Discussion (Apr 8)  Project 1  Classifier agent.

Plan:
1. Designing a classifier class
2. "Bag of Word" Feature
3. "tf-idf" feature
4. "Jax" for autograd

The "Zen" of Python programming
"Vectorization"

"Classifier agent" Class

Attributes:
- Dictionary Vocabulary: "word" -> 42
- params/weights: nparray/dict "word" -> weight
- Self: feature_map: "string" -> IRd/nparray

Methods:
- "parse()": string -> features (nparray, text, format)
- "predict()": string -> spam or not spam
- "gradient()": labeled examples (train data, self, weights) -> ∇ log(loss of self, weights) nparray
- "train()": (train data, update self)
- "sgd"/"adagrad"

Evaluation:
- "error()": target data -> IR
- "loss()": logreg loss/CE loss ->IR
- "loss()": save as file
  - Save
  - Local


```
"Feature Extractor" Class

Bow Features: 70k + items in dictionary

Vocab: ["Dear", "Prof", "This", "is", ...]

Sentence: "this is great"

\[ \text{Bow} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots \\ \end{bmatrix} \leq 75th \]

Sparse CSS array

\[ \text{index, value} \]

\[ \text{TF-IDF} = \text{TF} \times \text{IDF} \]

Term frequency, Inverse Document Frequency

\[ \text{TF} = \frac{\text{Bow}_i}{\sum_{i=1}^{d} \text{Bow}_i} \]

\[ \text{Exp} \begin{bmatrix} \frac{\theta}{3} \\ \frac{1}{3} \\ 0 \\ 0 \end{bmatrix} \]

\[ \log \left( \frac{\text{Total # of Documents}}{\text{# of times Word i appear in a Document}} \right) \]
```