Linear Text classification

Problem defintion: Given a text document, assign i
label $y \in \mathcal{Y}$ where is the set of possible labels
applications:

- Spam filter:
- Sentiment: $\mathcal{Y} = \{$Positive, negati
- Genre classification: $\mathcal{Y} = \{$sport $\cdots\}$

Bag-of-words representation of a document

A typical representatio
mathematically a vector of word counts:

$$x = [1, 5, 0, 3, 1, \dots]$$

where x_j is the cou ctor is the size of the vocabular he vector for all documents in a set are of the same length for apple-to-apple co

- "Vocabulary" here can't be understoo language. For instance, it could include all bigrams in a collection of documents.
- Alternatives to word counts include simple presence 1 or absence 0 of a word, tf/idf of a word, etc.
- By using word count we dropped all word order information

Feature function

- Not all words are eq a particular label. assign a score to each word in the vocabulary to indicate the "compatibility" with the label, e.g., "basketball" has a high compatibility with *sports*, "Gryffindo compatibility with *fiction*.
- These comp *s* and they are arranged in a ve
- Given a bag-of $\boldsymbol{\theta}$, we predict the label y by computing the total compa en $\boldsymbol{x}$ and y.
- In a linear function, this compatibility score is the inner product of between the weights $\boldsymbol{\theta}$ and a *feature function*:

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{\Psi}(\boldsymbol{x}, y)$$

$$\boldsymbol{\Psi}(\boldsymbol{x}, y) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y) = \sum_i \theta_j f_j(\boldsymbol{x}, y)$$

More on feature function

- The feature function has two arguments, the word counts $\boldsymbol{x}$ and the label y.
- It will return an feature vector where each elem vector might be:

$$f_j(\boldsymbol{x}, y) = \quad \text{tion} \quad 0,$$

- In this case the size of the feature vector is the size of the vocabulary, but it doesn't have to be.
- The output of the feature function also doesn't have to be the word count.

Shape of the feature vector

$$f(x, y = 1) = [x; \underbrace{0; 0; \cdots ; 0}_{(K-1)\times V}]$$

$$f(x, y = 2) = [\underbrace{0; 0; \cdots ; 0}_{V}; x; \underbrace{0; 0; \cdots ; 0}_{(K-2)\times V}]$$

$$f(x, y = K) = [\underbrace{0; 0; \cdots ; 0}_{(K-1)}; x]$$

where K is size of the label set, $\underbrace{0; 0; \cdots ; 0}_{(K-1)\times V}$ is a vector of $(K-1) \times V$ zeros and the semicolon indicates vertical concatenation.

Note: Think of a feature vector this way is good for mathematical presentation. It doesn't have to be implemented this way.

Bias

It is common to add a f the vector of word counts which is always 1, and then we need to add a zero to each of the zero vectors, to make the vector lengths match. The entire vector $\boldsymbol{f}(\boldsymbol{x}, y)$ wil

$$\boldsymbol{f}($$

$$\boldsymbol{f}(\boldsymbol{x}, y = 2) = [\underbrace{0; 0; \cdots}_{V} \quad \underbrace{0}_{1)}]$$

$$\boldsymbol{f}(\boldsymbol{x}, y = K) = [\underbrace{0; 0; \cdots ; 0}_{(K-1)\times(V+1)}; \boldsymbol{x}]$$

Bias: What's its effect on classification if it is the only feature?

Example "vocabulary" V

Vocabulary for a collection of docu

A	NEG	not funny at all
D	POS	funny stor
E	POS	good stor

$V = \{$ not, funny, painful, ok, overall, story, good, jokes $\}$

From vocabulary to feature function

Vocabulary for a collection of documents

A	NEG	not funny at all
B	NEG	painful, not fu

$V = \{$ not, funny, painful, ok, overall, story $\}$

$$f_1(\boldsymbol{x}, y) = \begin{cases} x_{not}, & \text{if} \quad y = \text{NEG} \\ 0, & \text{Otherwise} \end{cases}$$

From vocabulary to feature function

Vocabulary for a collection of documents

A	NEG	not funny at all
B	NEG	painful, not fu

$V = \{$ not, funny, painful, ok, overall, story $\}$

$$f_2(\boldsymbol{x}, y) = \begin{cases} x_{funny}, & \text{if} \quad y = \text{NEG} \\ 0, & \text{Otherwise} \end{cases}$$

From vocabulary to feature function

Vocabulary for a collection of documents

A	NEG	not funny at all
B	NEG	painful, not fu
C	NEG	ok overall

$V = \{$ not, funny, painful, ok, overall, story $\}$

Feature vector for A:

$$\boldsymbol{f}(\boldsymbol{x} = \text{featurize}(A), y = NEG) = [1; 1; 0; 0; 0; 0; 0; 0; 1; \underbrace{0; 0; \cdots; 0}_{(3-1)\times(8+1)}]$$

From vocabulary to feature function

Voca

A	NEG	not funny at all
B	NEG	painful, not fu
C	NEU	ok overall
D	POS	funny story

$V = \{$ not, funny, painful, ok, overall, story $\}$

Feature vector for C:

$$\boldsymbol{f}(\boldsymbol{x} = \text{featurize}(C), y = NEU) =$$

$$[\underbrace{0; 0; \cdots; 0}_{(8+1)}; 0; 0; 0; 1; 1; 0; 0; 0; 1; \underbrace{0; 0; \cdots; 0}_{(8+1)}]$$

From vocabulary to feature function

Voca

A	NEG	not funny at all
B	NEG	painful, not fu
C	NEU	ok overall
D	POS	funny story

$V = \{$ not, funny, painful, ok, overall, story $\}$

Feature vector for E

$$\boldsymbol{f}(\boldsymbol{x} = \text{featurize}(E), y = POS) =$$

$$[\underbrace{0; 0; \cdots; 0}_{(3-1)\times(8+1)}; 0; 0; 0; 0; 0; 0; 1; 1; 1]$$

The importance of feature functions

- The performance of a model to a large extent depends on the use of proper feature functions

- A lot of research went to find the most effective f developing m

- Models that c y don't perform as well

- The most effective features differ from ta relies on a good understanding of the problem at hand and domain knowledge

Assigning weights to features

Now we know about features. What about the weights ($\boldsymbol{\theta}$)?

- There are many different ways to estimat $\boldsymbol{\theta}$. That's why we have different machine le

- We find the optimal value of $\boldsymbol{\theta}$ with a set of training samples of size N: $\{\boldsymbol{x}^{1:N}, y^{1:N}\}$

Probability preliminaries

- Joint probability: $P(X = a, Y = b)$ or $P(a, b)$ where X and Y are random variables and a and b o the random variables

- Conditional $\frac{P(X=a,Y=b)}{P(Y=b)}$

- Bayes' theor $\frac{X=a)P(X=a)}{(Y=b)}$
- Marginalization (sum rule): $P($ $Y)$

- Independence: $P(Y|X) = P(Y$ (X), $P(X, Y) = P(X)P(Y)$
- Conditional independence: $P(X, Y|Z) = P(X|Z)P(Y|Z)$

Naïve Bayes: the objective

The objective is to maxi … labeled training docum … s the number of documents. This is known as the **maximum likelihood estimation**.

The goal of the training process is to find the weights $\boldsymbol{\theta}$ that maximizes this likelihood:

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{i=1}^{N} p(\boldsymbol{x}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta})$$

The notation $p(\boldsymbol{x}^{1:N}, y^{1:N}; \boldsymbol{\theta})$ indicate that $\boldsymbol{\theta}$ is a parameter of the probability function. Symbols in bold indicate a vector of variables rather than a single variable.

“Independent and Identically Distributed”

- “A collection of random variables is **nd identically distributed** if each rando e probability d independen

- Often shortened as *i.i.d*
- The basis on which we can break down the jo all samples into the product of the probability of each sample

The generative story of Naïve Bayes

The probability $p(\boldsymbol{x}^{1:N}, y^{1:N}; \boldsymbol{\theta})$ is defined through a **generative model**, an idealized random process that has generated the observed data. The algorithm that describes the ge underlying the Naïve Bayes classifier with the p $\Theta = \{\boldsymbol{\mu}, \phi\}$:

Algorithm 1 G ssification model

for Instance $i \in \{1, 2, \cdots, N\}$ **d**
  Draw the label $y^{(i)} \propto$ Categorical($\boldsymbol{\mu}$);
  Draw the word counts $\boldsymbol{x}^{(i)} | y^{(i)} \propto$ Multinomial($\phi^{(i)}$).
end for

Multinomial distribution

$$P_{X,Y}(\boldsymbol{x}^{(i)}, y^{(i)}) = P_{X|Y}(\boldsymbol{x}^{(i)} | y^{(i)}) \times P_Y(y^{(i)})$$

$P_{Y|X}$ is a **multinomial**, which a probabilisti vectors of non-negative counts. The probability this distribution is:

$$P_{multi}(\boldsymbol{x}, \phi) = B$$

$$B(\boldsymbol{x}) = \frac{\left(\sum_{j=1}^{V} x_j\right)!}{\prod_{j=1}^{V}(x_j!)}$$

Crucially, $B(\boldsymbol{x})$ is a multinomial coefficient that does not depend on ϕ, and can usually be ignored.

Parameter estimation

The generative story ab[illegible]

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta})$$

into

$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\mu}) = \sum_{i}^{N}$ [illegible] $^{(i)}$ [illegible] $(y^{(i)}; \boldsymbol{\mu})$

$= \sum_{i=1}^{N} \log B(\boldsymbol{x}^{(i)}) + \sum_{j=1}^{V}$ [illegible] $+ \sum_{i=1}^{N} \log \mu_{y^{(i)}}$

Maximum-likelihood estimation chooses $\boldsymbol{\phi}$ and $\boldsymbol{\mu}$ that maximize the log-likelihood of $\mathcal{L}$. Because we want these parameters to be probabilities, the solution must obey the following constraints:

$$\sum_{j=1}^{V} \phi_{y,j} = 1 \quad \forall y$$

Parameter Estimation

After incorporating the the constraints by adding a set of Lagrange multipliers, we get this new objective (we focus on the $\phi_{y,j}$ for now):

$$\ell(\Phi_y) = \sum_{i:y^{(i)}=y} \quad \phi_{y,j} - 1 \Bigg)$$

Differentiating with respect to the paramete ields,

$$\frac{\partial \ell(\phi_y)}{\partial \phi}_{y,j} = \sum_{i:y^{(i)}=y} x_j^{(i)} / \phi_{y,j} - \lambda$$

Parameter Estimation

There is a closed form solution to this, which can be obtained by setting each element in the vector of derivatives equal to zero:

$$\lambda\phi_{y,j} = \sum_{i:y^{(i)}=y} x_j^{(i)}$$

$$\phi_{y,j} \propto \sum_{i:y^{(i)}=y} x_j^{(i)} = \sum_{i=1}^{N} \delta\left(y^{(i)}=y\right) x_j^{(i)} = \text{count}(y,j)$$

where $\delta\left(y^{(i)}=y\right)$ is an indicator function which returns one if $y^{(i)}=y$. The symbol $\propto$ indicates that $\phi_{y,j}$ is proportional to the right-hand side of the equation

Parameter Estimation

- Recall the constraint that ϕ_y is a vector of probabilities: $\sum_{j=1}^{V} \phi_{y,j} = 1$. We have an exact solution:

$$\phi_{y,j} = \frac{count}{\quad}$$

- Similarly we can arrive at:

$$\mu_y = \frac{co}{\sum_{y' \in K} count(y')}$$

As is often the case, the result of the mathematical derivation (that needs to turned into code) is usually often much simpler:

Smoothing

Laplace smoothing: add α

$$\phi_{y,j} = \frac{\alpha + count}{V\alpha + \sum_{j'=1} co}$$

The **bias - varianc**

- Unbiased cla ta, yielding poor performance on unseen data.
- But if the smoothing is too large, the resulti **underfit** instead. In the limit of $\alpha \to \infty$, there is zero variance: you get the same classifier, regardless of the data.

How to determine the best α?

Grid Search

Setting hyperparameters with grid search: parameters and hyperparameters

- Try a set of values and find the one that maximize accuracy, bu
- The goal is to ma en data, so we shouldn't be d it on a *development* or *tuning*
- We should also set aside another set called set that you measure system performance on.
- If the data set is too small, use *cross-validation*

Prediction with Naïve Bayes

$$\hat{y} = \operatorname{argmax} \log \; (\; , y; \boldsymbol{\mu}, \boldsymbol{\phi})$$

$$= \operatorname{argmax} \log p(\boldsymbol{x} | y; \phi) + \log p(y; \boldsymbol{\mu})$$

Plug in the distribution from the generative story, w

$$\log p(\boldsymbol{x} | y; \phi) + \text{l} \qquad + \log \mu_y$$

$$= \log B \qquad \phi_{y,j}) + \log \mu_y$$

$$j=1$$

$$= \log B(\boldsymbol{x}) + \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y),$$

where

$$\boldsymbol{\theta} = [\boldsymbol{\theta}^{(1)}; \boldsymbol{\theta}^{(2)}; \cdots ; \boldsymbol{\theta}^{(K)}]$$

$$\boldsymbol{\theta}^{(y)} = [\log \phi_{y,1}; \log \phi_{y,2}; \cdots ; \log \phi_{y,V}; \log \mu_y]$$

Relation between mathematical models and computer science tools

- Mathematics provides the justification of why a model works the way it does, and computer science focuses with efficient algorithms and appropriate d
  - Computational algorithms often resor
    repeate nal
    implem
    Forward-Backfoward, CKY
- It's useful to think about where the mathe on ends and computational realization starts.
- The relation between a mathematical expression and its implementation in a programming language can be thought of as a translation process: it's not always word for word.

Advantages of Naïve Bayes

With a joint likelihood objective, the estimated par
Naïve Bayes model can be used for both classifica $p(y|x)$) and generation ($p(x\ y)$).

$$P(x, y) = P(x|y) \times P(y \qquad (x)$$

In practice, we rarely use Naïve Bayes for gene
used as a classification model.

Problems of Naïve Bayes

Let's say we want to inclu
morphemes).

$$
\begin{aligned}
& P(\text{word} = \textit{unfit}, \text{prefix} = \textit{un-} | y) \\
& = P(\text{prefix} = \textit{un-} | \text{word} = \textit{unfit}, y \quad \textit{nfit} | y) \\
& = 1 \times P(
\end{aligned}
$$

If we assume conditional independence,

$$
\begin{aligned}
& P(\text{word} = \textit{unfit}, \text{prefix} = \\
& \approx P(\text{word} = \textit{unfit} | y) \times P(\text{prefix} = \textit{un-} | y)
\end{aligned}
$$

Since $P(\text{word} = \textit{unfit} | y) \geq P(\text{word} = \textit{unfit} | y) \times P(\text{prefix} = \textit{un-} | y)$, conditional independence under-estimates the true probabilities of conjunction of positively correlated features.