An Experience with Text Classification in *Datadays 2019*

Majid Hajiheidari   Amirmohammad Asadi

April, 2019

Divar Posts Dataset

- Released for DataDays 2019
- One million posts

Columns

- id
- archive_by_user
- published_at
- **cat1**
- **cat2**
- **cat3**
- city
- **title**
- **desc**
- price
- image_count
- platform
- mileage
- brand
- year
- type

The Problem: Categorization

- We need to categorize posts based on other posts features;
- We only use text features(title & description)!

Features

Cat3	Cat2	Cat1	Desc	Title
fridge-and-freezer	utensils-and-appliances	for-the-home	یخچال ارج کاملا سالم	یخچال ارج
nan	childrens-clothing-and-shoe	personal	دونه ای 28 سن 7تا 9 تقریبا رنگ مناسب دختر وپسر میباشد مقطووووع پیامک پاسخگو نیستم	تعدادی کاپشن درحدنو
stereo-surround	audio-video	electronic-devices	سالم وباصدای فوق العاده قوی وباکیفیت. AUX رم وفلش هم میخوره یه ضبط دوتیکه LG آمپیلی دار هم دارم که داخله عکس مشخصه اونم تقدیم میکنم. یاعلی	سینماخانگی
light	cars	vehicles	همه امکانات رو داره	خودرو پژو۴۰۵
mobile-phones	mobile-tablet	Electronic-devices	بدون ضربه خوردگی و تعمیر	ایفون 6گری ۶۴گیگ

No. of Classes

- We concatenate three category columns into one; for example:

cat1	cat2	cat3	concatenate
vehicles	cars	light	vehicles::cars::light

- Then, we have 83 unique combinations of categories, eg. 83 classes in our classification task.

Feature Extraction

Feature extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set.

Vectorizing the Text: Count Vectorizer

An example: We want to vectorize these 4 setences[1]:

1. Hello, how are you!
2. Win money, win from home.
3. Call me now
4. Hello, Call you tomorrow?

[1] Example from Rahul Vasaikar

Vectorizing the Text: Count Vectorizer

1. We first build a vocabulary:
   $vocabulary =$
   $\{are, call, from, hello, home, how, me, money, now, tomorrow, win, you\}$
2. Then, we vectorize each sentence based on the occurness of each word:

	are	call	from	hello	home	how	me	money	now	tom...	win	you
1	1	0	0	1	0	1	0	0	0	0	0	1
2	0	0	1	0	1	0	0	1	0	0	2	0
3	0	1	0	0	0	0	1	0	1	0	0	0
4	0	1	0	1	0	0	0	0	0	1	0	1

Vectorizing the Text: Count Vectorizer

N pair of samples

Tf-idf Vectoizer

- Tf-idf stands for term frequency-inverse document frequency
- a statistical measure used to evaluate how important a word is to a document in a collection or corpus
- the tf-idf weight is composed by two terms:

TF **Term Frequency**, which measures how frequently a term occurs in a document.

$$TF(t) = \frac{\textit{Number of times term t appears in a document}}{\textit{Total number of terms in the document}}$$

IDF **Inverse Document Frequency**, which measures how important a term is

$$IDF(t) = \ln \frac{\textit{Total number of documents}}{\textit{Number of documents with term t in it}}$$

Tf-idf Vectorizer: An Example

Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., tf) for cat is then $tf(cat) = \frac{3}{100} = 0.03$. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as $idf(cat) = \ln \frac{10{,}000{,}000}{1{,}000} = 4$. Thus, the Tf-idf weight is the product of these quantities: $tf - idf(cat) = 0.03 * 4 = 0.12$.

Word Embedding

… when the input to a neural network contains symbolic categorical features (e.g. features that take one of k distinct symbols, such as words from a closed vocabulary), it is common to associate each possible feature value (i.e., each word in the vocabulary) with a d-dimensional vector for some d. These vectors are then considered parameters of the model, and are trained jointly with the other parameters.

— Page 49, Neural Network Methods in Natural Language Processing, 2017.

Word Embedding

It requires that document text be cleaned and prepared such that each word is one-hot encoded. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. **The embedding layer is used on the front end of a neural network and is fit in a supervised way using the Backpropagation algorithm**.[2]

[2] From the article What Are Word Embeddings for Text?

One-Hot Encoding for Word Embedding

1. Hello, how are you!
2. Win money, win from home.
3. Call me now
4. Hello, Call you tomorrow?

$vocabulary = \{are, call, from, hello, home, how, me, money, now, tomorrow, win, you\}$

Word	are	call	from	hello	home	how	me	money	now	tomorrow	win	you
Value	1	2	3	4	5	6	7	8	9	10	11	12

Sentence					
Hello, how are you!	4	6	1	12	0
Win money, win from home.	11	8	11	3	5
Call me now	2	7	9	0	0
Hello, Call you tomorrow?	4	2	12	10	0

Number of Parameters

Let's say that we want to embed sentences(or words) into a $\mathbb{R}^n$ vector space. If m is the size of vocabulary, our Embedding layer has $m * n$ parameters that can be fitted in a supervised way using the Backpropagation.

Classification Algorithms

We used different classifiers and applied different models on the data. The classifiers we tested are:

- Naive Bayes
- Linear Support Vector Machine(SVM)
- Passive Aggressive Classifier
- Convolutional Neural Network(CNN)

Naive Bayes Classifier

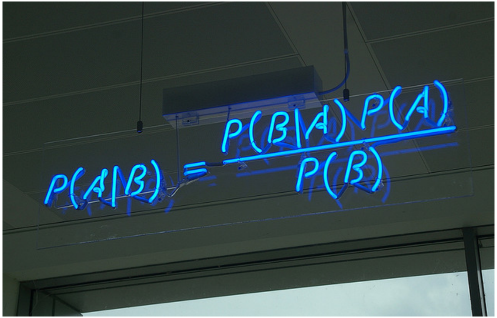


Photo by Matt Buck

Bayes Classifier: Naive One!

It is possible to show that accuracy is minimized, on average, by a very simple classifier that assigns each observation to the most likely class, given its predictor values. In other words, we should simply assign a test observation with predictor vector x_0 to the class j for which

$$P(Y = j \mid \mathbf{X} = \mathbf{x})$$

is largest.

Bayes Classifier: Naive One!

We make two assumptions:

1. $X_1, X_2, \ldots, and\ X_m$ are independent from each other;
2. $X_1, X_2, \ldots, X_m \mid Y \sim MN(\cdot, p_1, p_2, \ldots, p_m)$

$$\begin{aligned} P(Y = j \mid \mathbf{X} = (x_1, x_2, \ldots, x_m)) &= \frac{P(\mathbf{X} = (x_1, x_2, \ldots, x_m) \mid Y = j) \cdot P(Y = j)}{P(\mathbf{X} = \mathbf{x})} \\ &= \frac{P(X_1 = x_1 \mid Y = j) \cdot \ldots \cdot P(X_m = x_m \mid Y = j) \cdot P(Y = j)}{P(\mathbf{X} = \mathbf{x})}. \end{aligned}$$

$$\begin{aligned} \hat{y} &= \underset{j \in classes}{\arg\max} \frac{P(X_1 = x_1 \mid Y = j) \cdot \ldots \cdot P(X_m = x_m \mid Y = j) \cdot P(Y = j)}{P(\mathbf{X} = \mathbf{x})} \\ &= \underset{j \in classes}{\arg\max} P(X_1 = x_1 \mid Y = j) \cdot \ldots \cdot P(X_m = x_m \mid Y = j) \cdot P(Y = j). \end{aligned}$$

Bayes Classifier: Naive One!

Let's dive into code!

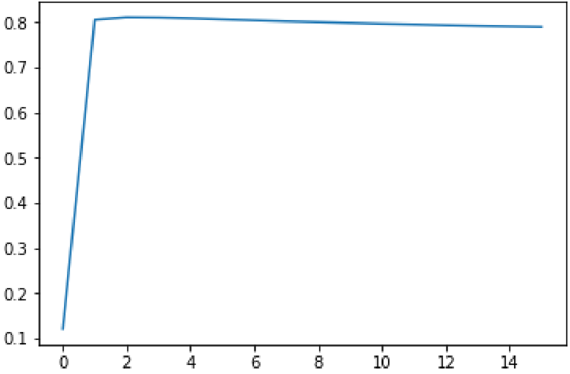
Hyperparameters

Two important hyperparameters:

1. Size of the vocabulary;
2. Laplace/ Lidstone smoothing parameter(α).

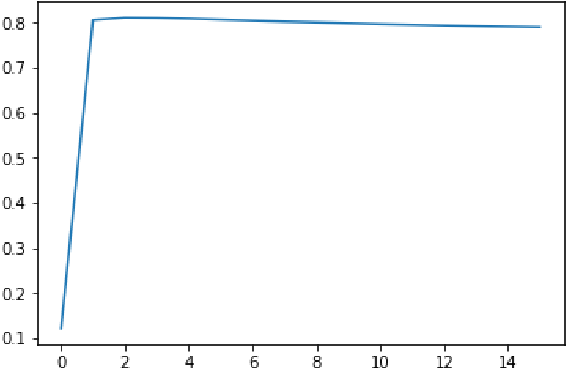
Size of Vocabulary

We can determine the size of our vocabulary.



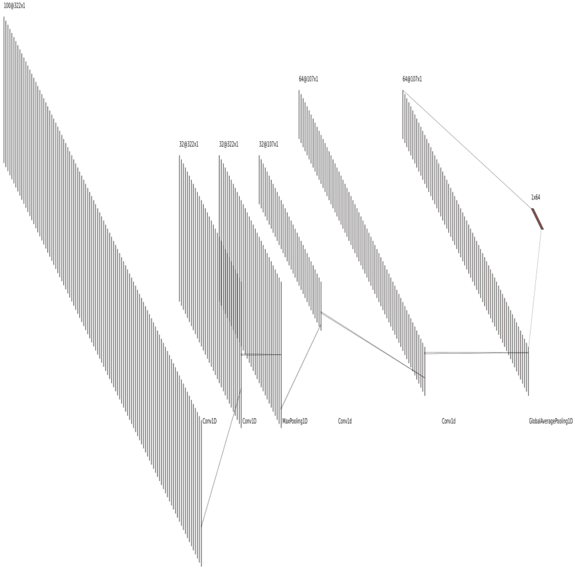
It is convex!

Laplace/ Lidstone Smoothing Parameter(α)

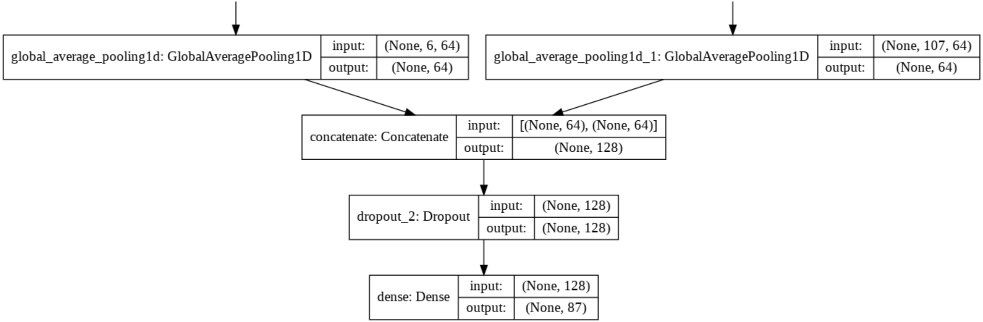


It is convex! (to be completed)

CNN Over Embedding Layer



CNN Over Embedding Layer



CNN Over Embedding Layer

	سامسونگ	سونی	لوستر	پراید
سامسونگ	0	1.9844	6.6001	4.9251
سونی	1.9844	0	6.3962	4.8678
لوستر	6.6001	6.3962	0	5.8193
پراید	4.9251	4.8678	5.8193	0

Majid Hajiheidari, Amirmohammad Asadi

Linear SVM

- A non-probablistic classifier
- A discriminative classifier formally defined by a separating hyperplane
- The algorithm outputs an optimal hyperplane which categorizes new examples
- A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class

Linear SVM: Behind the Scene

Given training vectors $x_i \in \mathbb{R}^p$, i=1,...,n, in two classes, and a vector $y \in \{1, -1\}^n$, SVM classifier solves the following primal problem:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$
$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i,$$
$$\zeta_i \geq 0, i = 1, ..., n$$

Its dual is:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
$$\text{subject to } y^T \alpha = 0$$
$$0 \leq \alpha_i \leq C, i = 1, ..., n$$

Linear SVM: Behind the Scene(cont'd)

where e is the vector of all ones, $C > 0$ is the upper bound, Q is an n by n positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. Here, training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ. The decision function is:

$$\operatorname{sgn}\left(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho\right)$$

Passive Aggressive

- A margin based online learning algorithm
- Perfect for classifying massive streams
- Easy to implement and very fast
- **Passive**: if correct classification, keep the model;
- **Aggressive**: if incorrect classification, update to adjust to this misclassified example
- See `http://koaning.io/passive-agressive-algorithms.html` for further reading

A Comparison among Models

Thanks for your attention!

Codes in slides (in my GitHub):(github link)
Divar posts dataset:(divar link)
Any questions?