Bayesian Classifier

- $f: X \rightarrow V$, finite set of values
- Instances $x \in X$ can be described as a collection of features

$$x = (x_1, x_2, \ldots x_n) \quad x_i \in \{0,1\}$$

- Given an example, assign it the most probable value in V
- Bayes Rule:

$$v_{MAP} = \text{argmax}_{v_j \in V} P(v_j \mid x) = \text{argmax}_{v_j \in V} P(v_j \mid x_1, x_2, \ldots, x_n)$$

$$v_{MAP} = \text{argmax}_{v_j \in V} \frac{P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \text{argmax}_{v_j \in V} P(x_1, x_2, \ldots, x_n \mid v_j) P(v_j)$$

- Notational convention: P(y) means P(Y=y)

Bayesian Classifier

$$V_{MAP} = \text{argmax}_v \; P(x_1, x_2, \ldots, x_n \mid v)P(v)$$

- Given training data we can estimate the two terms.
- Estimating $P(v)$ is easy. E.g., under the binomial distribution assumption, count the number of times v appears in the training data.
- However, it is not feasible to estimate $P(x_1, x_2, \ldots, x_n \mid v)$
- In this case we have to estimate, for each target value, the probability of each instance (most of which will not occur).
- In order to use a Bayesian classifiers in practice, we need to make assumptions that will allow us to estimate these quantities.

Naive Bayes

$V_{MAP} = \text{argmax}_v \; P(x_1, x_2, \ldots, x_n \mid v)P(v)$

$$
\begin{aligned}
&\mathbf{P(x_1, x_2, \ldots, x_n \mid v_j) =} \\
&\quad = \mathbf{P(x_1 \mid x_2, \ldots, x_n, v_j) P(x_2, \ldots, x_n \mid v_j)} \\
&\quad = \mathbf{P(x_1 \mid x_2, \ldots, x_n, v_j) P(x_2 \mid x_3, \ldots, x_n, v_j) P(x_3, \ldots, x_n \mid v_j)} \\
&\quad = \ldots\ldots. \\
&\quad = \mathbf{P(x_1 \mid x_2, \ldots, x_n, v_j) P(x_2 \mid x_3, \ldots, x_n, v_j) P(x_3 \mid x_4, \ldots, x_n, v_j) \ldots P(x_n \mid v_j)}
\end{aligned}
$$

- Assumption: feature values are independent given the target value

Naive Bayes (2)

$$V_{MAP} = \text{argmax}_v \; P(x_1, x_2, \ldots, x_n \mid v) P(v)$$

- Assumption: feature values are <u>independent given the target value</u>

$$P(x_1 = b_1, x_2 = b_2, \ldots, x_n = b_n \mid v = v_j) = \Pi_1^n P(x_n = b_n \mid v = v_j)$$

- Generative model:
- First choose a value $v_j \in V$ according to $P(v)$
- For each v_j : choose $x_1 \; x_2, \ldots, x_n$ according to $P(x_k \mid v_j)$

Naive Bayes (3)

$$V_{MAP} = \text{argmax}_v\ P(x_1, x_2, \ldots, x_n \mid v)P(v)$$

- Assumption: feature values are <u>independent given the target value</u>

  $P(x_1 = b_1, x_2 = b_2,\ldots,x_n = b_n \mid v = v_j) = \Pi_1^n P(x_i = b_i \mid v = v_j)$

- Learning method: Estimate $n|V| + |V|$ parameters and use them to make a prediction. (How to estimate?)

- Notice that this is learning without search. Given a collection of training examples, you just compute the best hypothesis (given the assumptions).

- This is learning without trying to achieve consistency or even approximate consistency.
- Why does it work?

Conditional Independence

- Notice that the features values are conditionally independent given the target value, and are not required to be independent.

- Example: The Boolean features are x and y.
  We define the label to be $\ell = f(x,y) = x \wedge y$
  over the product distribution: $p(x=0)=p(x=1)=1/2$ and $p(y=0)=p(y=1)=1/2$
  The distribution is defined so that x and y are independent: $p(x,y) = p(x)p(y)$

  That is:

	X=0	X=1
Y=0	¼ $(\ell = 0)$	¼ $(\ell = 0)$
Y=1	¼ $(\ell = 0)$	¼ $(\ell = 1)$

- But, given that $\ell = 0$:

  $$p(x=1 \mid \ell=0) = p(y=1 \mid \ell=0) = 1/3$$

  while: $$p(x=1,y=1 \mid \ell=0) = 0$$

  so x and y are not conditionally independent.

Conditional Independence

- The other direction also does not hold.
  x and y can be conditionally independent but not independent.

Example: We define a distribution s.t.:
ℓ =0: p(x=1| ℓ =0) =1, p(y=1| ℓ =0) = 0
ℓ =1: p(x=1| ℓ =1) =0, p(y=1| ℓ =1) = 1
and assume, that: p(ℓ =0) = p(ℓ =1)=1/2

	X=0	X=1
Y=0	0 (ℓ= 0)	½ (ℓ= 0)
Y=1	½ (ℓ= 1)	0 (ℓ= 1)

- Given the value of ℓ, x and y are independent (check)
- What about unconditional independence ?

p(x=1) = p(x=1| ℓ =0)p(ℓ =0)+p(x=1| ℓ =1)p(ℓ =1) = 0.5+0=0.5
p(y=1) = p(y=1| ℓ =0)p(ℓ =0)+p(y=1| ℓ =1)p(ℓ =1) = 0+0.5=0.5
But,
p(x=1, y=1)=p(x=1,y=1| ℓ =0)p(ℓ =0)+p(x=1,y=1| ℓ =1)p(ℓ =1) = 0

so x and y are not independent.

Naïve Bayes Example

$$v_{NB} = \mathrm{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i \mid v_j)$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Estimating Probabilities

$$v_{NB} = \text{argmax}_{v \in \{yes,no\}} P(v) \prod_i P(x_i = \text{observation} \mid v)$$

- How do we estimate $P(\text{observation} \mid v)$?

Example

$$v_{NB} = \mathrm{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

- Compute P(PlayTennis= yes); P(PlayTennis= no)
- Compute P(outlook= s/oc/r | PlayTennis= yes/no) (6 numbers)
- Compute P(Temp= h/mild/cool | PlayTennis= yes/no) (6 numbers)
- Compute P(humidity= hi/nor | PlayTennis= yes/no) (4 numbers)
- Compute P(wind= w/st | PlayTennis= yes/no) (4 numbers)

Example

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(x_i | v_j)$$

- Compute P(PlayTennis= yes); P(PlayTennis= no)
- Compute P(outlook= s/oc/r | PlayTennis= yes/no) (6 numbers)
- Compute P(Temp= h/mild/cool | PlayTennis= yes/no) (6 numbers)
- Compute P(humidity= hi/nor | PlayTennis= yes/no) (4 numbers)
- Compute P(wind= w/st | PlayTennis= yes/no) (4 numbers)

- Given a new instance:

  (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)

- Predict: PlayTennis= ?

Example

$$v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i \mid v_j)$$

- **Given:** (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)

P(PlayTennis= yes)=9/14=0.64

P(PlayTennis= no)=5/14=0.36

P(outlook = sunny | yes)= 2/9
P(temp = cool | yes) = 3/9
P(humidity = hi |yes) = 3/9
P(wind = strong | yes) = 3/9

P(outlook = sunny | no)= 3/5
P(temp = cool | no) = 1/5
P(humidity = hi | no) = 4/5
P(wind = strong | no)= 3/5

P(yes, …..) ~ 0.0053

P(no, …..) ~ 0.0206

Example

$$v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i \mid v_j)$$

•**Given:** (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)

P(PlayTennis= yes)=9/14=0.64

P(PlayTennis= no)=5/14=0.36

P(outlook = sunny | yes)= 2/9
P(temp = cool | yes) = 3/9
P(humidity = hi |yes) = 3/9
P(wind = strong | yes) = 3/9

P(outlook = sunny | no)= 3/5
P(temp = cool | no) = 1/5
P(humidity = hi | no) = 4/5
P(wind = strong | no)= 3/5

P(yes, …..) ~ 0.0053

P(no, …..) ~ 0.0206

P(no|instance) = 0.0206/(0.0053+0.0206)=0.795

What if we were asked about Outlook=OC ?

Estimating Probabilities

$$v_{NB} = \text{argmax}_{v \in \{like, dislike\}} P(v) \prod_i P(x_i = word_i \mid v)$$

- How do we estimate $P(word_k \mid v)$?
- As we suggested before, we made a Binomial assumption; then:

$$P(word_k \mid v) = \frac{\#(word_k \text{ appears in training in v documents})}{\#(\text{v documents})} = \frac{n_k}{n}$$

- Sparsity of data is a problem
  - -- if n is small, the estimate is not accurate
  - -- if n_k is 0, it will dominate the estimate: we will never predict v if a word that never appeared in training (with v) appears in the test data

Robust Estimation of Probabilities

$$v_{NB} = \text{argmax}_{v \in \{like, dislike\}} P(v) \prod_i P(x_i = word_i \mid v)$$

- This process is called **smoothing**.
- There are many ways to do it, some better justified than others;
- An empirical issue.

$$P(x_k \mid v) = \frac{n_k + mp}{n + m}$$

Here:

- n_k is # of occurrences of the word in the presence of v
- n is # of occurrences of the label v
- p is a prior estimate of v (e.g., uniform)
- m is *equivalent sample size* (# of labels)
  - Is this a reasonable definition?

Robust Estimation of Probabilities

Smoothing:

$$P(x_k \mid v) = \frac{n_k + mp}{n + m}$$

Common values:

Laplace Rule: for the Boolean case, p=1/2 , m=2

$$P(x_k \mid v) = \frac{n_k + 1}{n + 2}$$

Learn to classify text: p = 1/(|values|) (uniform)
m= |values|

Robust Estimation

- Assume a Binomial r.v.:
  - $p(k|n,\theta) = C_n^k \theta^k (1-\theta)^{n-k}$
- We saw that the maximum likelihood estimate is $\theta_{ML} = k/n$
- In order to compute the MAP estimate, we need to assume a prior.
- It's easier to assume a prior of the form:
  - $p(\theta) = \theta^{a-1} (1-\theta)^{b-1}$ (a and b are called the hyper parameters)
  - The prior in this case is the beta distribution, and it is called a conjugate prior, since it has the same form as the posterior. Indeed, it's easy to compute the posterior:
  - $p(\theta|D) \sim= p(D|\theta)p(\theta) = \theta^{a+k-1} (1-\theta)^{b+n-k-1}$
- Therefore, as we have shown before (differentiate the log posterior)
- $\theta_{map} = k+a-1/(n+a+b-2)$
- The posterior mean:
- $E(\theta|D) = \int_0^1 \theta p(\theta|D)d\theta = a+k/(a+b+n)$
- Under the uniform prior, the posterior mean of observing (k,n) is: k+1/n+2

Naïve Bayes: Two Classes

$$v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i \mid v_j)$$

- Notice that the naïve Bayes method gives a method for predicting rather than an explicit classifier.
- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 1)}{P(v_j = 0) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 0)} > 1$$

Naïve Bayes: Two Classes

$$v_{NB} = \text{argmax}_{v_j \in V} P(v_j) \prod_i P(x_i \mid v_j)$$

- Notice that the naïve Bayes method gives a method for predicting rather than an explicit classifier.
- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 1)}{P(v_j = 0) \bullet \prod_{i=1}^{n} P(x_i \mid v_j = 0)} > 1$$

$$\text{Denote}: p_i = P(x_i = 1 \mid v = 1), \quad q_i = P(x_i = 1 \mid v = 0)$$

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1 - q_i)^{1-x_i}} > 1$$

Naïve Bayes: Two Classes

- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1 - q_i)^{1-x_i}} = \frac{P(v_j = 1) \bullet \prod_{i=1}^{n} (1 - p_i) (\frac{p_i}{1 - p_i})^{x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} (1 - q_i) (\frac{q_i}{1 - q_i})^{x_i}} > 1$$

Naïve Bayes: Two Classes

- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1 - q_i)^{1-x_i}} = \frac{P(v_j = 1) \bullet \prod_{i=1}^{n} (1 - p_i)(\frac{p_i}{1 - p_i})^{x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} (1 - q_i)(\frac{q_i}{1 - q_i})^{x_i}} > 1$$

Take logarithm; we predict $v = 1$ iff :

$$\underline{\log \frac{P(v_j = 1)}{P(v_j = 0)} + \sum_i \log \frac{1 - p_i}{1 - q_i}} + \underline{\sum_i (\log \frac{p_i}{1 - p_i} - \log \frac{q_i}{1 - q_i}) x_i} > 0$$

Naïve Bayes: Two Classes

- In the case of two classes, $v \in \{0,1\}$ we predict that $v=1$ iff:

$$\frac{P(v_j = 1) \bullet \prod_{i=1}^{n} p_i^{x_i} (1-p_i)^{1-x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} q_i^{x_i} (1-q_i)^{1-x_i}} = \frac{P(v_j = 1) \bullet \prod_{i=1}^{n} (1-p_i)(\frac{p_i}{1-p_i})^{x_i}}{P(v_j = 0) \bullet \prod_{i=1}^{n} (1-q_i)(\frac{q_i}{1-q_i})^{x_i}} > 1$$

Take logarithm; we predict $v = 1$ iff :

$$\log \frac{P(v_j = 1)}{P(v_j = 0)} + \sum_i \log \frac{1-p_i}{1-q_i} + \sum_i (\log \frac{p_i}{1-p_i} - \log \frac{q_i}{1-q_i}) x_i > 0$$

- We get that naive Bayes is a linear separator with

$$w_i = \log \frac{p_i}{1-p_i} - \log \frac{q_i}{1-q_i} = \log \frac{p_i}{q_i} \frac{1-q_i}{1-p_i}$$

if $p_i = q_i$ then $w_i = 0$ and the feature is irrelevant

Naïve Bayes: Two Classes

- **In the case of two classes we have that:**

$$\log \frac{P(v_j = 1 | x)}{P(v_j = 0 | x)} = \sum_i w_i x_i - b$$

- **but since**

$$P(v_j = 1 | x) = 1 - P(v_j = 0 | x)$$

- **We get:**

$$P(v_j = 1 | x) = \frac{1}{1 + \exp(-\sum_i w_i x_i + b)}$$

- **Which is simply the logistic function.**

- **The linearity of NB provides a better explanation for why it works.**

We have:
A = 1-B; Log(B/A) = -C.
Then:
Exp(-C) = B/A =
= (1-A)/A = 1/A – 1
= 1 + Exp(-C) = 1/A
A = 1/(1+Exp(-C))

A few more NB examples

Example: Learning to Classify Text

$$v_{NB} = \text{argmax}_{v \in V} P(v) \prod_i P(x_i \mid v)$$

- Instance space X: Text documents
- Instances are labeled according to f(x)=like/dislike

- Goal: Learn this function such that, given a new document you can use it to decide if you like it or not

- How to represent the document ?
- How to estimate the probabilities ?
- How to classify?

Document Representation

- Instance space X: Text documents
- Instances are labeled according to y = f(x) = like/dislike
- **How to represent the document ?**
- A document will be represented as a list of its words
- The representation question can be viewed as the generation question

- We have a dictionary of n words (therefore 2n parameters)
- We have documents of size N: can account for word position & count
- Having a parameter for each word & position may be too much:
  - # of parameters: 2 x N x n (2 x 100 x 50,000 ~ 10^7)
- Simplifying Assumption:
  - The probability of observing a word in a document is independent of its location
  - This still allows us to think about two ways of generating the document

Classification via Bayes Rule (B)

- We want to compute

  $\text{argmax}_y P(y|D) = \text{argmax}_y P(D|y) P(y)/P(D) =$
  $= \text{argmax}_y P(D|y)P(y)$

Parameters:
1. Priors: $P(y=0/1)$
2. $\forall w_i \in$ Dictionary
   $p(w_i = 0/1 \mid y=0/1)$

- Our assumptions will go into estimating $P(D|y)$:

1. Multivariate Bernoulli
   I. To generate a document, first decide if it's good (y=1) or bad (y=0).
   II. Given that, consider your dictionary of words and choose w into your document with probability $p(w \mid y)$, irrespective of anything else.
   III. If the size of the dictionary is $|V|=n$, we can then write
   $P(d|y) = \Pi_1^n P(w_i=1|y)^{b_i} P(w_i=0|y)^{1-b_i}$

- Where:

  $p(w=1/0|y)$: the probability that w appears/does-not in a y-labeled document.

  $b_i \in \{0,1\}$ indicates whether word w_i occurs in document d

- 2n+2 parameters:

  Estimating $P(w_i =1|y)$ and $P(y)$ is done in the ML way as before (counting).

A Multinomial Model

- We want to compute

  $\text{argmax}_y P(y|D) = \text{argmax}_y P(D|y) P(y)/P(D) =$

  $= \text{argmax}_y P(D|y)P(y)$

- Our assumptions will go into estimating $P(D|y)$:

2. Multinomial
   I. To generate a document, first decide if it's good ($y=1$) or bad ($y=0$).
   II. Given that, place N words into d, such that w_i is placed with probability $P(w_i|y)$, and $\sum_i^N P(w_i|y) = 1$.
   III. The Probability of a document is:

$$P(d|y) \; N!/n_1!\ldots n_k! \;\; P(w_1|y)^{n_1} \ldots p(w_k|y)^{n_k}$$

- Where n_i is the # of times w_i appears in the document.
- Same # of parameters: $2n+2$, where $n = |\text{Dictionary}|$, but the estimation is done a bit differently. (HW).

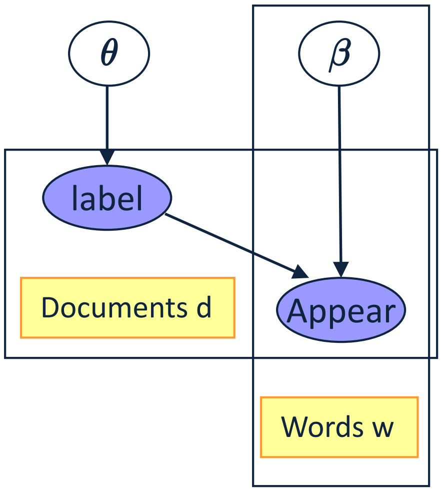
Parameters:
1. Priors: $P(y=0/1)$
2. $\forall \; w_i \in$ Dictionary

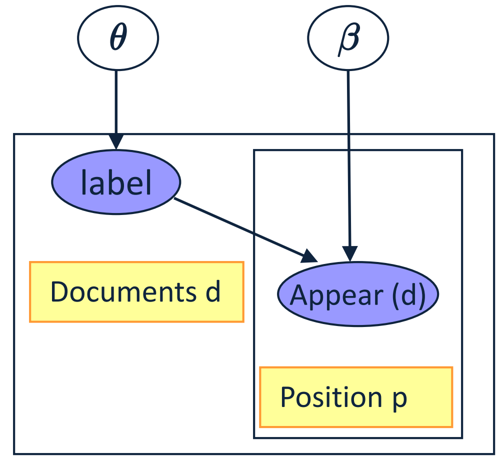
   $p(w_i = 0/1 \; | y=0/1)$

   N dictionary items are chosen into D

Model Representation

- The generative model in these two cases is different





Bernoulli: A binary variable corresponds to a document d and a dictionary word w, and it takes the value 1 if w appears in d. Document topic/label is governed by a prior θ, its topic (label), and the variable in the intersection of the plates is governed by θ and the Bernoulli parameter β for the dictionary word w

Multinomial: Words do not correspond to dictionary words but to positions (occurrences) in the document d. The internal variable is then W(D,P). These variables are generated from the same multinomial distribution β, and depend on the topic/label.

General NB Scenario

- We assume a mixture probability model, parameterized by **μ.**
- Different components $\{c_1, c_2, \ldots c_k\}$ of the model are parameterize by disjoint subsets of μ.

The generative story: A document *d* is created by

(1) selecting a component according to the priors, $P(c_j / \mu)$, then

(2) having the mixture component generate a document according to its own parameters, with distribution $P(d/c_j, \mu)$

- So we have:

$$P(d|\mu) = \sum_1^k P(c_j|\mu)\, P(d|c_j,\mu)$$

- In the case of document classification, we assume a one to one correspondence between components and labels.

Naïve Bayes: Continuous Features

- X_i can be continuous
- We can still use

$$P(X_1, \ldots, X_n | Y) = \prod_i P(X_i | Y)$$

- And

$$P(Y = y | X_1, \ldots, X_n) = \frac{P(Y=y) \prod_i P(X_i | Y=y)}{\sum_j P(Y=y_j) \prod_i P(X_i | Y=y_j)}$$

Naïve Bayes: Continuous Features

- X_i can be continuous
- We can still use

$$P(X_1, \ldots, X_n|Y) = \prod_i P(X_i|Y)$$

- And

$$P(Y = y|X_1, \ldots, X_n) = \frac{P(Y{=}y) \prod_i P(X_i|Y{=}y)}{\sum_j P(Y{=}y_j) \prod_i P(X_i|Y{=}y_j)}$$

- Naïve Bayes classifier:

$$Y = \arg\max_y P(Y = y) \prod_i P(X_i|Y = y)$$

Naïve Bayes: Continuous Features

- X_i can be continuous
- We can still use

$$P(X_1, \ldots, X_n|Y) = \prod_i P(X_i|Y)$$

- And

$$P(Y = y|X_1, \ldots, X_n) = \frac{P(Y=y) \prod_i P(X_i|Y=y)}{\sum_j P(Y=y_j) \prod_i P(X_i|Y=y_j)}$$

- Naïve Bayes classifier:

$$Y = \arg\max_y P(Y = y) \prod_i P(X_i|Y = y)$$

- Assumption: $P(X_i|Y)$ has a Gaussian distribution

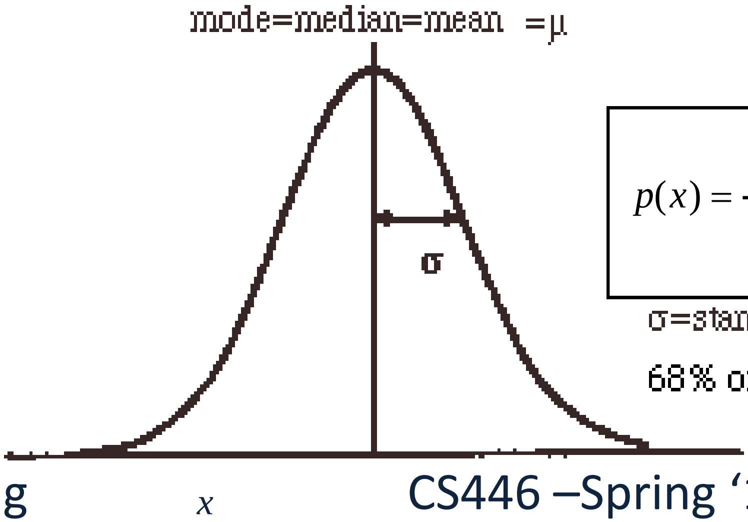
The Gaussian Probability Distribution

- **Gaussian probability distribution also called *normal* distribution.**
- **It is a continuous distribution with pdf:**

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

μ = mean of distribution
σ^2 = variance of distribution
***x* is a continuous variable ($-\infty \leq x \leq \infty$)**

- **Probability of *x* being in the range [*a*, *b*] cannot be evaluated analytically (has to be looked up in a table)**



$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad gaussian$$

σ=standard deviation

68% of area within ±1σ

Naïve Bayes: Continuous Features

- **$P(X_i|Y)$ is Gaussian**

- **Training: estimate mean and standard deviation**

$$\mu_i = E[X_i|Y = y]$$
$$\sigma_i^2 = E[(X_i - \mu_i)^2|Y = y]$$

Note that the following slides abuse notation significantly. Since P(x) =0 for continues distributions, we think of P (X=x| Y=y), not as a classic probability distribution, but just as a function f(x) = N(x, μ, σ^2).
f(x) behaves as a probability distribution in the sense that $\forall$ x, f(x) $\geq$ 0 and the values add up to 1. Also, note that f(x) satisfies Bayes Rule, that is, it is true that:

$$f_Y(y|X = x) = f_X(x|Y = y)\, f_Y(y)/f_X(x)$$

Naïve Bayes: Continuous Features

- **$P(X_i|Y)$ is Gaussian**

- **Training: estimate mean and standard deviation**

$$\mu_i = E[X_i|Y=y]$$
$$\sigma_i^2 = E[(X_i - \mu_i)^2|Y=y]$$

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	.4	1
2	0.3	0	0
2.2	1.1	0	1

Naïve Bayes: Continuous Features

- **$P(X_i|Y)$ is Gaussian**
- **Training: estimate mean and standard deviation**

$$\mu_i = E[X_i|Y=y]$$
$$\sigma_i^2 = E[(X_i-\mu_i)^2|Y=y]$$

X_1	X_2	X_3	Y
2	3	1	1
-1.2	2	.4	1
2	0.3	0	0
2.2	1.1	0	1

$$\mu_1 = E[X_1|Y=1] = \frac{2+(-1.2)+2.2}{3} = 1$$
$$\sigma_1^2 = E[(X_1-\mu_1)|Y=1] = \frac{(2-1)^2+(-1.2-1)^2+(2.2-1)^2}{3} = 2.43$$

Recall: Naïve Bayes, Two Classes

- In the case of two classes we have that:

$$\log \frac{P(v=1|x)}{P(v=0|x)} = \sum_i w_i x_i - b$$

- but since

$$P(v=1|x) = 1 - P(v=0|x)$$

- We get:

$$P(v=1|x) = \frac{1}{1+\exp(-\sum_i w_i x_i + b)}$$

- Which is simply the logistic function (also used in the neural network representation)
- The same formula can be written for continuous features

Logistic Function: Continuous Features

- **Logistic function for Gaussian features**

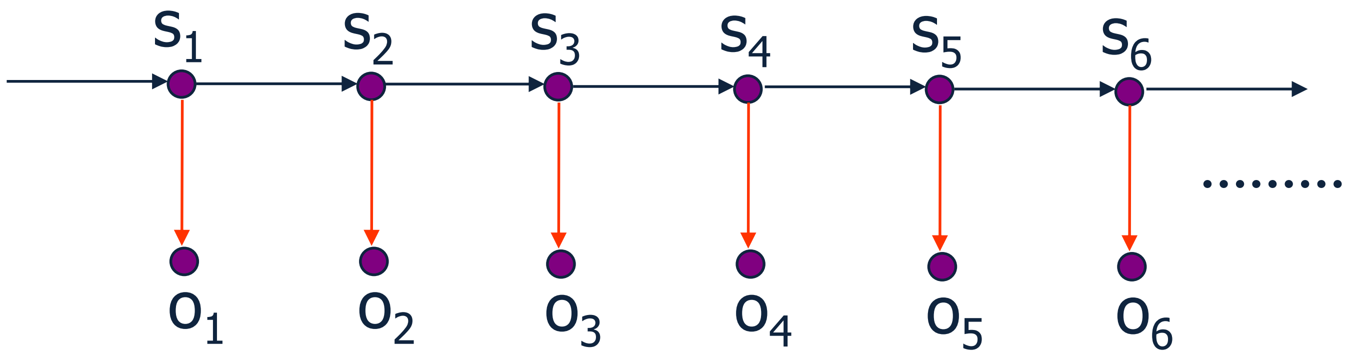
$$P(v=1|x) = \frac{1}{1+exp(log\frac{P(v=0|x)}{P(v=1|x)})}$$
$$= \frac{1}{1+exp(log\frac{P(v=0)P(x|v=0)}{P(v=1)P(x|v=1)})}$$
$$= \frac{1}{1+exp(log\frac{P(v=0)}{P(v=1)}+\sum_i log\frac{P(x_i|v=0)}{P(x_i|v=1)})}$$

Note that we are using ratio of probabilities, since x is a continuous variable.

$$\sum_i log\frac{P(x_i|v=0)}{P(x_i|v=1)} = \sum_i log\frac{\frac{1}{\sqrt{2\pi\sigma_i^2}}exp\left(\frac{-(x_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}}exp\left(\frac{-(x_i-\mu_{i1})^2}{2\sigma_i^2}\right)}$$
$$= \sum_i log\ exp\left(\frac{(x_i-\mu_{i1})^2-(x_i-\mu_{i0})^2}{2\sigma_i^2}\right)$$
$$= \sum_i\left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}x_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)$$

Hidden Markov Model (HMM)

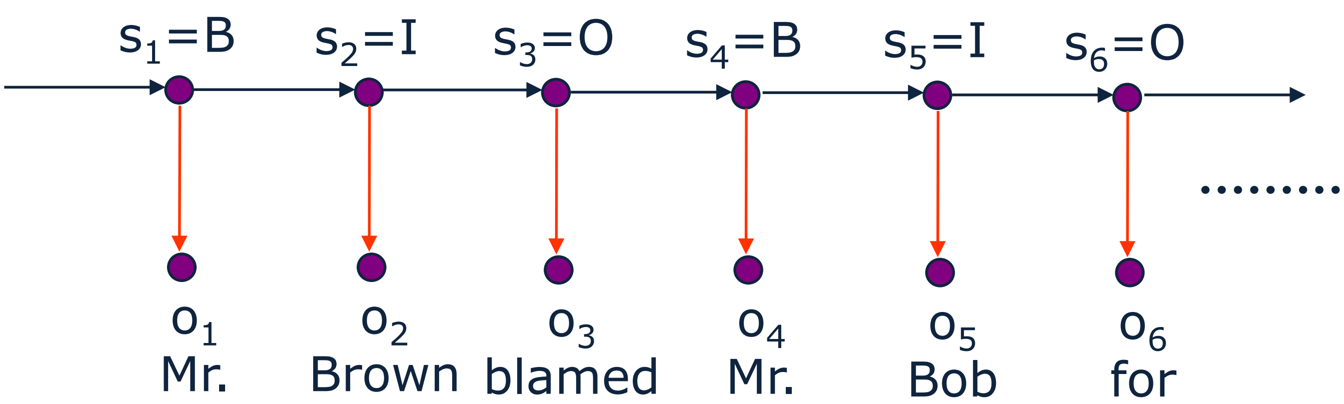
- A probabilistic generative model: models the generation of an observed sequence.
- At each time step, there are two variables: Current state (hidden), Observation



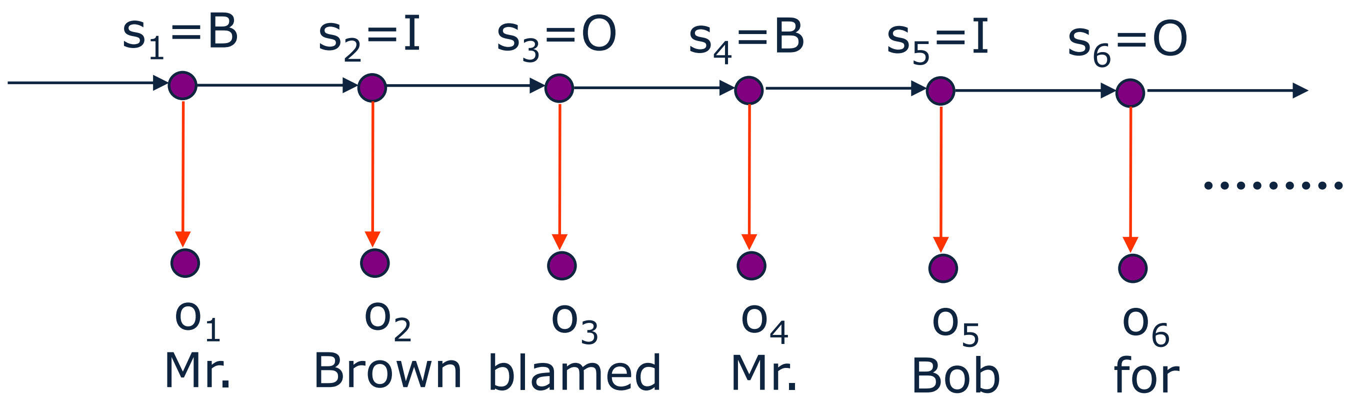
- Elements
  - Initial state probability $P(s_1)$ (|S| parameters)
  - Transition probability $P(s_t|s_{t-1})$ (|S|^2 parameters)
  - Observation probability $P(o_t|s_t)$ (|S|x |O| parameters)
- As before, the graphical model is an encoding of the independence assumptions:
  - $P(s_t|s_{t-1}, s_{t-2},\ldots s_1) = P(s_t|s_{t-1})$
  - $P(o_t| s_T,\ldots,s_t,\ldots s_1, o_T,\ldots,o_t,\ldots o_1) = P(o_t|s_t)$
- Examples: POS tagging, Sequential Segmentation

HMM for Shallow Parsing

- States:
  - {B, I, O}
- Observations:
  - Actual words and/or part-of-speech tags



HMM for Shallow Parsing



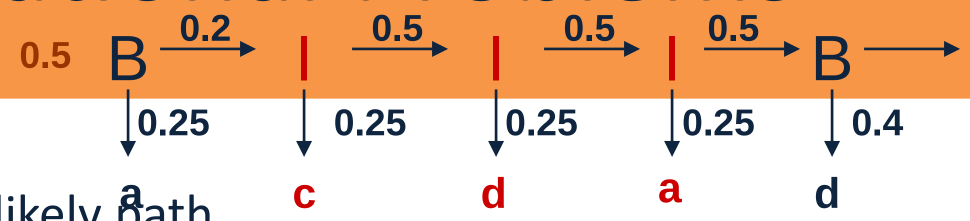
Initial state probability:
$P(s_1=B), P(s_1=I), P(s_1=O)$

Transition probabilty:
$P(s_t=B|s_{t-1}=B), P(s_t=I|s_{t-1}=B), P(s_t=O|s_{t-1}=B),$
$P(s_t=B|s_{t-1}=I), P(s_t=I|s_{t-1}=I), P(s_t=O|s_{t-1}=I),$
…

Observation Probability:
$P(o_t=\text{'Mr.'}|s_t=B), P(o_t=\text{'Brown'}|s_t=B), \ldots,$
$P(o_t=\text{'Mr.'}|s_t=I), P(o_t=\text{'Brown'}|s_t=I), \ldots,$
…

- Given a sentences, we can ask what the most likely state sequence is

Three Computational Problems



- Decoding – finding the most likely path
  - Have: model, parameters, observations (data)
  - Want: most likely states sequence

$$S_1^* S_2^* \ldots S_T^* = \arg\max_{S_1 S_2 \ldots S_T} p(S_1 S_2 \ldots S_T \mid O) = \arg\max_{S_1 S_2 \ldots S_T} p(S_1 S_2 \ldots S_T, O)$$

- Evaluation – computing observation likelihood
  - Have: model, parameters, observations (data)
  - Want: the likelihood to generate the observed data

$$p(O \mid \lambda) = \sum_{S_1 S_2 \ldots S_T} p(O \mid S_1 S_2 \ldots S_T) p(S_1 S_2 \ldots S_T)$$

- In both cases – a simple minded solution depends on $|S|^T$ steps
- Training – estimating parameters
  - Supervised: Have: model, annotated data(data + states sequence)
  - Unsupervised: Have: model, data
  - Want: parameters

Finding most likely state sequence in HMM (1)

$$
\begin{aligned}
&P(s_k, s_{k-1}, \ldots, s_1, o_k, o_{k-1}, \ldots, o_1) \\
&= P(o_k | o_{k-1}, o_{k-2}, \ldots, o_1, s_k, s_{k-1}, \ldots, s_1) \\
&\quad \cdot P(o_{k-1}, o_{k-2}, \ldots, o_1, s_k, s_{k-1}, \ldots, s_1) \\
&= P(o_k | s_k) \cdot P(o_{k-1}, o_{k-2}, \ldots, o_1, s_k, s_{k-1}, \ldots, s_1) \\
&= P(o_k | s_k) \cdot P(s_k | s_{k-1}, s_{k-2}, \ldots, s_1, o_{k-1}, o_{k-2}, \ldots, o_1) \\
&\quad \cdot P(s_{k-1}, s_{k-2}, \ldots, s_1, o_{k-1}, o_{k-2}, \ldots, o_1) \\
&= P(o_k | s_k) \cdot P(s_k | s_{k-1}) \\
&\quad \cdot P(s_{k-1}, s_{k-2}, \ldots, s_1, o_{k-1}, o_{k-2}, \ldots, o_1) \\
&= P(o_k | s_k) \cdot [\prod_{t=1}^{k-1} P(s_{t+1} | s_t) \cdot P(o_t | s_t)] \cdot P(s_1)
\end{aligned}
$$

Finding most likely state sequence in HMM (2)

$$\arg\max_{s_k, s_{k-1}, \ldots, s_1} P(s_k, s_{k-1}, \ldots, s_1 | o_k, o_{k-1}, \ldots, o_1)$$

$$= \arg\max_{s_k, s_{k-1}, \ldots, s_1} \frac{P(s_k, s_{k-1}, \ldots, s_1, o_k, o_{k-1}, \ldots, o_1)}{P(o_k, o_{k-1}, \ldots, o_1)}$$

$$= \arg\max_{s_k, s_{k-1}, \ldots, s_1} P(s_k, s_{k-1}, \ldots, s_1, o_k, o_{k-1}, \ldots, o_1)$$

$$= \arg\max_{s_k, s_{k-1}, \ldots, s_1} P(o_k | s_k) \cdot [\prod_{t=1}^{k-1} P(s_{t+1} | s_t) \cdot P(o_t | s_t)] \cdot P(s_1)$$

Finding most likely state sequence in HMM (3)

A function of s_k

$$\max_{s_k, s_{k-1}, \ldots, s_1} P(o_k|s_k) \cdot [\prod_{t=1}^{k-1} P(s_{t+1}|s_t) \cdot P(o_t|s_t)] \cdot P(s_1)$$

$$= \max_{s_k} P(o_k|s_k) \cdot \max_{s_{k-1}, \ldots, s_1} [\prod_{t=1}^{k-1} P(s_{t+1}|s_t) \cdot P(o_t|s_t)] \cdot P(s_1)$$

$$= \max_{s_k} P(o_k|s_k) \cdot \max_{s_{k-1}} [P(s_k|s_{k-1}) \cdot P(o_{k-1}|s_{k-1})]$$

$$\cdot \max_{s_{k-2}, \ldots, s_1} [\prod_{t=1}^{k-2} P(s_{t+1}|s_t) \cdot P(o_t|s_t)] \cdot P(s_1)$$

$$= \max_{s_k} P(o_k|s_k) \cdot \max_{s_{k-1}} [P(s_k|s_{k-1}) \cdot P(o_{k-1}|s_{k-1})]$$

$$\cdot \max_{s_{k-2}} [P(s_{k-1}|s_{k-2}) \cdot P(o_{k-2}|s_{k-2})] \cdot \ldots$$

$$\cdot \max_{s_1} [P(s_2|s_1) \cdot P(o_1|s_1)] \cdot P(s_1)$$

Finding most likely state sequence in HMM (4)

$$\max_{s_k} P(o_k|s_k) \cdot \max_{s_{k-1}}[P(s_k|s_{k-1}) \cdot P(o_{k-1}|s_{k-1})]$$
$$\cdot \max_{s_{k-2}}[P(s_{k-1}|s_{k-2}) \cdot P(o_{k-2}|s_{k-2})] \cdot \ldots$$
$$\cdot \max_{s_2}[P(s_3|s_2) \cdot P(o_2|s_2)] \cdot$$
$$\cdot \max_{s_1}[P(s_2|s_1) \cdot P(o_1|s_1)] \cdot P(s_1)$$

- Viterbi's Algorithm
  - Dynamic Programming

Learning the Model

- Estimate
  - Initial state probability $P(s_1)$
  - Transition probability $P(s_t|s_{t-1})$
  - Observation probability $P(o_t|s_t)$
- Unsupervised Learning (states are not observed)
  - EM Algorithm
- Supervised Learning (states are observed; more common)
  - ML Estimate of above terms directly from data

- Notice that this is completely analogues to the case of naive Bayes, and essentially all other models.