Lecture 6: Classification with Naive Bayes

COMP90049
Introduction to Machine Learning

Semester 1, 2021

Lea Frermann, CIS

Roadmap

Last time...

- Machine learning concepts and approaches
- Review of probability
- Review of (basic) optimization

Today

- Back to Machine learning: Naive Bayes Classification
- Deriving the classifier (drawing on foundations in lecture 3)
- Finding the optimal parameters (drawing on foundations in lecture 4)
- Example and implementation

Naive Bayes Theory

A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
sunny	cool	normal	false	yes
overcast	cool	high	true	no

What (do you think) is the class of `sunny`, `cool`, `normal`, `false`?

A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
rainy	hot	normal	true	yes
rainy	hot	normal	true	no
sunny	cool	normal	false	yes
sunny	mild	high	false	no
overcast	cool	high	true	no

What (do you think) is the class of `rainy`, `hot`, `normal`, `true`?

A little thought experiment...

Given the following dataset:

Outlook	Temp	Humidity	Windy	Class
`overcast`	`mild`	`normal`	`true`	yes
`sunny`	`mild`	`normal`	`false`	yes
`overcast`	`hot`	`high`	`true`	yes
`sunny`	`cool`	`high`	`false`	yes
`rainy`	`cool`	`normal`	`true`	no
`overcast`	`hot`	`normal`	`true`	no
`sunny`	`hot`	`normal`	`false`	no
`sunny`	`mild`	`normal`	`true`	no
`rainy`	`cool`	`high`	`true`	no

What (do you think) is the class of `overcast`, `mild`, `high`, `false`?

Notation

- y label (e.g., spam, play, ...)
- x observation (e.g., email, day, ...)
- $x_m, m \in \{1, 2, ...M\}$ features of x (e.g., temperature, word, ...)
- (x^i, y^i) observation-label pair; the ith data point
- $\theta, \phi, \psi, ...$ parameters
- $f(x, y; \theta)$ function of x and y with parameters θ, equivalently: $f_\theta(x, y)$

A Probabilistic Learner I

- Let's come up with a **supervised machine learning** method
- We build a probabilistic model of the training data D^{train},

$$P_\theta(x, y) = \prod_{i \in D^{train}} P_\theta(x^i, y^i)$$

- We learn our model parameters θ such that they maximize the data log likelihood
- We subsequently use that trained model to predict the class labels of the test data
- So, *given* a test instance $x \in D^{test}$, which class y is most likely?

$$\hat{y} = \underset{y \in Y}{\mathrm{argmax}}\, P(y|x)$$

THE UNIVERSITY OF MELBOURNE

A Probabilistic Learner II

The obvious way of doing this:

- For each class y:
  - Find the instances in the training data labelled as y
  - Count the number of times x has been observed
- Choose $\hat{y}$ with the greatest frequency of observed x

A Probabilistic Learner III

The obvious way of doing this:

- Would require an *enormous* amount of data
- A test instance x is a bundle of attribute values: to classify an (as-yet) unseen instance would require that *every possible* combination of attribute values has been attested in the training data a non-trivial number of times
- For m attributes, each taking k different values, and $|Y|$ classes, this means $\mathcal{O}(|Y| \cdot k^m)$ instances
    - Weather example: perhaps 100s of instances
    - 2-class problem, 20 binary attributes: at least 2M instances
    - 4 classes, 60 ternary attributes: at least 10^{28} instances
- Would only be meaningful for the instances that we've actually seen

Bayes' Rule

Reformulate the probability of class under features as probability of features under class

$$P(x, y) = P(y|x)P(x) = P(x|y)P(y)$$

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Bayes' Rule

Reformulate the probability of class under features as probability of features under class

$$P(x, y) = P(y|x)P(x) = P(x|y)P(y)$$

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Recall our objective

$$\hat{y} = \operatorname*{argmax}_{y \in Y} P(y|x)$$

Bayes' Rule

Reformulate the probability of class under features as probability of features under class

$$P(x, y) = P(y|x)P(x) = P(x|y)P(y)$$
$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Recall our objective

$$\begin{aligned}\hat{y} &= \operatorname*{argmax}_{y \in Y} P(y|x) \\ &= \operatorname*{argmax}_{y \in Y} \frac{P(x|y)P(y)}{P(x)} \\ &= \operatorname*{argmax}_{y \in Y} P(x|y)P(y)\end{aligned}$$

THE UNIVERSITY OF MELBOURNE

Bayes' Rule

Reformulate the probability of class under features as probability of features under class

$$P(x, y) = P(y|x)P(x) = P(x|y)P(y)$$
$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Recall our objective

$$\begin{aligned}\hat{y} &= \operatorname*{argmax}_{y \in Y} P(y|x) \\ &= \operatorname*{argmax}_{y \in Y} \frac{P(x|y)P(y)}{P(x)} \\ &= \operatorname*{argmax}_{y \in Y} P(x|y)P(y)\end{aligned}$$

Recall that each observation consists of many features $x = x_1, x_2, ...x_M$

$$\hat{y} = \operatorname*{argmax}_{y \in Y} P(x_1, x_2, ..., x_M|y)P(y)$$

That is still infeasible!

Putting ‘Naive’ in Naive Bayes

To arrive at a more feasible solution, we make a naive assumption

$$P(x_1, x_2, ..., x_M|y)P(y) \approx P(x_1|y)P(x_2|y)...P(x_M|y)P(y)$$
$$= P(y)\prod_{m=1}^{M} P(x_m|y)$$

- The **conditional independence assumption**: Conditioned on the class y, the features are assumed to be independent
- Intuitively: if I know that the class of the email is `spam`, none of the words depend on their surrounding words
- Clearly, this is nonsense. But the model works surprisingly well, anyway!

The Naive Bayes Model: Generative Story

The complete Naive Bayes Classifier

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} P(y) P(x_1, x_2, x_3, x_4, ...x_n | y)$$

$$= \underset{y \in Y}{\operatorname{argmax}} P(y) \prod_{m=1}^{M} P(x_m | y)$$

The Underlying Probabilistic Model

$$P(x, y) = \prod_{i=1}^{N} P(y^i) \prod_{m=1}^{M} P(x_m^i | y^i)$$

Intuition

Algorithm 1 Generative Story of Naive Bayes

1: **for** Observation $i \in \{1, 2, ...N\}$ **do**
2: Generate the label y^i from $P(y)$
3: **for** Feature $m \in \{1, 2, ...M\}$ **do**
4: Generate feature value x_m^i given that label=y^i from $P(x_m^i | y^i)$

Naive Bayes Assumptions

$$P(x, y) = \prod_{i=1}^{N} P(y^i) \prod_{m=1}^{M} P(x_m^i | y^i)$$

- Features of an instance are conditionally independent given the class
- Instances are independent of each other
- The distribution of data in the training instances is the same as the distribution of data in the test instances

Gaussian Naive Bayes with 2 classes

Observations	real-valued feature vectors of length M labelled with binary class (0,1)
Example	
Model	y drawn from Bernoulli distribution x_m drawn from Gaussian distribution

$$p(x, y) = p_{\phi,\psi}(x_1, x_2, \cdots, x_m, y) = p_\phi(y) \prod_m^M p_\psi(x_k|y)$$

$$= BN(y|\phi) \prod_m^M N(x_k|\psi = \{\mu_{m,y}, \sigma_{m,y}\})$$

$$= \phi^y (1-\phi)^{(1-y)} \prod_{m=1}^M \frac{1}{\sqrt{2\pi\sigma^2_{m,y}}} exp\Big(-\frac{1}{2} \frac{(x_m - \mu_{m,y})^2}{\sigma^2_{m,y}} \Big)$$

Prediction	$\hat{y} = argmax_y p(y|x)$

Bernoulli Naive Bayes with 2 classes

Observations	binary feature vectors of length M labelled with binary class (0,1)
Example	
Model	y drawn from Bernoulli distribution x_m drawn from Bernoulli distribution

$$p(x,y) = p_{\phi,\psi}(x_1, x_2, \cdots, x_m, y) = p_\phi(y)\prod_m^M p_\psi(x_k|y)$$

$$= BN(y|\phi)\prod_m^M BN(x_k|\psi_{m,y})$$

$$= \phi^y(1-\phi)^{1-y}\prod_{m=1}^M(\psi_{y,m})^{x_m}(1-\psi_{y,m})^{(1-x_m)}$$

Prediction	$\hat{y} = argmax_y p(y\|x)$

Categorical Naive Bayes with C classes

Observations	categorical feature vectors of length M labelled with one of C classes ($C > 2$)
Example	
Model	y drawn from Categorical distribution with C classes x_m drawn from Categorical distribution over K values $$p(x,y) = p_{\phi,\psi}(x_1, x_2, \cdots, x_m, y) = p_\phi(y) \prod_m^M p_\psi(x_k \mid y)$$ $$= Cat(y \mid \phi) \prod_m^M Cat(x_k \mid \psi_{m,y})$$ $$= \phi_y \prod_{m=1}^M \prod_{k=1}^K (\psi_{y,m,k})$$
Prediction	$\hat{y} = argmax_y p(y \mid x)$

Maximum Likelihood Estimation for Categorical Naive Bayes

But where do the parameters come from?

- Parameters ϕ of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{count(y)}{N}$$

- Parameters ψ of the**Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{count(y, m)}{count(y)}$$

Maximum Likelihood Estimation for Categorical Naive Bayes

But where do the parameters come from?

- Parameters ϕ of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{count(y)}{N}$$

- Parameters ψ of the**Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{count(y,m)}{count(y)}$$

- These parameters maximize the probability of the observed dataset $P(\{(x^i, y^i)\}_{i=1}^N; \phi, \psi)$. They are the **maximum likelihood estimate** of ϕ and ψ.
- You are invited to derive this result using the optimization techniques we learnt in the last lecture!

Maximum Likelihood Estimation for Categorical Naive Bayes

But where do the parameters come from?

- Parameters ϕ of the **Categorical distribution over class labels** are the relative frequencies of classes observed in the training data

$$\phi_y = \frac{count(y)}{N}$$

- Parameters ψ of the**Categorical distributions over features given a class label** are the observed relative frequencies of (class, label) among all instances with that class

$$\psi_{y,m} = \frac{count(y, m)}{count(y)}$$

- The
$P(\{$
and ψ.

> **Activity**: What are the four steps we would follow in finding the optimal parameters?

- You are invited to derive this result using the optimization techniques we learnt in the last lecture!

Maximum Likelihood Estimation for Gaussian Naive Bayes

For each class y and each feature x_m, we learn an individual Gaussian distribution parameterized by a mean $\mu_{y,m}$ and a standard deviation $\sigma_{y,m}$

Mean: the average of all observed feature value for x_m under class y

$$\mu_{y,m} = \frac{1}{count(y)} \sum_{i:y_i=y} x_m^i$$

Standard deviation: Sum of squared differences of observed values from the mean. Normalized, and square rooted.

$$\sigma_{y,m} = \sqrt{\frac{\sum_{i:y_i=y}(x_m^i - \mu_{y,m})^2}{count(y)}}$$

Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

We need $P(y = k)$, $P(x = f|y = k)$, for every possible value k for y and every possible value f for x

Naive Bayes Example I

Given a training data set, what probabilities do we need to estimate?

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

We need $P(y = k)$, $P(x = f|y = k)$, for every possible value k for y and every possible value f for x

$$\begin{aligned}
P(Flu) &= 3/5 \\
P(Headache = severe|Flu) &= 2/3 \\
P(Headache = mild|Flu) &= 1/3 \\
P(Headache = no|Flu) &= 0/3 \\
P(Sore = severe|Flu) &= 1/3 \\
P(Sore = mild|Flu) &= 2/3 \\
P(Sore = no|Flu) &= 0/3 \\
P(Temp = high|Flu) &= 1/3 \\
P(Temp = normal|Flu) &= 2/3 \\
P(Cough = yes|Flu) &= 3/3 \\
P(Cough = no|Flu) &= 0/3
\end{aligned}$$

$$\begin{aligned}
P(Cold) &= 2/5 \\
P(Headache = severe|Cold) &= 0/2 \\
P(Headache = mild|Cold) &= 1/2 \\
P(Headache = no|Cold) &= 1/2 \\
P(Sore = severe|Cold) &= 1/2 \\
P(Sore = mild|Cold) &= 0/2 \\
P(Sore = no|Cold) &= 1/2 \\
P(Temp = high|Cold) &= 0/2 \\
P(Temp = normal|Cold) &= 2/2 \\
P(Cough = yes|Cold) &= 1/2 \\
P(Cough = no|Cold) &= 1/2
\end{aligned}$$

Naive Bayes Example II

Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a *Cold*, or the *Flu*?

Naive Bayes Example II

Ann comes to the clinic with a mild headache, severe soreness, normal temperature and no cough. Is she more likely to have a *Cold*, or the *Flu*?

Cold:

$$\begin{array}{ccc} P(Co) & \times & P(H=m|Co)P(S=s|Co)P(T=n|Co)P(C=n|Co) \\ \frac{2}{5} & \times & (\frac{1}{2})(\frac{1}{2})(\frac{2}{2})(\frac{1}{2}) = 0.05 \end{array}$$

Flu:

$$\begin{array}{ccc} P(Fl) & \times & P(H=m|Fl)P(S=s|Fl)P(T=n|Fl)P(C=n|Fl) \\ \frac{3}{5} & \times & (\frac{1}{3})(\frac{1}{3})(\frac{2}{3})(\frac{0}{3}) = 0 \end{array}$$

Naive Bayes Example III

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccl} P(Co) & \times & P(H=s|Co)P(S=m|Co)P(T=h|Co)P(C=n|Co) \\ \frac{2}{5} & \times & (\frac{0}{2})(\frac{0}{2})(\frac{0}{2})(\frac{1}{2}) = 0 \end{array}$$

Flu:

$$\begin{array}{ccl} P(Fl) & \times & P(H=s|Fl)P(S=m|Fl)P(T=h|Fl)P(C=n|Fl) \\ \frac{3}{5} & \times & (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\frac{0}{3}) = 0 \end{array}$$

Smoothing: Introduction

The problem with unseen features

- If any term $P(x_m|y) = 0$ then the class probability $P(y|x) = 0$
- But, we already established that in any realistic scenario we won't see every class-feature combination during training
- A single zero renders many additional meaningful observations irrelevant
- **Solution:** no event is impossible: $P(x_m|y) > 0 \forall x_m \forall y$
- We need to readjust the remaining model parameters to maintain valid probability distributions ($\sum_i \psi_i = 1$)

Epsilon Smoothing

Simplest approach

- if we calculate $P(x_m|y) = 0$, we replace 0 with a very (!) small constant typically called ϵ
- ϵ needs to be smaller (preferably much smaller) than $\frac{1}{N}$ (N=the number of training instances). **Why?**
- Effectively it reduces most comparisons to the cardinality of ϵ (fewest ϵs wins)
- We assume that ϵ is so small that $1 + \epsilon \approx 1$, so we do not need to renormalize or adjust the other probabilities in the model

Epsilon Smoothing

Bob comes to the clinic with a severe headache, mild soreness, high temperature and no cough. Is he more likely to have a cold, or the flu?

Cold:

$$\begin{array}{ccc} P(Co) & \times & P(H=s|Co)P(S=m|Co)P(T=h|Co)P(C=n|Co) \\ \frac{2}{5} & \times & (\epsilon)(\epsilon)(\epsilon)(\frac{1}{2}) = \frac{\epsilon^3}{5} \end{array}$$

Flu:

$$\begin{array}{ccc} P(Fl) & \times & P(H=s|Fl)P(S=m|Fl)P(T=h|Fl)P(C=n|Fl) \\ \frac{3}{5} & \times & (\frac{2}{3})(\frac{2}{3})(\frac{1}{3})(\epsilon) = \frac{12\epsilon}{135} = \frac{4\epsilon}{45} \end{array}$$

Laplace Smoothing

Add a "pseudocount" α to each feature count observed during training

$$P(x_m = j | y = k) = \frac{\alpha + count(y = k, x_m = j)}{M\alpha + count(y = k)}$$

- the value of α is a parameter; very often $\alpha = 1$
- all **counts** are incremented to ensure to maintain monotonicity (for $\alpha = 1$: 0 becomes 1, 1 becomes 2, 2 becomes 3, ...)
- M is the number of values x_m can take on

Laplace Smoothing

Add a "pseudocount" α to each feature count observed during training

$$P(x_m = j | y = k) = \frac{\alpha + count(y = k, x_m = j)}{M\alpha + count(y = k)}$$

Example

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

	original estimate	smoothed estimate ($\alpha = 1$)
$P(Headache = severe \| Flu)$	$2/3$	$(2+1)/(3+3) = 3/6$
$P(Headache = mild \| Flu)$	$1/3$	$(1+1)/(3+3) = 2/6$
$P(Headache = no \| Flu)$	$0/3$	$(0+1)/(3+3) = 1/6$
$P(Cough = yes \| Flu)$	$3/3$	$(3+1)/(3+2) = 4/5$
$P(Cough = no \| Flu)$	$0/3$	$(0+1)/(3+2) = 1/5$
...		

Laplace Smoothing

Add a "pseudocount" α to each feature count observed during training

$$P(x_m = j | y = k) = \frac{\alpha + count(y = k, x_m = j)}{M\alpha + count(y = k)}$$

- Probabilities are changed drastically when there are few instances; with a large number of instances, the changes are small
- Laplace smoothing (and smoothing in general) **reduces variance** of the NB classifier because it reduces sensitivity to individual (non-)observations in the training data
- Laplace smoothing (and smoothing in general) **adds bias** to the NB classifier. We no longer have a true maximum likelihood estimator.
- How to choose α?
- There are other smoothing methods, including Good-Turing, Kneser-Ney, Regression, ... (outside the scope of this class)

Implementation of Categorical Naive Bayes

Implementing a Naive Bayes Classifier

Naive Bayes is a supervised machine learning method:

- We need to build a model (“training phase”)
- We need to make predictions using that model and evaluate the predictions against the ground truth (“testing phase”)

Training a NB Classifier I

Our model consists of two kinds of probabilities:

- *priors* $P(Y = k)$ (one per class)
- *likelihoods* $P(X = j|Y = k)$ (one per attribute value, per class)

Calculating priors by counting I

There is one prior $P(Y = k)$ per class

X_1 (Headache)	X_2 (Sore)	X_3 (Temperature)	X_4 (Cough)	Y (Diagnosis)
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
2	3

Calculating priors by counting I

There is one prior $P(Y = k)$ per class

X_1 (Headache)	X_2 (Sore)	X_3 (Temperature)	X_4 (Cough)	Y (Diagnosis)
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Cold	Flu
2	3

We need to normalize these counts by the total number of training instances N. Options:

- divide each entry by the sum of the entries in the list
- keep a separate counter for the total number of instances N, which is often useful

Calculating likelihoods by counting I

There is one likelihood $P(x = j|y = k)$ per attribute value, per class: 2D array?

Calculating likelihoods by counting I

There is one likelihood $P(x = j|y = k)$ per attribute value, per class, **for each attribute** X: ~~2D array?~~ 3D array?

- But each attribute might have a different number of possible attribute values,
- And we might not know all of the various attribute values before we start counting.
- So...
  - 2D array of dictionaries?
  - 1D array of dictionaries of dictionaries?
  - Dictionary of dictionaries of dictionaries?

THE UNIVERSITY OF MELBOURNE

Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache

Cold:	{}
Flu:	{severe:1}

Temperature

Cold:	{}
Flu:	{}

Sore

Cold:	{}
Flu:	{}

Cough

Cold:	{}
Flu:	{}

Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	**mild**	high	yes	**Flu**
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	yes	Flu

Headache			Temperature	
Cold:	{}		Cold:	{}
Flu:	{severe:1}		Flu:	{}

Sore			Cough	
Cold:	{}		Cold:	{}
Flu:	{mild:1}		Flu:	{}

Calculating likelihoods by counting II

Headache	Sore	Temperature	Cough	Diagnosis
severe	mild	high	yes	Flu
no	severe	normal	yes	Cold
mild	mild	normal	yes	Flu
mild	no	normal	no	Cold
severe	severe	normal	**yes**	**Flu**

Headache

Cold:	{no:1, mild:1}
Flu:	{severe:2, mild:1}

Temperature

Cold:	{normal:2}
Flu:	{high:1, normal:2}

Sore

Cold:	{severe:1, no:1}
Flu:	{mild:2, severe:1}

Cough

Cold:	{yes:1, no:1}
Flu:	{yes:3}

Calculating likelihoods by counting II

We need to know the number of instances of class c_j to turn these counts into probabilities:

- The slow way: sum the entries in the corresponding dictionary
- The fast way: read off the class array

Smoothing can be done:

- When accessing values, e.g. if value is 0, replace with ϵ
- Using a `defaultdict`, e.g. default for Laplace is 1

Making predictions using a NB Classifier i

$$\hat{y} = \underset{k \in Y}{\operatorname{argmax}} P(y = k) \prod_{m} P(x_m = j | y = k)$$

- These values can be read off the data structures from the training phase.
- We only care about the class corresponding to the maximal value, so as we progress through the classes, we can keep track of the greatest value so far.

Making predictions using a NB Classifier ii

We're multiplying a bunch of numbers $(0, 1]$ together — because of our floating-point number representation, we tend to get **underflow**.

One common solution is a **log-transformation**:

$$\begin{aligned}\hat{y} &= \underset{k \in Y}{\operatorname{argmax}}\, P(y = k) \prod_{m} P(x_m = j | y = k) \\ &= \underset{k \in Y}{\operatorname{argmax}} \left[\log\ P(y = k) + \sum_{m} \log\ P(x_m = j | y = k) \right]\end{aligned}$$

Evaluating a NB classifier

Evaluation in a supervised ML context (for NB and other methods):

- fundamentally based around comparing predicted labels with the actual labels

We'll talk about this in much more detail in the upcoming lectures.

Naive Bayes: Final thoughts

Why does it work given that it's a blatantly wrong model of the data?

- we don't need the true distribution over $P(y|x)$, we just need to be able to identify the most likely outcome

Advantages of Naive Bayes

- easy to build and estimate
- easy to scale to many feature dimensions (e.g., words in the vocabulary) and data sizes
- reasonably easy to explain why a specific class was predicted
- good starting point for a classification project

Summary

Naive Bayes

- What is the Naive Bayes algorithm?
- What is Bayes' Rule and how does it relate to the Naive Bayes algorithm
- What are the simplifying assumptions we make?
- How and why do we use smoothing in Naive Bayes?
- How can we implement a Naive Bayes classifier?

Next Lecture: Evaluation

References

Jacob Eisenstein. *Natural Language Processing*. MIT Press (2019). Chapter 2.2

MLE of Categorical Naive Bayes – for the Math hungry

- The **likelihood** is the probability of the data as a function of only the parameters:

$$\mathcal{L}(\phi, \psi) = \log \ Cat(y^i|\phi) + \sum_i^N \log \ Cat(x^i; \psi_{y^i})$$

- Focussing only on terms involving ψ (the same steps can be applied to ϕ, separately)

$$\begin{aligned}\mathcal{L}(\psi) &= \sum_i^N \log \ Cat(x^i; \psi_{y^i}) \\ &= \sum_i^N \sum_{f=1}^V x_f \times \log \psi_{y^i,f}\end{aligned}$$

- now choose ψ to maximize $\mathcal{L}$ under the constraint that

$$\sum_{f=1}^V \psi_{y,f} = 1 \quad \forall y,$$

(i.e., all possible outcomes add up to 1)

MLE of Categorical Naive Bayes – for the Math hungry

- integrate the constraint by adding a set of Lagrange multipliers

$$\ell(\psi_y) = \sum_{i:y^i=y} \sum_{f=1}^{V} x_f \times \log \psi_{y,f} - \lambda\Big(\sum_{f=1}^{V} \psi_{y,f} - 1\Big)$$

- we differentiate the likelihood wrt. $\psi_{y,f}$ i.e., the probability of feature value f under class y

$$\frac{\partial \ell(\psi_y)}{\partial \psi_{y,f}} = \sum_{i:y^i=y} x_f/\psi_{y,f} - \lambda$$

- set the derivatives to zero, and rearrange

$$\lambda\psi_{y,f} = \sum_{i:y^i=y} x_f$$

$$\psi_{y,f} \propto \sum_{i:y^i=y} x_f \quad = \quad \text{count}(y,f)$$

- and the only way to find an exact solution that obeys our sum-to-one constraint is

$$\psi_{y,f} = \frac{count(y,f)}{\sum_{f' \in V} count(y,f')} = \frac{count(y,f)}{count(y)}$$

...which is the exact quantity we defined on slide 14. Hurray!

MLE of Categorical Naive Bayes – for the Math hungry

Following an almost identical procedure we can derive that

$$\phi_y = \frac{count(y)}{N}$$