Lecture 2

QM 701: Advanced Data Analytics

Fuqua School of Business

2024

Lecture outline

- Evaluating Classification Models
- Basic Vectorization Techniques
- Feedforward Neural Networks
- Introduction to Language Models

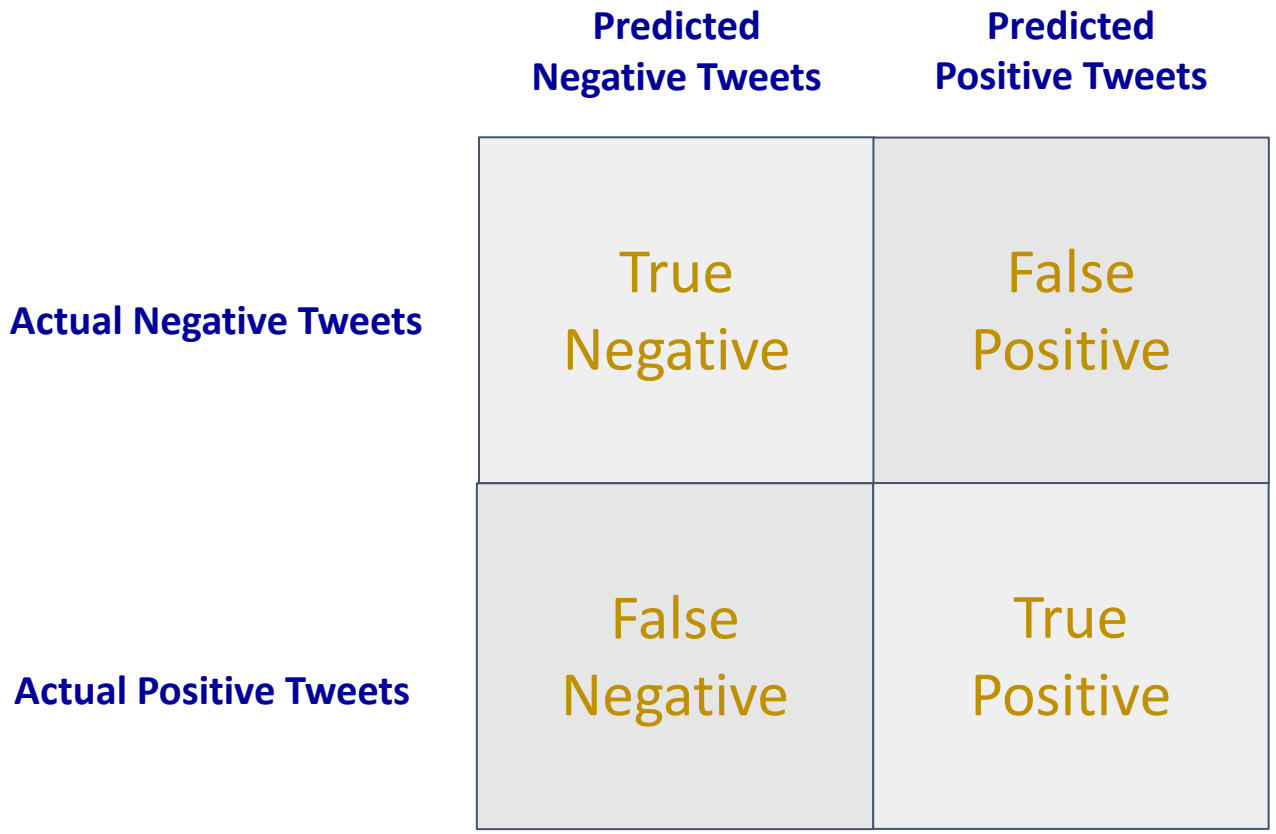
Evaluating Classification Models

Evaluating Classification Models

Question based on HW 1: how do we evaluate a sentiment analysis model, for the ‘twitter_samples’ dataset?

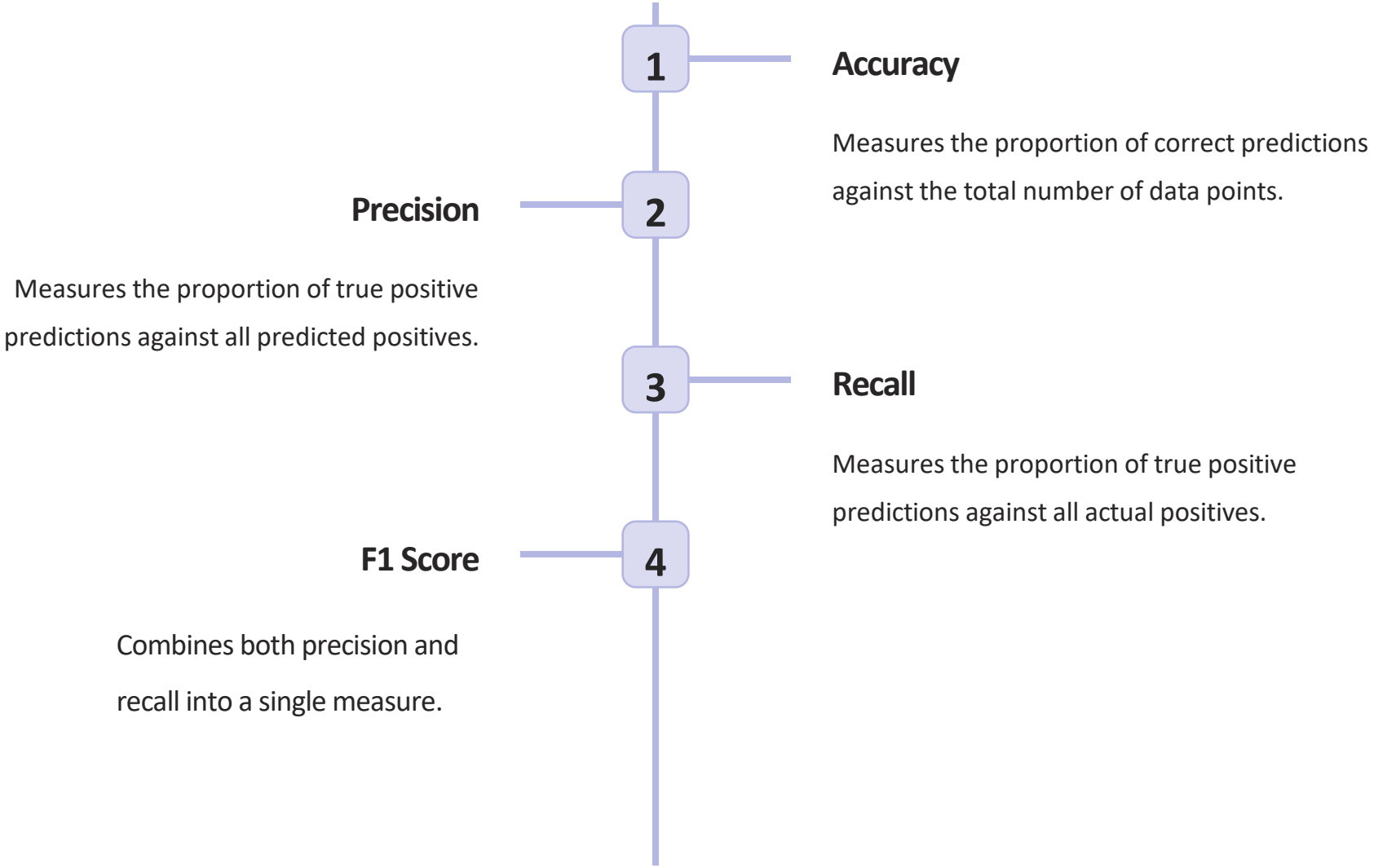
Confusion Matrix for Sentiment Analysis (from Python sklearn)

Assuming actual negative tweets are labeled as ‘0’ and actual positive tweets are labeled as ‘1’



Warning: While it the same idea, the confusion matrix in the textbook by Jurafsky and Martin is set up differently

Common Evaluation Metrics for Classification Models



See "PreClass2_Evaluation.ipynb" for examples of using these evaluation metrics.

Evaluation with Multiple Classes

- After we evaluate a classification model with multiple classes, we get a precision, recall, and F1 for each class
- To get a single score, we can aggregate the evaluations across classes using weighted average, macro-average or micro-average
  - The aggregation choice may be important with class-imbalanced dataset

A Procedure for Model Selection and Evaluation

We can also apply k-folds cross validation for picking the best model, but it would further increase the complexity of the procedure

1. Split the entire dataset into three parts: training set, validation set, and testing set.

2. Hyperparameter tuning on the training set.

   - The tuning is done by further splitting the training set into k-folds cross-validation internally, and use it to evaluate each set of hyperparameters.

     See "Class2_LR_and_FNN.ipynb" for examples of tuning hyperparameters.

     - The evaluation in this step is often done with a simple aggregated score such as accuracy or weighted f1-score

3. Evaluate the performance of different models and their hyperparameter settings on the validation set to select the best model.

   - The evaluation in this step can be more nuanced, e.g., looking at multiple performance metrics

4. Once the best model is chosen, evaluate its performance on the testing set to assess the generalization performance.

For simplicity, we skip the final evaluation step in our notebooks and use testing set to select the best model. In practice, you should always consider withhold a testing set for final evaluation.

Key Considerations for Model Selection and Evaluation

- Task at hand
  - Different tasks may require different evaluation metrics
- Characteristics of the data
  - E.g., when the number of examples for each class is uneven, we need to choose metrics that can handle class imbalance
- Algorithmic Bias

Algorithmic Bias in NLP Classification Tasks

- Algorithmic bias are typically not identified by standard evaluation metrics, as the biases is often contained the training/testing data

- Examples of Algorithmic Bias in Text Classification:
  - Certain types of language may be unfairly flagged
  - Harmful language directed at marginalized groups goes undetected

Reducing Algorithmic Bias

Some techniques to reduce algorithmic bias of NLP:

- Ensure that training data is diverse and representative of the population.
- Audit models and data for common sources of bias, such as stereotyping or underrepresentation of certain groups.
- Continuously evaluate models for bias, and update models as necessary to reduce bias over time.

Basic Vectorization Techniques

Example of Bag-of-Words (BOW) Vector Representation

Document	Text	(Feature) Vector Representation
1	dog barks man	[1, 1, 1, 0, 0, 0]
2	man barks dog	[1, 1, 1, 0, 0, 0]
3	man sits beneath tree	[1, 0, 0, 1, 1, 1]
4	man dog sits man tree	[1, 0, 2, 1, 0, 1]

Feature Names:
[dog, barks, man, sits, beneath, tree]

Note that BOW does not capture the positions of the words in each document

Example of TF-IDF Vector Representation

Document	Text	(Feature) Vector Representation
1	dog barks man	[0.14, 0.33, 0.00, 0.00, 0.00, 0.00]
2	man barks dog	[0.14, 0.33, 0.00, 0.00, 0.00, 0.00]
3	man sits beneath tree	[0.00, 0.00, 0.00, 0.25, 0.50, 0.25]
4	man dog sits man tree	[0.08, 0.00, 0.00, 0.20, 0.00, 0.20]

Feature Names:
[dog, barks, man, sits, beneath, tree]

- Intuition: the terms that appears in most of the documents should have a smaller weight
- Note that TF-IDF also does not capture the positions of the words in each document

Example of 2-gram BOW Vector Representation

Document	Text	(Feature) Vector Representation
1	dog barks man	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	man barks dog	[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
3	man sits beneath tree	[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0]
4	man dog sits man tree	[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]

Feature Names:
[dog barks, barks man, man barks, barks dog, man sits, sits beneath, beneath tree, man dog, dog sits, sits man, man tree]

Key Idea: instead of having each token as a feature, have every two consecutive tokens as a feature

- We can also have n-gram representations for n = 3, 4, or 5

Feedforward Neural Networks

Multinomial Logistic Regression Model

General Multinomial Logistic Regression Model:

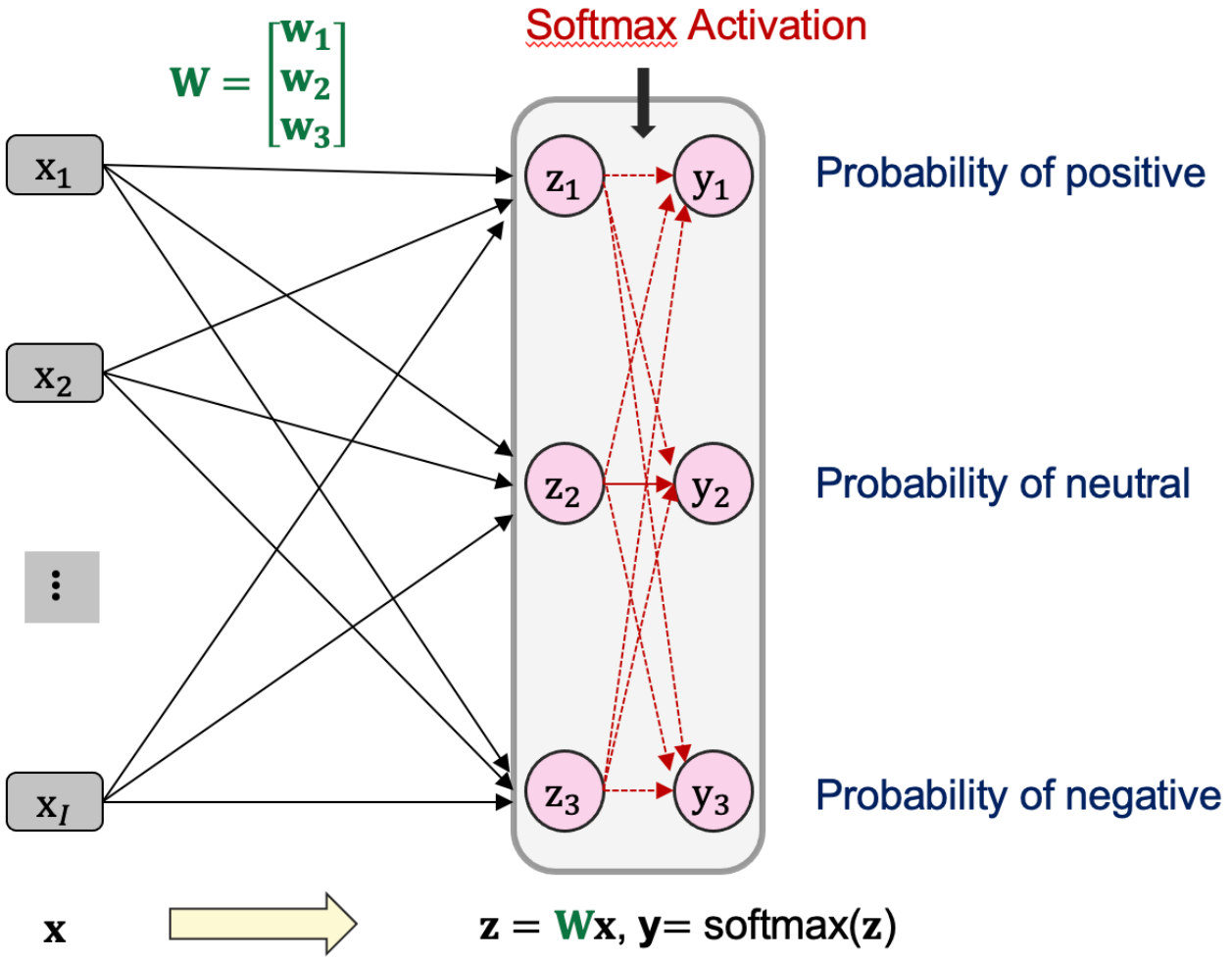
$$\mathrm{P}(\mathrm{y}=\mathrm{k}|\,\mathbf{x}) = \frac{e^{\mathbf{w}_\mathrm{k}\cdot\mathbf{x}}}{\sum_{1\le \mathrm{i}\le \mathrm{K}} e^{\mathbf{w}_\mathrm{i}\cdot\mathbf{x}}}$$

Note: the values of $[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\mathrm{K}]$ *are trained by minimizing the negative log-likelihood/cross-entropy function via stochastic gradient descent.*

- $\mathbf{w}_\mathrm{k} \cdot \mathbf{x} = w_{k1}x_1 + w_{k2}x_2 + \cdots + w_{kn}x_n$, is the score for class k given feature vector $\mathbf{x}$.
- $[\mathrm{P}(\mathrm{y}=1|\,\mathbf{x}), \mathrm{P}(\mathrm{y}=2|\,\mathbf{x}), \ldots, \mathrm{P}(\mathrm{y}=\mathrm{K}|\,\mathbf{x})] = \mathrm{softmax}(\mathbf{w}_1 \cdot \mathbf{x}, \ldots, \mathbf{w}_\mathrm{K} \cdot \mathbf{x})$
  - Some people use the notation $\mathrm{P}(\mathrm{y}=\mathrm{k}|\,\mathbf{x}) = \mathrm{softmax}(\mathbf{w}_\mathrm{k} \cdot \mathbf{x})$
- Example of softmax function: $\mathrm{softmax}(0.25, 1.23, -0.8) \rightarrow [0.249, 0.664, 0.087]$
  - softmax ≈ give a high probabilities to the larger elements

Diagram for Multinomial Logistic Regression Model

Single-Layer Neural Network:

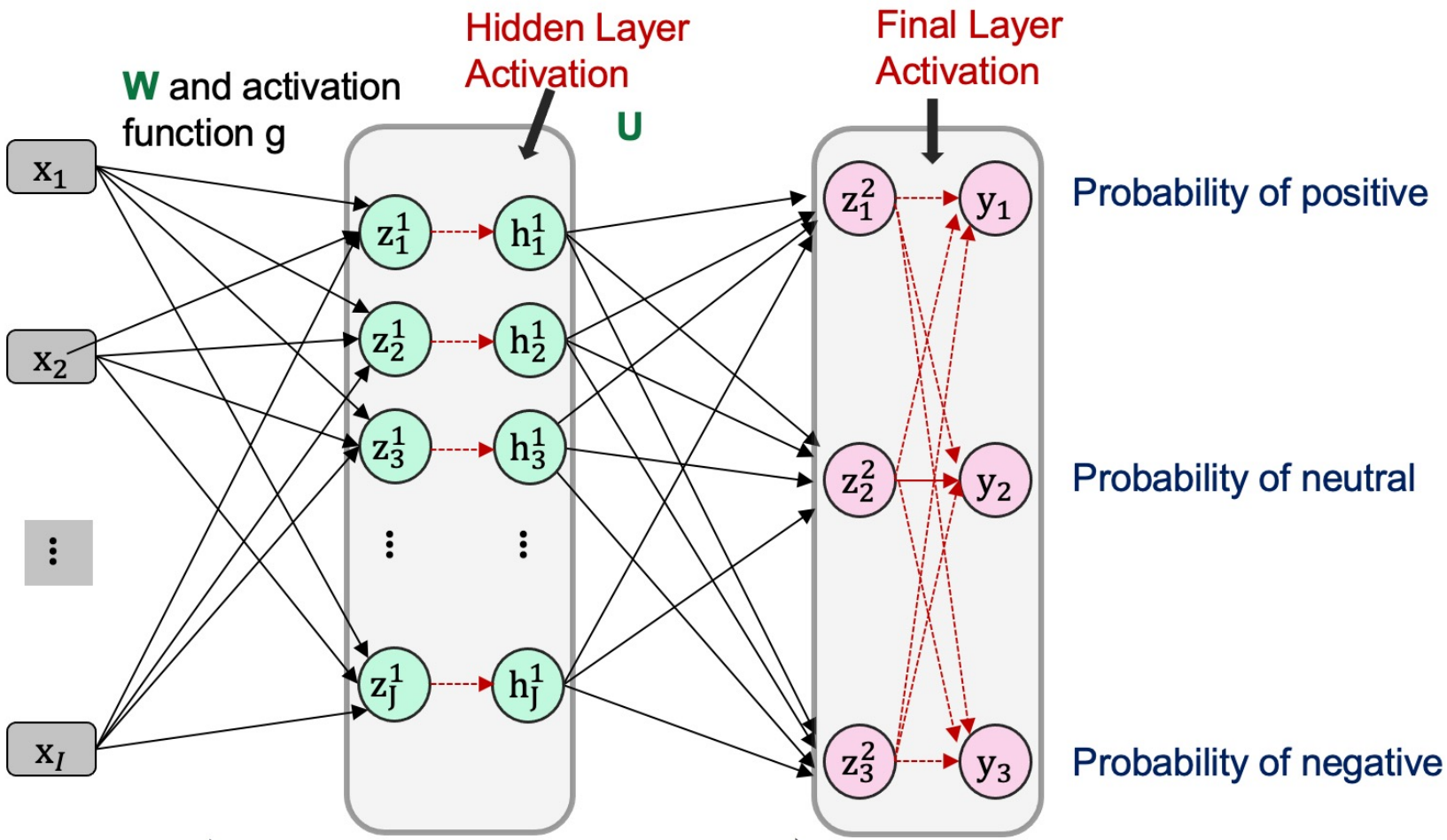


Equations:

$\mathbf{x} \Rightarrow \mathbf{z} = \mathbf{W}\mathbf{x}, \mathbf{y} = \text{softmax}(\mathbf{z})$

Diagram for 2-layer Feedforward Network

Two-Layer Feedforward Neural Network:



Equations: $\mathbf{x} \Rightarrow \mathbf{z}^1 = \mathbf{W}\mathbf{x},\ \mathbf{h}^1 = g(\mathbf{z}^1), \Rightarrow \mathbf{z}^2 = \mathbf{U}\mathbf{h}^1,\ \mathbf{y} = \text{softmax}(\mathbf{z}^2)$

Introduction to Language Models

What is a Language Model?

- A language model (LM), or causal LM, is a model that estimates the probability distribution of the next word in a sentence based on the previous texts
  - Using the estimates on the next word, LM can assign probabilities to entire sentences
- The n-gram language model “estimates the probability of the last word of an n-gram given the previous words”

Example of n-gram Language Models

Predict the word for “crystal waters shimmer under _____”

1-gram LM: P(sunlight | crystal waters shimmer under) ≈ P(sunlight)

2-gram LM: P(sunlight | crystal waters shimmer under) ≈ P(sunlight | under)

3-gram LM: P(sunlight | crystal waters shimmer under) ≈ P(sunlight | shimmer under)

Features of n-gram Language Models

- One of the oldest and most basic language models
- Trained using unlabeled text data, similar to other language models
  - In comparison, text classification models are trained on labeled data
- Number of parameters quickly increases with n, so n is typically no more than 5
- Cannot handle the long-range dependencies of natural languages
- Later in this course, we will discuss models such as LSTM and transformers that are much better at handling long-range dependencies

Homework 2

Analyze multi-categorical dataset of tweets

- Q1: Data Labeling: Map the initial emotions to the simpler categories
- Q2: Text Pre-processing: pre-process and clean the data for sentiment analysis
- Q3: Vectorization: Convert the text to data using BoW, TF-IDF, or n-gram
- Q4: Using Naive-Bayes to fit your model and evaluate the results
- Q5: Using Multinomial Logistic Regression to fit your model and evaluate the results
- Q6: Apply the TextBlob sentiment analysis to build a classification system and evaluate the results

Hint: review the notebooks posted on Canvas as you work through the questions