

CS 533: Natural Language Processing

Linear Classification


Karl Stratos



Rutgers University

Review: NLP as a Classification Problem

text \mapsto **expected human response**

Input	Output	
failed to not disappoint the dog saw the cat	-1 D N V D N PRED DET SBJ OBJ DET * the dog saw the cat	(sentiment) (POS tagging) (parsing)
산을 갔다	I went to the mountain	(translation)
SONOMA, Calif. — Wine country was shrouded in a thick layer of smoky haze here on Tuesday as firefighters continued to battle wildfires that have left at least 13 people dead and have damaged or destroyed more than 1,500 structures.	Wildfires sweep across northern California; 13 dead	(summarization)
Who was the richest man in 2020?	Jeff Bezos	(QA)
Trump spent years pushing the untrue “birther” claim that the nation’s first black president was not born in the U.S.	Trump spent years pushing the untrue “birther” claim that the nation's first black president was not born in the U.S. 	(linking)
Open the pod bay doors, HAL.	I'm sorry, Dave. I'm afraid I can't do that.	(dialogue)

Review: Classification as Optimization

- ▶ **Training:** Find model parameters that minimize some loss on training data D

$$\theta^* = \arg \min_{\theta \in \Theta} \mathbf{loss}_D(\theta)$$

- ▶ **Inference:** Given input x , find valid output with maximum score under trained parameters $\hat{\theta}$

$$y^* = \arg \max_{y \in \mathcal{Y}(x)} \mathbf{score}_{\hat{\theta}}(x, y)$$

Review: Supervised ML Pipeline

- ▶ **Problem definition:** Define a supervised learning problem.
- ▶ **Data collection:** Start with training data for which we know the correct outcome.
- ▶ **Representation:** Choose how to represent the data.
- ▶ **Modeling:** Choose a **hypothesis class** – a set of possible explanations for the connection between input (e.g., image) and output (e.g., class label).
- ▶ **Estimation:** Find best hypothesis you can in the chosen class. This is what people usually think of as “learning”.
- ▶ **Model selection:** If we have different hypothesis classes, we can select one based on some criterion.

Topic Classification with Linear Classifiers

- ▶ Problem definition
- ▶ Data collection
- ▶ Representation
- ▶ Modeling
- ▶ Estimation
 - ▶ Softmax
 - ▶ Training Objective

Problem and Data

- ▶ **Problem.** Classify a document $x \in \mathcal{X}$ to a topic $y \in \mathcal{Y}$
- ▶ **Data.** Many annotated datasets
 - ▶ **AG News:** News articles, one of 4 types (*World, Sports, Business, or Sci/Tech*)
 - ▶ **DBpedia:** Wikipedia articles, one of 14 types (*Company, Artist, Film, Animal, etc.*)
 - ▶ **Yahoo! Answers:** Online questions/answers, one of 10 types (*Health, Sports, Science & Mathematics, etc.*)
- ▶ Related: sentiment classification
 - ▶ **IMDB:** Movie reviews, one of 2 “types” (positive or negative)
 - ▶ **Yelp:** Restaurant reviews, one of 5 “types” (number of stars)
 - ▶ **Amazon:** Product reviews, one of 5 “types” (number of stars)

Topic Classification with Linear Classifiers

- ▶ Problem definition
- ▶ Data collection
- ▶ Representation
- ▶ Modeling
- ▶ Estimation
 - ▶ Softmax
 - ▶ Training Objective

Text Preprocessing

- ▶ Text data can be extremely noisy.
 - ▶ Non-language strings: HTML code, URLs
 - ▶ Noise in language: typos, ungrammatical sentences, numerical values (“87.175”)
- ▶ A lot of data cleaning/preprocessing efforts may be necessary.
 - We only retained lines that ended in a terminal punctuation mark (i.e. a period, exclamation mark, question mark, or end quotation mark).
 - We discarded any page with fewer than 5 sentences and only retained lines that contained at least 3 words.
 - We removed any page that contained any word on the “List of Dirty, Naughty, Obscene or Otherwise Bad Words”.⁶
 - Many of the scraped pages contained warnings stating that Javascript should be enabled so we removed any line with the word Javascript.
 - Some pages had placeholder “lorem ipsum” text; we removed any page where the phrase “lorem ipsum” appeared.
 - Some pages inadvertently contained code. Since the curly bracket “{” appears in many programming languages (such as Javascript, widely used on the web) but not in natural text, we removed any pages that contained a curly bracket.
 - To deduplicate the data set, we discarded all but one of any three-sentence span occurring more than once in the data set.

(Raffel et al. (2020) on cleaning Common Crawl)
- ▶ We will assume text is already reasonable clean.

Tokenization

“the dog didn't see the cat.”

- ▶ **Whitespace?** [*the, dog, didn't, see, the, cat.*] (length 6)
 - ▶ Can't handle non-whitespace splits (“cat.”)
- ▶ **Rule-based?** [*the, dog, did, n't, see, the, cat, .*] (length 8)
 - ▶ Language specific, needs experts to develop manual rules
- ▶ **Bytes???** [`\xec, \x95, . . . , \xeb, \x85, \x95`]
 - ▶ UTF-8: can encode all > 1 million valid character code points in Unicode using ≤ 4 one-byte (8-bit) code units.
 - ▶ Language agnostic, but completely unreadable. Also the space of possible token types is too big (2^{32}).

Modern NLP: automatically induce tokenization rules

- ▶ Given a budget (e.g., at most 10,000 token types), optimizing some unsupervised objective on large quantities of text
- ▶ WordPiece (Wu et al., 2016), byte-pair encoding (BPE) (Sennrich et al., 2016)

Index Mapping

Once text is cleaned and tokenized

- ▶ Every piece of text = **sequence of integers**

[the, dog, saw, the, cat] → [7, 10, 3870, 7, 13]

- ▶ **Vocabulary** $\mathcal{V} = \{1 \dots V\}$: Set of possible token types
- ▶ **Label set** $\mathcal{Y} = \{1 \dots L\}$: Set of possible label types
- ▶ More jargons in NLP:
 - ▶ **Corpus**: Unlabeled text dataset
 - ▶ **n -gram**: Sequence of n word types. For $n = 1, 2, 3$, called **unigram, bigram, trigram**

At this point we work with integers only. There is no string.

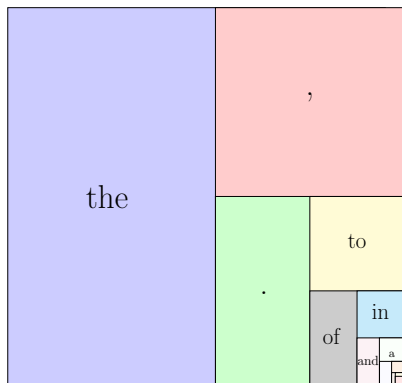
Aside: Zipf's Law

Let $w_1 \dots w_V \in \mathcal{V}$ be unigrams sorted in decreasing probability.
Empirically the following seems to hold for all $1 \leq i < V$:

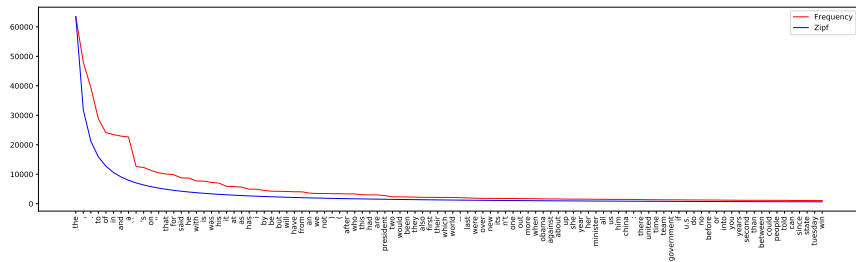
Probability of seeing w_i in text

$\approx 2 \times$ Probability of seeing w_{i+1} in text

First four words: 93% of the unigram probability mass?



Zipf's Law in Practice



Document Representation

- ▶ Represent a document as a vector to classify.
- ▶ **Bag-of-words (BOW)** representation: $x \in \{0, 1\}^V$ indicating the presence of word types

$$\text{document} = [4, 2, 2, 1, 2] \quad \rightarrow \quad x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \{0, 1\}^V$$

- ▶ What information are we losing in this representation?
- ▶ Benefits: Simple, memory-efficient (sparse), good for modeling topics (presence of “keywords”)

Topic Classification with Linear Classifiers

- ▶ Problem definition
- ▶ Data collection
- ▶ Representation
- ▶ **Modeling**
- ▶ Estimation
 - ▶ Softmax
 - ▶ Training Objective

Hypothesis Space of Linear Classifiers

- ▶ Possible models: all mappings from \mathbb{R}^V to $\{1 \dots L\}$
- ▶ **Linear classifiers:** all $(W, b) \in \mathbb{R}^{V \times L} \times \mathbb{R}^L$ defining

$$x \mapsto \arg \max_{y=1}^L \underbrace{[\underbrace{W^\top}_{L \times V} \underbrace{x}_{V \times 1} + \underbrace{b}_{L \times 1}]}_{L \times 1}]_y$$

Hypothesis Space of Linear Classifiers

- ▶ Possible models: all mappings from \mathbb{R}^V to $\{1 \dots L\}$
- ▶ **Linear classifiers:** all $(W, b) \in \mathbb{R}^{V \times L} \times \mathbb{R}^L$ defining

$$x \mapsto \arg \max_{y=1}^L \underbrace{[\underbrace{W^\top}_{L \times V} \underbrace{x}_{V \times 1} + \underbrace{b}_{L \times 1}]}_{L \times 1}]_y$$

- ▶ Can be seen as defining **input-label scores**

$$\mathbf{score}_{W,b}(x, y) := w_y^\top x + b_y =: h_y$$

where $w_y \in \mathbb{R}^V$ is the y -th column of $W = [w_1 \dots w_L]$

- ▶ $h \in \mathbb{R}^L$ called **scores** or **logits** (for x)

Clarification on Terminology

- ▶ The **parameters** of a function class is the set of variables that defines the behavior of a function in the class
 - ▶ Example: $\mathcal{F} := \{ax + b \cos(x) + c : a, b, c \in \mathbb{R}\}$
 - ▶ The parameter space associated with \mathcal{F} is \mathbb{R}^3
 - ▶ $f(x) = 5x + 2 \cos(x) - 5$ has parameter values $(5, 2, -5)$
- ▶ We will generally write
 - ▶ Θ : parameter space
 - ▶ $\theta \in \Theta$: specific parameter value
- ▶ For linear topic classifiers:
 - ▶ $\Theta = \mathbb{R}^{V \times L} \times \mathbb{R}^L$
 - ▶ $\theta = (W, b)$
 - ▶ Without loss of generality, we view $\theta \in \mathbb{R}^{(V+1)L}$ as a vector

Linear Classifier with BOW Representation

$$\mathbf{score}_{W,b}(x, y) = \sum_{i=1: x_i=1}^V [w_y]_i + b_y$$

“Interpretable”: For $x = \text{BOW}(\text{“market market up up”})$,

$$\begin{aligned} \mathbf{score}_{W,b}(x, \text{business}) &= [w_{\text{business}}]_{\text{“market”}} + [w_{\text{business}}]_{\text{“up”}} + b_{\text{business}} \\ &\gg [w_{\text{sports}}]_{\text{“market”}} + [w_{\text{sports}}]_{\text{“up”}} + b_{\text{sports}} \\ &= \mathbf{score}_{W,b}(x, \text{sports}) \end{aligned}$$

Geometry of Projections

Each label $y \in \{1 \dots L\}$ defines a hyperplane (not a subspace)

$$S(y) := \{z \in \mathbb{R}^V : w_y^\top z + b_y = 0\}$$

Claim. For document $x \in \mathbb{R}^V$, shortest distance from $S(y)$ to x is

$$\frac{w_y^\top x + b_y}{\|w_y\|}$$

The distance is *signed*. Negative if x lies on the “left” side.

Geometry of Projections: Continued

“Max distance rule”: Pick label that puts x furthest (right side)

$$x \mapsto \arg \max_{y=1}^L \frac{w_y^\top x + b_y}{\|w_y\|}$$

Linear classifier:

$$x \mapsto \arg \max_{y=1}^L w_y^\top x + b_y$$

Equivalent if weight vectors have the same norm, else different

- ▶ Only direction of w_y matters in max distance rule (if you ignore the bias term)
- ▶ $\|w_y\|$ can affect the decision of linear classifiers (even if you ignore the bias term)

Redundancy of Parameters

Suppose $\exists w_y \in \mathbb{R}^V$ such that $w_y^\top x > 0$. Then

$$\text{score}_{W,b}(x, y) = \underbrace{w_y^\top x}_{>0} + b_y$$

can be sent to ∞ without changing the direction of w_y by

- ▶ Scale $b_y \rightarrow \infty$, or
- ▶ Scale $\|w_y\| \rightarrow \infty$

So technically the bias term is unnecessary in this scenario (still necessary for encoding prior beliefs about labels).

- ▶ Typically harmless to have some redundancy in model parameters

Justification of the Claim

1. Let $S_0(\mathbf{y}) = \{z \in \mathbb{R}^V : w_{\mathbf{y}}^T z = 0\}$. This is a $(V - 1)$ -dimensional subspace **orthogonal** to $w_{\mathbf{y}} \in \mathbb{R}^V$.

Justification of the Claim

1. Let $S_0(\mathbf{y}) = \{z \in \mathbb{R}^V : w_{\mathbf{y}}^T z = 0\}$. This is a $(V - 1)$ -dimensional subspace **orthogonal** to $w_{\mathbf{y}} \in \mathbb{R}^V$.
2. The shortest distance from $S_0(\mathbf{y})$ to \mathbf{x} is $w_{\mathbf{y}}^T \mathbf{x} / \|w_{\mathbf{y}}\|$.

Justification of the Claim

1. Let $S_0(\mathbf{y}) = \{z \in \mathbb{R}^V : w_{\mathbf{y}}^T z = 0\}$. This is a $(V - 1)$ -dimensional subspace **orthogonal** to $w_{\mathbf{y}} \in \mathbb{R}^V$.
2. The shortest distance from $S_0(\mathbf{y})$ to \mathbf{x} is $w_{\mathbf{y}}^T \mathbf{x} / \|w_{\mathbf{y}}\|$.
3. Note that $S(\mathbf{y})$ shifts every point in $S_0(\mathbf{y})$ by $-b_{\mathbf{y}}/[w_{\mathbf{y}}]_i$ along each axis $i = 1 \dots V$:

$$z_i = (-\sum_{j \neq i} [w_{\mathbf{y}}]_j z_j) / [w_{\mathbf{y}}]_i \quad z \in S_0(\mathbf{y})$$

$$z_i = (-\sum_{j \neq i} [w_{\mathbf{y}}]_j z_j) / [w_{\mathbf{y}}]_i - b_{\mathbf{y}} / [w_{\mathbf{y}}]_i \quad z \in S(\mathbf{y})$$

Thus the shortest distance from $S_0(\mathbf{y})$ to $S(\mathbf{y})$ is $-b_{\mathbf{y}} / \|w_{\mathbf{y}}\|$.

Justification of the Claim

1. Let $S_0(\mathbf{y}) = \{z \in \mathbb{R}^V : w_{\mathbf{y}}^\top z = 0\}$. This is a $(V - 1)$ -dimensional subspace **orthogonal** to $w_{\mathbf{y}} \in \mathbb{R}^V$.
2. The shortest distance from $S_0(\mathbf{y})$ to \mathbf{x} is $w_{\mathbf{y}}^\top \mathbf{x} / \|w_{\mathbf{y}}\|$.
3. Note that $S(\mathbf{y})$ shifts every point in $S_0(\mathbf{y})$ by $-b_{\mathbf{y}}/[w_{\mathbf{y}}]_i$ along each axis $i = 1 \dots V$:

$$z_i = (-\sum_{j \neq i} [w_{\mathbf{y}}]_j z_j) / [w_{\mathbf{y}}]_i \quad z \in S_0(\mathbf{y})$$

$$z_i = (-\sum_{j \neq i} [w_{\mathbf{y}}]_j z_j) / [w_{\mathbf{y}}]_i - b_{\mathbf{y}} / [w_{\mathbf{y}}]_i \quad z \in S(\mathbf{y})$$

Thus the shortest distance from $S_0(\mathbf{y})$ to $S(\mathbf{y})$ is $-b_{\mathbf{y}} / \|w_{\mathbf{y}}\|$.

4. From 2 and 3, it follows that the shortest distance from $S(\mathbf{y})$ to \mathbf{x} is $(w_{\mathbf{y}}^\top \mathbf{x} + b_{\mathbf{y}}) / \|w_{\mathbf{y}}\|$.

See the provided Jupyter notebook for further illustration.

Topic Classification with Linear Classifiers

- ▶ Problem definition
- ▶ Data collection
- ▶ Representation
- ▶ Modeling
- ▶ Estimation
 - ▶ Softmax
 - ▶ Training Objective

The Softmax Function

- ▶ Given any $h \in \mathbb{R}^L$, we define $\text{softmax}(h) \in [0, 1]^L$ as

$$\text{softmax}_i(h) := \frac{\exp(h_i)}{\sum_{j=1}^L \exp(h_j)} \quad \forall i = 1 \dots L$$

- ▶ Check nonnegativity and normalization

The Softmax Function

- ▶ Given any $h \in \mathbb{R}^L$, we define $\text{softmax}(h) \in [0, 1]^L$ as

$$\text{softmax}_i(h) := \frac{\exp(h_i)}{\sum_{j=1}^L \exp(h_j)} \quad \forall i = 1 \dots L$$

- ▶ Check nonnegativity and normalization
- ▶ Check shift-invariance: for any $c \in \mathbb{R}$ (elementwise addition)

$$\text{softmax}(h + c) = \text{softmax}(h)$$

The Softmax Function

- ▶ Given any $h \in \mathbb{R}^L$, we define $\text{softmax}(h) \in [0, 1]^L$ as

$$\text{softmax}_i(h) := \frac{\exp(h_i)}{\sum_{j=1}^L \exp(h_j)} \quad \forall i = 1 \dots L$$

- ▶ Check nonnegativity and normalization
- ▶ Check shift-invariance: for any $c \in \mathbb{R}$ (elementwise addition)

$$\text{softmax}(h + c) = \text{softmax}(h)$$

- ▶ Transforms any length- L real-valued vector into a **distribution** over $\{1 \dots L\}$

$$\text{softmax}([-0.23, 1.51, -2.11]) = [0.15, 0.83, 0.02]$$

$$\text{softmax}([-\infty, -\infty, 1.00, 2.00]) = [0.00, 0.00, 0.27, 0.73]$$

Converting Logits to a Distribution by Softmax

- ▶ Softmax on logits makes any model **probabilistic**

$$p_{\theta}(y|x) := \frac{\exp(\mathbf{score}_{\theta}(x, y))}{\sum_{y'=1}^L \exp(\mathbf{score}_{\theta}(x, y'))} \quad \forall y = 1 \dots L$$

Converting Logits to a Distribution by Softmax

- ▶ Softmax on logits makes any model **probabilistic**

$$p_{\theta}(y|x) := \frac{\exp(\mathbf{score}_{\theta}(x, y))}{\sum_{y'=1}^L \exp(\mathbf{score}_{\theta}(x, y'))} \quad \forall y = 1 \dots L$$

- ▶ Softmax does not affect model inference:

$$\begin{aligned} \log p_{\theta}(y|x) &= \mathbf{score}_{\theta}(x, y) + \text{SomeFunctionOf}(x) \\ \arg \max_{y=1}^L p_{\theta}(y|x) &\equiv \arg \max_{y=1}^L \mathbf{score}_{\theta}(x, y) \end{aligned}$$

- ▶ Then why do this? For training!

Topic Classification with Linear Classifiers

- ▶ Problem definition
- ▶ Data collection
- ▶ Representation
- ▶ Modeling
- ▶ Estimation
 - ▶ Softmax
 - ▶ Training Objective

Learning as Conditional Density Estimation

- ▶ **Critical assumption.** Data comes from a population distribution p_{pop}

Learning as Conditional Density Estimation

- ▶ **Critical assumption.** Data comes from a population distribution \mathbf{pop}
- ▶ Dream classifier (but difficult to find)

$$\text{Classifier}^* = \arg \min_{\text{Classifier: } x \mapsto y} \Pr_{(x,y) \sim \mathbf{pop}} (\text{Classifier}(x) \neq y)$$

Learning as Conditional Density Estimation

- ▶ **Critical assumption.** Data comes from a population distribution **pop**
- ▶ Dream classifier (but difficult to find)

$$\text{Classifier}^* = \arg \min_{\text{Classifier: } x \mapsto y} \Pr_{(x,y) \sim \mathbf{pop}} (\text{Classifier}(x) \neq y)$$

- ▶ But we already know the (Bayes optimal) classifier satisfies

$$\text{Classifier}^*(x) = \arg \max_{y=1}^L \mathbf{pop}(y|x)$$

Learning as Conditional Density Estimation

- ▶ **Critical assumption.** Data comes from a population distribution \mathbf{pop}
- ▶ Dream classifier (but difficult to find)

$$\text{Classifier}^* = \arg \min_{\text{Classifier: } x \mapsto y} \Pr_{(x,y) \sim \mathbf{pop}} (\text{Classifier}(x) \neq y)$$

- ▶ But we already know the (Bayes optimal) classifier satisfies

$$\text{Classifier}^*(x) = \arg \max_{y=1}^L \mathbf{pop}(y|x)$$

- ▶ New goal: Estimate the conditional distribution $\mathbf{pop}(y|x)$!

$$p_{\theta}(y|x) \approx \mathbf{pop}(y|x)$$

What minimization problem can we set up to achieve this?

Cross-Entropy Loss

Cross entropy between $\mathbf{pop}(y|x)$ and $p_\theta(y|x)$

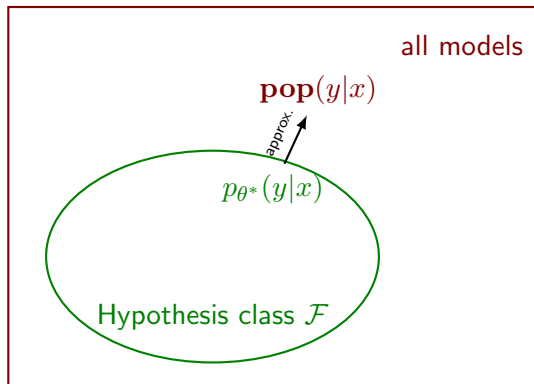
$$J(\theta) := \mathbf{E}_{(x,y) \sim \mathbf{pop}} [-\log p_\theta(y|x)]$$

If the hypothesis class Θ is expressive enough to model \mathbf{pop} ,

$$\theta^* = \arg \min_{\theta \in \Theta} J(\theta) \quad \Rightarrow \quad \mathbf{pop}(y|x) = p_{\theta^*}(y|x) \quad \forall x, y$$

If not, it will still find some projection of \mathbf{pop} onto Θ

Picture



Empirical Cross-Entropy Loss

Don't know **pop**, but can **sample** $\underbrace{(x_1, y_1) \dots (x_N, y_N)}_{\text{(training data)}} \stackrel{\text{iid}}{\sim} \mathbf{pop}$

$$\hat{J}_N(\theta) := -\frac{1}{N} \sum_{i=1}^N \log p_{\theta}(y_i | x_i)$$

By the law of large numbers, $\hat{J}_N(\theta) \rightarrow J(\theta)$ as $N \rightarrow \infty$

- ▶ $\hat{J}_N(\theta)$ is defined by N training examples as well as θ !
- ▶ But we view it only as a function of θ for optimization

Empirical Cross-Entropy Loss for Linear Classifiers

Given N iid samples of document-label pairs (x_i, y_i) from human annotators, define

$$\hat{J}_N(W, b) = \frac{1}{N} \sum_{i=1}^N \log \left(\sum_{y=1}^L \exp(w_y^\top x_i + b_y) \right) - w_{y_i}^\top x_i - b_{y_i}$$

Training: Unconstrained optimization problem

$$(W^*, b^*) = \arg \min_{W \in \mathbb{R}^{V \times L}, b \in \mathbb{R}^L} \hat{J}_N(W, b)$$

Important questions

- ▶ When is \hat{J}_N minimized for finite N ?
- ▶ Are there multiple local minima (i.e., is \hat{J}_N nonconvex)?
- ▶ Can we optimize \hat{J}_N efficiently even if N is really large?