness is limited by long preprocessing times and the high dimensionality of the data.

The method described in section 4.3, listed here as `Threshold`, achieves superior test error in comparison to `NN` and `Uniform` methods. In particular, it is competitive with `Oracle - 1`, an oracle which labels each unlabeled sample with one negative label which the sample is not a class of.

It is no surprise that `Oracle - 3` improves substantially over `Oracle - 1`, and it is not inconceivable to develop methods which can accurately select a small number of

negative labels, and these may lead to even better results when combined with other SSL methods.

We stress that this is not a definitive list of methods to implement negative sampling in SSL, and our fast proposed method, when combined with other SSL, already improves over the state-of-the-art.

Table 6: Test error achieved by various NS^{3}L techniques on F-MNIST with all but 2,000 labels removed. We use a small CNN trained for 50 epochs. Where applicable, the number after the dash indicates the number of negative labels per sample selected.

F-MNIST	2,000
Supervised	$17.25 \pm .22$
Uniform - 1	$18.64 \pm .38$
Uniform - 3	$19.35 \pm .33$
Exclude class of NN - 1	$17.12 \pm .15$
Exclude 4 nearest classes with NN - 1	$17.13 \pm .21$
Furthest class with NN - 1	$16.76 \pm .15$
Threshold $T = 0.03$	$16.47 \pm .18$
Threshold $T = 0.05$	$16.59 \pm .19$
Oracle - 1	$16.37 \pm .12$
Oracle - 3	$15.20 \pm .66$