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 programing

"Vectorization"

"classifier_agent" Class

Attributes: Dictionary Vocabulary: "word" → 42

params/weights: np.array / dict "word" → weight

self.feature_map: "string" → $\mathbb{R}^d$ / np.array ← self.dim

Methods:

- "parse": String → features ∈ np.array (feat_extract)
- "predict": "string" → spam or nonspam ("score")
- "gradient": labeled examples from data, self.weights: → ∇loss(self.weights) np.array.
- "train": train_data → update self.params
- "sgd" / "gd"

Evaluation

"error": "test data" → $\mathbb{R}$

"loss": logistic loss / CE loss : "test data" → $\mathbb{R}$

"logging" Save as file:

- Save
- Load

"feature_extractor" class

BoW features: 70k+ d-items in dictionary

Vocab: ["Dear", "prof", "this", "is", ...]

Sentence: "this is great"

BoW (Sentence) = [0 0 1 1 0 0 1 0 0 0] ∈ 75th

Sparse.csr_array

(index, value)

{

( )

Tf-IDF = Tf x IDF

Term frequency. Inverse Document Freq.

$TF = \frac{\overrightarrow{BoW}}{\sum_{i=1}^{d} BoW(i)}$ exp: [0 1/3 1/3 0 0 1/3 0]

$IDF(i) = \log \frac{\text{Total \# of Documents}}{\text{\# of times Word } i \text{ appeared in a Document}}$