8.1

In this course, so far, we only discussed classifiers, which work on non-sequential data (or alternatively also data with a fixed sequence).

a) Why might those algorithms have trouble predicting the correct output for sequential data (like time-series, text, etc.)?

Traditional classifiers, like decision trees or support vector machines, are designed to handle data that doesn't depend on previous values. These algorithms treat each piece of data as an independent entity, which becomes problematic when working with sequences like text or time-series data. For example, in predicting stock prices, a classifier wouldn't be able to recognize that yesterday's price directly influences today's prediction, since it doesn't account for the temporal relationship between data points.

b) How do Recurrent Neural Networks (RNNs) help with sequential data?

Recurrent Neural Networks (RNNs) are specifically designed for sequential data. Unlike traditional neural networks, RNNs have loops that allow them to remember previous inputs and use that memory to inform future predictions. This makes them ideal for tasks like text analysis or time-series forecasting, where the order and context of data points are crucial to making accurate predictions.

c) What is Backpropagation Through Time, and what are the issues with gradients?

Backpropagation Through Time is applied to train RNN networks which can work on sequential data like stream data, and time series data.

- It works by unrolling all input timesteps.
- Each timestep has one input timestep, one copy of the network, and one output.
- Errors are calculated and collected for each timestep.
- The network is rolled back up all the weights are updated and the process is repeated.

The problem of vanishing and exploding gradient: Backpropagation Through time can be computationally expensive as the number of timesteps increases. In turn, if the input sequences consist of thousands of timesteps, then this much amount of derivatives must be required for a single weight update. This can cause the weights to vanish or explode (go to zero or over-shoot) leading to a slow learning process.

d) How do LSTMs address the vanishing gradient problem?

LSTM has a Forget gate in its architecture which helps in processing long input sequences, this was a major drawback in RNN where while processing longer input sequences, they would encounter vanishing gradient problems. Also, they possess a unique additive gradient structure that includes direct access to the forget gate's activations, enabling the network to encourage desired behavior from the error gradient using frequent gate updates at each time step of the learning process.

e) Summary in bullet points:

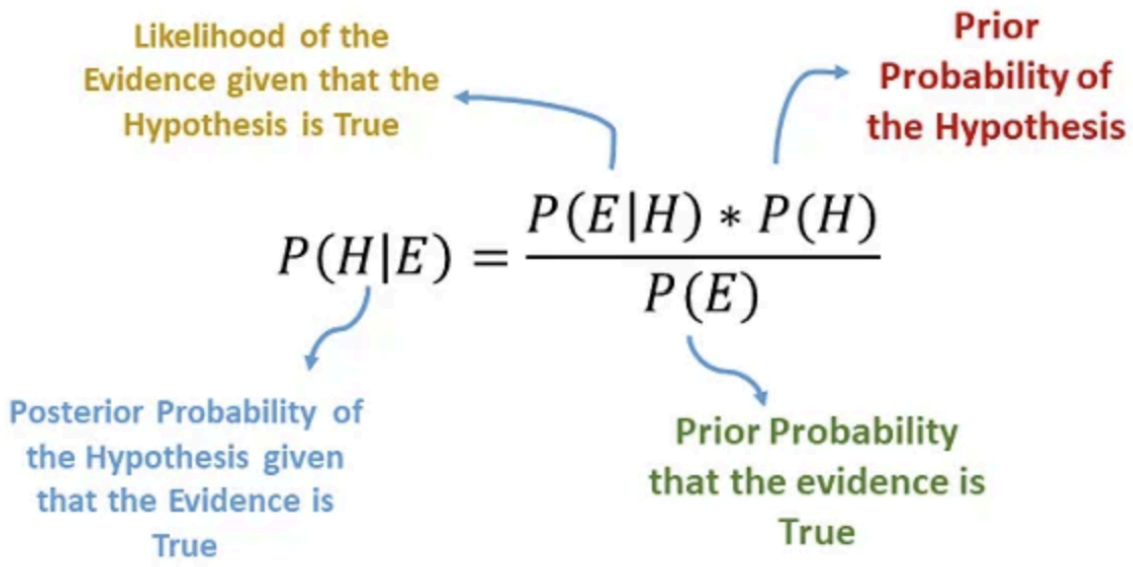
- For sequential data, RNNs should be used and a classifier built for non-sequential data may not generalize all problems but needs to work on sequential data as well.
- RNNs can process long sequences due to the presence of a feedback loop.
- RNNs should be trained using the BPTT algorithm.
- LSTM can solve the vanishing gradient problem by replacing the multiplicative part with addition.

8.2

Explain in your own words the functionality of a Naive Bayes classifier. Cover the following points in your discussion:

a) What are the formulas used in Naive Bayes and how are they related to a Maximum-A-Posteriori hypothesis?

Naive Bayes classifiers are based on Bayes Theorem, which describes the probability of an event happening given prior knowledge of related events. The formula for Naive Bayes is:



P(H | E) is the posterior probability of the class H given the feature set E.
P(E | H) is the likelihood of the feature set E given the class H.
P(H) is the prior probability of class H.
P(E) is the probability of the feature set E.

In Naive Bayes, we use the **Maximum A-Posteriori (MAP)** hypothesis, which essentially seeks to maximize the posterior probability P(H | E). To simplify calculations, Naive Bayes assumes that the features are conditionally independent given the class, which allows us to express the likelihood P(E | H) as the product of individual feature likelihoods:

In many cases, some set of candidate hypotheses H are considered by the learner who is interested in finding the most probable hypothesis h belongs to H given the observed data E. Any such maximally probable hypothesis is called a Maximum-A-Posteriori (MAP) hypothesis. To calculate this hypothesis, Bayes theorem can be used.

P(E) is a const & independent of h.

b) What is the “naive” assumption made here and why is it relevant?

The naive assumption in Naive Bayes is that all features are conditionally independent given the class label. This means that the classifier assumes that knowing the value of one feature does not provide any additional information about the other features, once the class is known.
Relevance: Relying on the exact duplicates in the training set is of no use in classification.

c) How are the probabilities in a Naive Bayes estimated and used for Classification?

In Naive Bayes, probabilities are estimated from the training data. Here's how:
Prior Probability: The prior probability P(C) is estimated by counting how often each class appears in the training data.

P(C)= Number of instances of class C / Total number of instances

Likelihood: The likelihood $P(x_i|C)$ for each feature x_i is estimated by counting how often each feature value appears for instances of class C and dividing by the total number of instances in class C.

$P(x_i|C)$=Number of instances where feature x_i appears in class C/ Total instances in class C

Once these probabilities are estimated, Naive Bayes uses them to classify new instances. For a new data point, the classifier calculates the posterior probability $P(C|X)$) for each class using the formula derived from Bayes Theorem, and assigns the class with the highest posterior probability.

d) How can we deal with missing values in a data set and zero frequency problems? (These are two different problems!)

Missing Values: When a feature value is missing for a new instance, we can handle this by estimating the probability of that feature value being observed given the class. One common approach is to ignore the missing feature during the classification or impute the missing values by using the mean, median, or mode of the feature values across the training dataset.

Zero Frequency Problem: If a particular feature value never appears in the training data for a given class, its likelihood $P(x_i|C)$ would be zero, which would lead to a zero posterior probability for that class. To address this, we use Laplace smoothing, where a small constant (usually 1) is added to the numerator and the number of unique feature values to the denominator:

$P(x_i|C)$=Number of instances where feature x_i appears in class C+1 / (Total instances in class C + Number of unique feature values)

This ensures that no probability is ever zero, even if a feature value does not appear for a particular class.

8.3

Assignment 8.3

Consider the following PlayTennis example. The given training data is:

day	outlook	temperature	humidity	wind	play tennis
1	rain	mild	high	strong	no
2	overcast	hot	normal	weak	yes
3	overcast	mild	high	strong	yes
4	sunny	mild	normal	strong	yes
5	rain	mild	normal	weak	yes
6	sunny	cool	normal	weak	yes
7	sunny	mild	high	weak	no
8	overcast	cool	normal	strong	yes
9	rain	cool	normal	strong	no
10	rain	cool	normal	weak	yes

a) Classify the following two examples using Naive Bayes and a maximum likelihood estimation:

$$i_1 = (sunny, cool, high, strong)$$

$$i_2 = (overcast, mild, normal, weak)$$

Compute also the posteriori probabilities.

b) Consider the PlayTennis example from the previous subtask again. This time, create a Naive Bayes classifier using the formula on slide 31, with $m = 1$ and assuming a uniform distribution of values. What differences do you observe, when classifying the instances from subtask a)?

7.3 $P(yes) = 7/10 \quad P(no) = 3/10$

$P(i_1) = 3/10 \cdot 4/10 \cdot 3/10 \cdot 5/10 = 0.018$

$P(i_1|yes) = P(Sunny|yes)\,P(cool|yes)\,P(high|yes)\,P(strong|yes)$
$= 2/7 \cdot 3/7 \cdot 1/7 \cdot 3/7 = 7.4968 \times 10^{-3}$

yes class

Prob	P(sun/yes)	P(cool|yes)	P(high|yes)	P(strong|yes)
Values	2/7	3/7	1/7	3/7

no class

P	P(s|no)	P(c|no)	P(h|no)	P(s|yes)
V	1/3	1/3	2/3	2/3

Bayes Theorem

$P(yes|i_1) = \frac{P(i_1|yes)\,P(yes)}{P(i_1)} = \frac{7.4968 \times 10^{-3} \times 7/10}{0.018} = 0.29$

$P(no|i_1) = 1 - 0.29 = 0.71$

$P(i_1|no) = P(Sunny|no)\,P(cool|no)\,P(high|no)\,P(strong|no)$
$= 1/3 \cdot 1/3 \cdot 2/3 \cdot 2/3 = 0.049$

Using Bayes theorem

$P(no|i_1) = \frac{P(i_1|no)\,P(no)}{P(i_1)} = \frac{0.049 \times 3/10}{0.018} = 0.82$

$V_{NB}^{yes} = \frac{0.29}{0.29 + 0.82} = 0.26 \quad V_{VB}^{no} = \frac{0.82}{0.29 + 0.82} = 0.73$

$\Rightarrow V_{VB}^{no} > V_{NB}^{yes}$, i_1 is classified as no

Using Maximum likelihood Estimate

Let $h_1 = yes$, $h_2 = no$

$P(i_1|yes) \times P(yes) = 2/7\ 3/7\ 1/7\ 3/7\ 7/10 = 5.24 \times 10^{-3}$

$P(i_1|no) \times P(no) = 1/3\ 1/3\ 2/3\ 2/3\ 3/10 = 0.0148$

$h_{ML} = \underset{h_1 h_2 \in H}{\text{argmax}}\ P(i_1|yes).P(yes).P(i_1|no).P(no)$

$= P(i_1|no).P(no)$

i_1 is classified as no

Using Naive Bayes

Considering i_2

Probability	P(o|yes)	P(m|yes)	P(r|yes)	P(w|yes)
Value	3/7	3/7	6/7	4/7
P	P(o|no)	P(m|no)	P(r|no)	P(w|no)
V	0/3	2/3	1/3	1/3

$P(i_2) = 3/10 \cdot 5/10 \quad 7/10 \cdot 5/10 = 0.0525$

$P(i_2|yes) = 3/7 \cdot 3/7 \cdot 6/7 \cdot 4/7 = 0.09$

Using Bayes theorem

$P(yes|i_2) = \frac{P(i_2|yes)P(yes)}{P(i_2)} = \frac{0.09 \times 7/10}{0.0525} = 1.2$

n_c	(S|y)	Cool|yes	high|yes	strong|yes
Count	2	3	1	3
n_c	Sunny|no	c|n	h|n	S|n
Count	1	1	2	2

$n_{yes} = 7 \qquad n_{no} = 3$

for i_1,

$P_{outlook} = 1/3 \quad P_{temp} = 1/3 \quad P_{humidity} = 1/2 \quad P_{wind} = 7$

$\Rightarrow P(Sunny|yes) = \frac{2 + (1 \cdot 1/3)}{7+1} = 7/24$

$P(cool|yes) = \frac{3 + (1 \cdot 1/3)}{7+1} = 5/12$

$P(high|yes) = \frac{1 + (1 \cdot 1/2)}{7+1} = 3/16$

BRUNNEN

$P(\text{strong}/\text{yes}) = \frac{3+(1\times 1/2)}{7+1} = 7/16$

$P(\text{yes}|i_1) = P(s/y)\,P(c/y)\cdot P(h/y)\,P(s/y)\,P(y)$
$= 7/14 \;\; 5/12 \;\; 3/16 \;\; 7/16 \;\; 7/10 = 6.9\times 10^{-3}$

$P(s|no) = \frac{1+(1\cdot 1/3)}{3+1} = 1/3$

$P(h/no) = \frac{2+(1\cdot 1/2)}{3+1} = 5/8$

$P(c|no) = \frac{1+(1\cdot 1/3)}{3+1} = 1/3$

$P(s|no) = \frac{2+(1\cdot 1/2)}{3+1} = 5/8$

$P(no|i_1) = P(s|no)\,P(c|no)\,P(h|no)\,P(s|no)\,P(no)$
$= 1/3\cdot 1/3\cdot 5/8\cdot 5/8\cdot 3/10 = 0.01$

$\therefore P(no|i_1) > P(yes|i_1)$, i_1 is classified as no

for i_2

Nc	Overcast/yes	mild/yes	normal/yes	weak/yes
Count	3	3	6	6

Nc	Overcast/no	m/no	n/no	w/no
Count	0	2	1	1

$P_{overcast} = 1/3$ $P_{mild} = 1/3$ $P_{normal} = 1/2$ $P_{weak} = 1/2$

$N_{yes} = 7$ $N_{no} = 3$

$P(\text{Overcast}/\text{yes}) = \frac{3+(1\cdot 1/3)}{7+1} = 5/12$

$P(\text{mild}/\text{yes}) = \frac{3+(1\cdot 1/3)}{7+1} = 5/12$

$$P(\text{normal}|\text{yes}) = \frac{6+(1\cdot 1/2)}{7+1} = 13/16$$

$$P(\text{weak}|\text{yes}) = \frac{4+(1\cdot 1/2)}{7+1} = 9/16$$

$$P(\text{yes}|i_2) = P(o|y)\,P(m|y)\,P(r|y)\,P(w|y)\,P(y)$$
$$= 5/12 \quad 5/12 \quad 13/16 \quad 9/16 \quad 7/10 = 0.055$$

11 by $P(o|no) = \frac{0+(1\cdot 1/3)}{3+1} = 1/12$ $P(m|no) = \frac{2+(1\cdot 1/3)}{3+1} = 7/12$

$P(r|no) = \frac{1+(1\cdot 1/2)}{3+1} = 1/3$ $P(w|no) = \frac{1+(1\cdot 1/2)}{3+1} = 1/3$

$$P(no|i_2) = P(o|no)\,P(m|no).P(r|no)P(w|no)\,P(no)$$
$$= 1/12 \cdot 7/12 \cdot 1/3 \cdot 1/3 \cdot 1/3 \cdot 3/10 = 1.6 \times 10^{-3}$$

$P(yes|i_2) > P(no|i_2)$, i_2 is classified as yes

8.4

Consider using Naive Bayes for classifying e-mails into spam and ham (not-spam). Very shortly discuss the following three things:
a) How can the probability of a word belonging to spam or not-spam be estimated?

c) What problems might occur, when using a regular Naive Bayes classification routine?

7th a Input to Naive Bayes classified be bag of of words, w such that $w_1, w_2, w_3 \dots w_n \in W$ which can process into List, L comprising words, w

$P(\text{not spam} \mid L[w])$ = Prob of that email is not spam given list of words

$$= \frac{P[L[w] \mid \text{not spam})\, P(\text{not spam})}{P[L[w]]}$$

$$P(\text{spam} \mid L[w]) = \frac{P(L[w] \mid \text{spam})\, P(\text{spam})}{P(L[w])}$$

from training data

$$P(\text{not spam}) = \frac{\text{no. of documents belonging to not spam}}{\text{Total no. of documents}}$$

$$P(\text{spam}) = \frac{\text{no. of documents belonging to spam}}{\text{Total no. of documents}}$$

from frequency table,

$P(L[w] \mid \text{not spam}) = P(w_1 \mid \text{not spam}), \dots P(w_n \mid \text{not spam})$

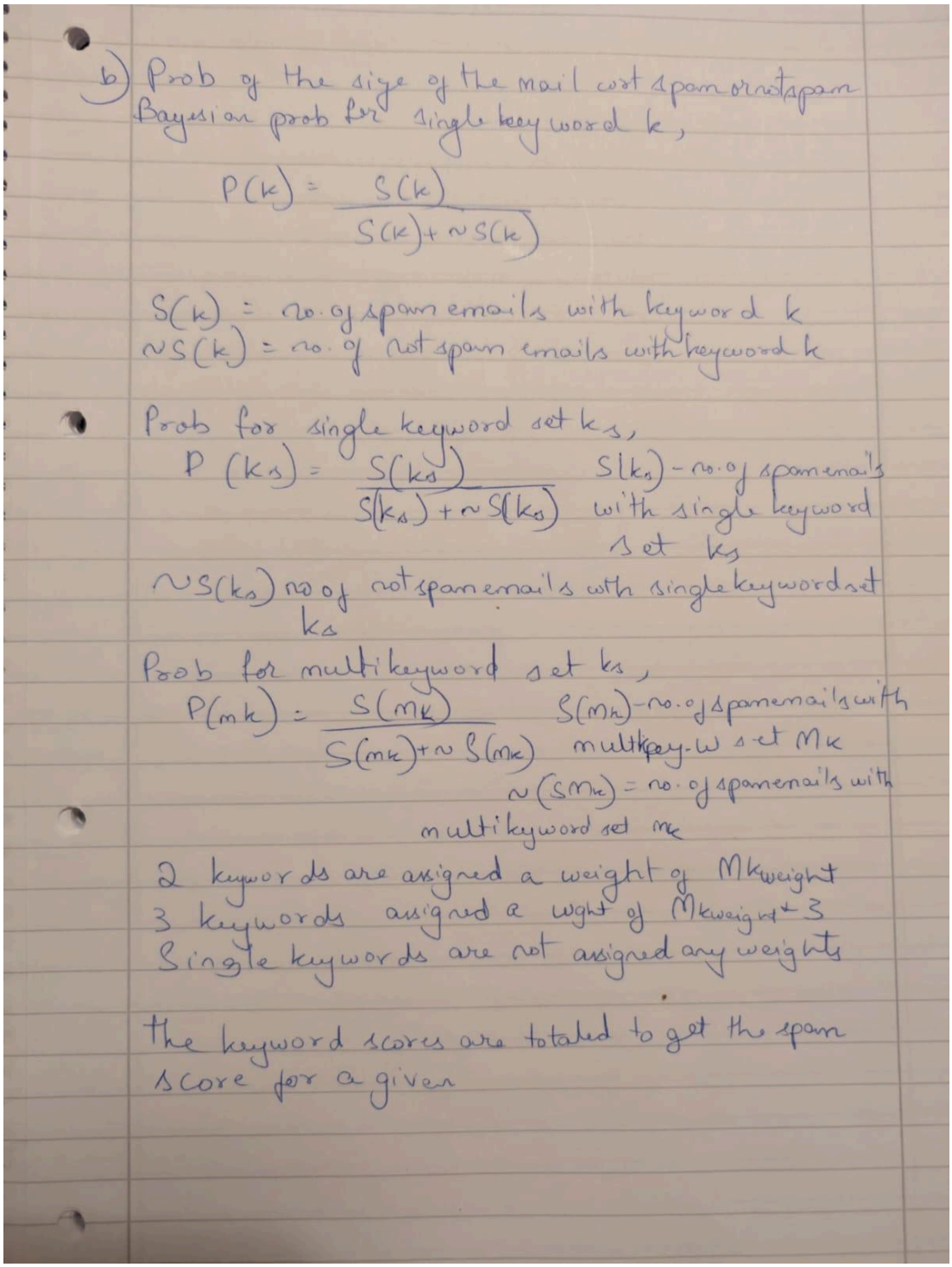
$P(L[w] \mid \text{spam}) = P(w_1 \mid \text{spam}), \dots P(w_n \mid \text{spam})$

$$P(w_1 \mid \text{spam}) = \frac{\text{Count of word 1 in spam category}}{\text{Total no. of words in spam category}}$$

$$P(w_1 \mid \text{not spam}) = \frac{\text{Count of word 1 in not spam category}}{\text{Total no. of words in not spam cat.}}$$

$P(L[w])$ is const

b) How can the probability of the size of the mail in regards to spam or not-spam be calculated?



b) Prob of the size of the mail wrt spam or not spam
Bayesian prob for single keyword k,

$$P(k) = \frac{S(k)}{S(k) + \sim S(k)}$$

$S(k)$ = no. of spam emails with keyword k
$\sim S(k)$ = no. of not spam emails with keyword k

Prob for single keyword set k_s,

$$P(k_s) = \frac{S(k_s)}{S(k_s) + \sim S(k_s)}$$

$S(k_s)$ - no. of spam emails with single keyword set k_s

$\sim S(k_s)$ no of k_s not spam emails with single keyword set

Prob for multikeyword set k_s,

$$P(mk) = \frac{S(m_k)}{S(m_k) + \sim S(m_k)}$$

$S(m_k)$ - no. of spam emails with multikey-w set M_k

$\sim (SM_k)$ = no. of spam emails with multikeyword set m_k

2 keywords are assigned a weight of $M_{kweight}$
3 keywords assigned a wght of $M_{kweight} * 3$
Single keywords are not assigned any weights

the keyword scores are totaled to get the spam score for a given

c) What problems might occur, when using a regular Naive Bayes classification routine?
Problems when using regular Naive Bayes classifier.

There is the possibility that our classifier detects a new word that is not present in the training data. In that case it multiplicative prob will be equal to zero. To overcome this, we use the Laplace estimate.

P(W/spam) = count of words belonging to spam + 1 / Total count of words belonging to spam + No.of distinct words in training data.