Generative Models; Naive Bayes

Dan Roth
danroth@seas.upenn.edu|http://www.cis.upenn.edu/~danroth/|461C, 3401 Walnut

Slides were created by Dan Roth (for CIS519/419 at Penn or CS446 at UIUC), Eric Eaton for CIS519/419 at Penn, or from other authors who have made their ML slides available.

Penn Engineering

Administration

- Projects:
  - Come to my office hours at least once to discuss the project.
  - **Posters** for the projects will be presented on the last meeting of the class, Monday, December 9; we'll start earlier: 9am – noon.
  - We will also ask you to prepare a 3-minute video
- Final reports will only be due after the Final exam, on December 19
  - Specific instructions are on the web page and will be sent also on Piazza.

Administration (2)

- Exam:
  - The exam will take place on the originally assigned date, 12/19.
    - Location: TBD
    - Structured similarly to the midterm.
    - 120 minutes; closed books.
  - What is covered:
    - Cumulative!
    - Slightly more focus on the material covered after the previous mid-term.
    - However, notice that the ideas in this class are cumulative!!
    - Everything that we present in class and in the homework assignments
    - Material that is in the slides but is not discussed in class is not part of the material required for the exam.
      - Example 1: We talked about Boosting. But not about boosting the confidence.
      - Example 2: We talked about multiclass classification: OvA, AvA, but not Error Correcting codes, and not about constraint classification (in the slides).
    - We will give practice exams. HW5 will also serve as preparation.

Applied Machine Learning

- The exams are mostly about understanding Machine Learning.
- HW is about applying machine learning.

Go to Bayesian Learning

Recap: Error Driven Learning

- Consider a distribution D over space $\boldsymbol{X} \times Y$
- $\boldsymbol{X}$ - the instance space; Y - set of labels. (e.g. ± 1)

- Can think about the data generation process as governed by $D(\boldsymbol{x})$, and the labeling process as governed by $D(y|\boldsymbol{x})$, such that

$$D(\boldsymbol{x}, y) = D(\boldsymbol{x})\, D(y|\boldsymbol{x})$$

- This can be used to model both the case where labels are generated by a function $y = f(\boldsymbol{x})$, as well as noisy cases and probabilistic generation of the label.

- If the distribution D is known, there is no learning. We can simply predict $y = argmax_y\, D(y|\boldsymbol{x})$

- If we are looking for a hypothesis, we can simply find the one that minimizes the probability of mislabeling:

$$h = argmin_h\, E_{(\boldsymbol{x},y) \sim D}[[\, h(\boldsymbol{x}) \neq y]]$$

Recap: Error Driven Learning (2)

- Inductive learning comes into play when the distribution is not known.
- Then, there are two basic approaches to take.
  - Discriminative (Direct) Learning
- and
  - Bayesian Learning (Generative)
- Running example:
  - Text Correction:
    - "I saw the girl it the park" → I saw the girl in the park

1: Direct Learning

- Model the problem of text correction as a problem of learning from examples.
- Goal: learn directly how to make predictions.

PARADIGM

- Look at many (positive/negative) examples.
- Discover some regularities in the data.
- Use these to construct a prediction policy.

- A policy (a function, a predictor) needs to be specific.
  [it/in] rule: if "the" occurs after the target $\Rightarrow$in
- Assumptions comes in the form of a hypothesis class.

Bottom line: approximating $h: \boldsymbol{X} \rightarrow Y$ is estimating $P(Y|\boldsymbol{X})$.

Direct Learning (2)

- Consider a distribution D over space $\boldsymbol{X}\times Y$
- $\boldsymbol{X}$ - the instance space; Y - set of labels. (e.g. ± 1)
- Given a sample $\{(\boldsymbol{x}, y)\}_1^m$, and a loss function $L(\boldsymbol{x}, y)$
- Find $h \in H$ that minimizes

$$\sum_{i=1,m} D(\boldsymbol{x}_i, y_i) L(h(\boldsymbol{x}_i), y_i) \; + \; Reg$$

L can be: $L(h(\boldsymbol{x}), y) = 1, h(\boldsymbol{x}) \neq y$, otherwise $L(h(\boldsymbol{x}), y) = 0$ (0-1 loss)

$L(h(\boldsymbol{x}), y) = (h(\boldsymbol{x}) - y)^2$ (L_2 loss)

$L(h(\boldsymbol{x}), y) = \max\{0, 1 - y\, h(\boldsymbol{x})\}$ (hinge loss)

$L(h(\boldsymbol{x}), y) = \exp\{-y\, h(\boldsymbol{x})\}$ (exponential loss)

- Guarantees: If we find an algorithm that minimizes loss on the observed data, then learning theory guarantees good future behavior (as a function of $|H|$).

2: Generative Model

- Model the problem of text correction as that of generating correct sentences.
- Goal: learn a model of the language; use it to predict.

PARADIGM

- Learn a probability distribution over all sentences
  - In practice: make assumptions on the distribution's type
- Use it to estimate which sentence is more likely.
  - $\Pr(I\ saw\ the\ girl\ it\ the\ park) >< \Pr(I\ saw\ the\ girl\ in\ the\ park)$

The model is called "generative" since it makes an assumption on how data $\boldsymbol{X}$ is generated given y

Bottom line: the generating paradigm approximates

$$P(\boldsymbol{X}, Y) = P(\boldsymbol{X}|Y)\, P(Y)$$

- Guarantees: We need to assume the "right" probability distribution

Probabilistic Learning

- There are actually two different notions.
- Learning probabilistic concepts
  - The learned concept is a function $c: \mathbf{X} \rightarrow [0,1]$
  - $c(\boldsymbol{x})$ may be interpreted as the probability that the label 1 is assigned to $\boldsymbol{x}$
  - The learning theory that we have studied before is applicable (with some extensions).
- Bayesian Learning: Use of a probabilistic criterion in selecting a hypothesis
  - The hypothesis can be deterministic, a Boolean function.
- It's not the hypothesis – it's the process.
- In practice, as we'll see, we will use the same principles as before, often similar justifications, and similar algorithms.

Remind yourself of basic probability

Probabilities

- 30 years of AI research danced around the fact that the world was inherently uncertain
- Bayesian Inference:
  - Use probability theory and information about independence
  - Reason diagnostically (from evidence (effects) to conclusions (causes))...
  - ...or causally (from causes to effects)
- Probabilistic reasoning only gives probabilistic results
  - i.e., it summarizes uncertainty from various sources
- We will only use it as a tool in Machine Learning.

Concepts

- Probability, Probability Space and Events
- Joint Events
- Conditional Probabilities
- Independence
  - Next week's recitation will provide a refresher on probability
  - Use the material we provided on-line
- Next I will give a very quick refresher

(1) Discrete Random Variables

- Let X denote a random variable
  - X is a mapping from a space of outcomes to R.
    - $[X = x]$ represents an event (a possible outcome) and,
    - Each event has an associated probability

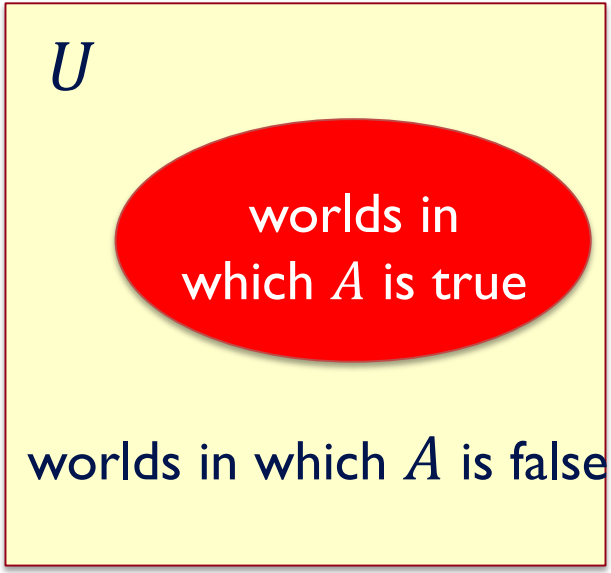
- Examples of binary random variables:
  - A = I have a headache
  - A = Sally will be the US president in 2020

- $P(A = True)$ is "the fraction of possible worlds in which A is true"
  - We could spend hours on the philosophy of this, but we won't

Visualizing A

- Universe U is the event space of all possible worlds
  - Its area is 1
  - $P(U) = 1$
- $P(A)$ = area of red oval
- Therefore:

$$P(A) + P(\neg A) = 1$$
$$P(\neg A) = 1 - P(A)$$



Axioms of Probability

- Kolmogorov showed that three simple axioms lead to the rules of probability theory
  - de Finetti, Cox, and Carnap have also provided compelling arguments for these axioms

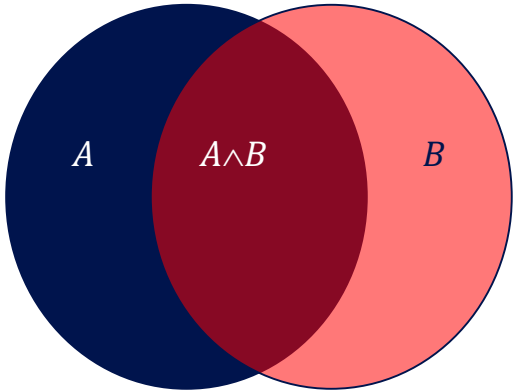
1. All probabilities are between 0 and 1:
$$0 \leq P(A) \leq 1$$

2. Valid propositions (tautologies) have probability 1, and unsatisfiable propositions have probability 0:
$$P(true) = 1; \quad P(false) = 0$$

3. The probability of a disjunction is given by:
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Interpreting the Axioms

- $0 \leq P(A) \leq 1$
- $P(true) = 1$
- $P(false) = 0$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$



- From these you can prove other properties:
- $P(\neg A) = 1 - P(A)$
- $P(A) = P(A \wedge B) + P(A \wedge \neg B)$

Multi-valued Random Variables

- A is a *random variable with arity k* if it can take on exactly one value out of $\{v_1, v_2, \ldots, v_k\}$
- Think about tossing a die
- Thus…
  - $P(A = v_i \wedge A = v_j) = 0 \; if \quad i \neq j$
  - $P(A = v_1 \vee A = v_2 \vee \ldots \vee A = v_k) = 1$

$$1 = \sum_{i=1}^{k} P(A = v_i)$$

Based on slide by Andrew Moore

Multi-valued Random Variables

- We can also show that:

$$P(B) = P(B \wedge [A = v_1 \vee A = v_2 \vee \cdots \vee A = v_k])$$

$$\mathrm{P}(B) = \sum_{i=1}^{k} P(B \wedge A = v_i)$$

- This is called marginalization over *A*

(2) Joint Probabilities

- Joint probability: matrix of combined probabilities of a set of variables

Russell & Norvig's Alarm Domain: (boolean RVs)

- A world has a specific instantiation of variables:
  $(alarm \wedge burglary \wedge \neg earthquake)$
- The joint probability is given by:

$P(alarm, burglary) =$

	$alarm$	$\neg alarm$
$burglary$	0.09	0.01
$\neg burglary$	0.1	0.8

Probability of burglary:
$P(burglary) = 0.1$

by marginalization over $alarm$

The Joint Distribution

- Recipe for making a joint distribution of d variables:

e.g., Boolean variables A, B, C

The Joint Distribution

- Recipe for making a joint distribution of d variables:
  1. Make a truth table listing all combinations of values of your variables (if there are d Boolean variables then the table will have 2^d rows).

e.g., Boolean variables A, B, C

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

The Joint Distribution

- Recipe for making a joint distribution of d variables:
  1. Make a truth table listing all combinations of values of your variables (if there are d Boolean variables then the table will have 2^d rows).
  2. For each combination of values, say how probable it is.

e.g., Boolean variables A, B, C

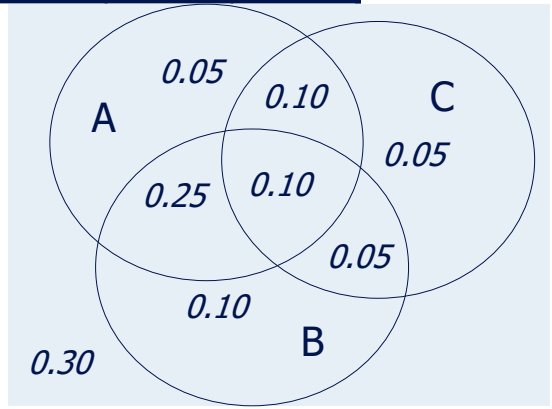
A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

The Joint Distribution

- Recipe for making a joint distribution of *d* variables:
  1. Make a truth table listing all combinations of values of your variables (if there are *d* Boolean variables then the table will have 2^d rows).
  2. For each combination of values, say how probable it is.
  3. If you subscribe to the axioms of probability, those numbers must sum to 1.

e.g., Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



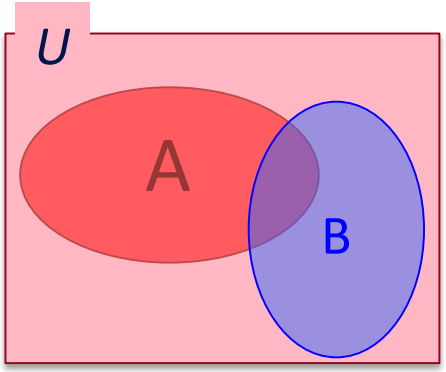
Inferring Probabilities from the Joint

	alarm		¬alarm	
	earthquake	¬earthquake	earthquake	¬earthquake
burglary	0.01	0.08	0.001	0.009
¬burglary	0.01	0.09	0.01	0.79

- $P(alarm) = \sum_{b,e} P(alarm \wedge burglary = b \wedge earthquake = e)$
  $= 0.01 + 0.08 + 0.01 + 0.09 = 0.19$
- $P(burglary) = \sum_{a,e} P(alarm = a \wedge burglary \wedge earthquake = e)$
  $= 0.01 + 0.08 + 0.001 + 0.009 = 0.1$

(3) Conditional Probability

- $P(A \mid B)$ = Fraction of worlds in which B is true that also have *A* true



What if we already know that B is true?

That knowledge changes the probability of *A*

- Because we know we're in a world where *B* is true

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

$$P(A \wedge B) = P(A|B) \times P(B)$$

Example: Conditional Probabilities

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$
$$P(A \wedge B) = P(A|B) \times P(B)$$

$P(alarm, burglary)$ =

	alarm	¬alarm
burglary	0.09	0.01
¬burglary	0.1	0.8

$P(burglary \mid alarm) = P(burglary \wedge alarm) / P(alarm)$
$= 0.09 / 0.19 = 0.47$

$P(alarm \mid burglary) = P(burglary \wedge alarm) / P(burglary)$
$= 0.09/0.1 = 0.9$

$P(burglary \wedge alarm) = P(burglary \mid alarm)\, P(alarm)$
$= 0.47 * 0.19 = 0.09$

(4) Independence

- When two event do not affect each others' probabilities, we call them independent
- Formal definition:

$$A \perp\!\!\!\perp B \quad \leftrightarrow P(A \wedge B) = P(A) \times P(B)$$
$$\leftrightarrow P(A|B) = P(A)$$

Exercise: Independence

P(smart ∧ study ∧ prep)	smart		¬ smart	
	study	¬ study	study	¬ study
prepared	0.432	0.16	0.084	0.008
¬prepared	0.048	0.16	0.036	0.072

- Is *smart* independent of *study*?
- Is *prepared* independent of *study*?

Exercise: Independence

P(smart ∧ study ∧ prep)	smart		¬smart	
	study	¬ study	study	¬ study
prepared	0.432	0.16	0.084	0.008
¬prepared	0.048	0.16	0.036	0.072

Is *smart* independent of *study*?

$$P(study \wedge smart) = 0.432 + 0.048 = \boxed{0.48}$$
$$P(study) = 0.432 + 0.048 + 0.084 + 0.036 = 0.6$$
$$P(smart) = 0.432 + 0.048 + 0.16 + 0.16 = 0.8$$
$$P(study) \times P(smart) = 0.6 \times 0.8 = \boxed{0.48}$$

So yes!

Is *prepared* independent of *study*?

Conditional Independence

- Absolute independence of A and B:

$$A \perp\!\!\!\perp B \quad \leftrightarrow P(A \wedge B) = P(A) \times P(B)$$
$$\leftrightarrow P(A|B) = P(A)$$

- Conditional independence of A and B given C

$$A \perp\!\!\!\perp B|\, \mathrm{C} \leftrightarrow P(A \wedge B\,|C) = P(A|C)\ \times P(B|C)$$

- This lets us decompose the joint distribution:
  - Conditional independence is different than absolute independence, but still useful in decomposing the full joint
  - $P(A, B, C)$ = [Always] $P(A, B|C)\ P(C)$ =
  - = [Independence] $P(A|C)\ P(B|C)\ P(C)$

Exercise: Independence

P(smart ∧ study ∧ prep)	smart		¬ smart	
	study	¬ study	study	¬ study
prepared	0.432	0.16	0.084	0.008
¬ prepared	0.048	0.16	0.036	0.072

- Is *smart* independent of *study*?
- Is *prepared* independent of *study*?

Take Home Exercise: Conditional independence

P(smart ∧ study ∧ prep)	smart		¬ smart	
	study	¬ study	study	¬ study
prepared	0.432	0.16	0.084	0.008
¬ prepared	0.048	0.16	0.036	0.072

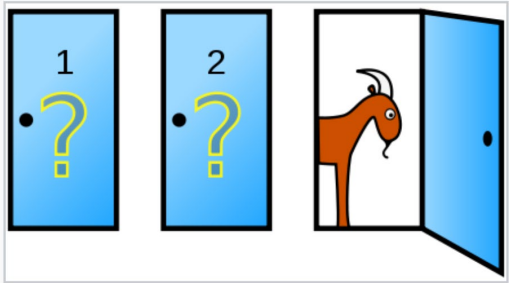
- Is *smart* conditionally independent of *prepared*, given *study*?
- Is *study* conditionally independent of *prepared*, given *smart*?

Summary: Basic Probability

- Product Rule: $P(A,B) = P(A|B)P(B) = P(B|A)P(A)$
- If A and B are independent:
  - $P(A,B) = P(A)P(B)$; $P(A|B) = P(A), P(A|B,C) = P(A|C)$
- Sum Rule: $\mathrm{P}(\mathrm{A} \vee B) = P(A) + P(B) - P(A,B)$
- Bayes Rule: $P(A|B) = P(B|A)\,P(A)/P(B)$
- Total Probability:
  - If events $A_1, A_2, \ldots A_n$ are mutually exclusive: $A_i \wedge\cap A_j = \Phi, \sum_i P(A_i) = 1$
  - $P(B) = \sum P(B, A_i) = \sum_i P(B|A_i)\,P(A_i)$
- Total Conditional Probability:
  - If events $A_1, A_2, \ldots A_n$ are mutually exclusive: $A_i \wedge\cap A_j = \Phi, \sum_i P(A_i) = 1$
  - $P(B|C) = \sum P(B, A_i|C) = \sum_i P(B|A_i, C)\,P(A_i|C)$

Summary: The Monty Hall problem

- Suppose you're on a game show, and you're given the choice of three doors.
  - Behind one door is a car; behind the others, goats.
  - You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat.
  - He then says to you, "Do you want to switch to door No. 2?"
- Is it to your advantage to switch your choice?
- Try to develop an argument using the concepts we discussed.



In search of a new car, the player picks a door, say 1. The game host then opens one of the other doors, say 3, to reveal a goat and offers to let the player pick door 2 instead of door 1.

Bayes' Rule

- Exactly the process we just used
- The most important formula in probabilistic machine learning

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

(Super Easy) Derivation:

$$P(A \wedge B) = P(A|B) \times P(B)$$
$$P(B \wedge A) = P(B|A) \times P(A)$$

Just set equal...

$$P(A|B) \times P(B) = P(B|A) \times P(A)$$

and solve...

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

Bayes' Rule for Machine Learning

- Allows us to reason from evidence to hypotheses
- Another way of thinking about Bayes' rule:

$$P(\text{hypothesis} \mid \text{evidence}) = \frac{P(\text{evidence} \mid \text{hypothesis}) \times P(\text{hypothesis})}{P(\text{evidence})}$$

Basics of Bayesian Learning

- Goal: find the best hypothesis from some space H of hypotheses, given the observed data (evidence) D.

- Define best to be: most probable hypothesis in H
- In order to do that, we need to assume a probability distribution over the class H.

- In addition, we need to know something about the relation between the data observed and the hypotheses (E.g., a coin problem.)

  - As we will see, we will be Bayesian about other things, e.g., the parameters of the model

Basics of Bayesian Learning

- $P(h)$ - the prior probability of a hypothesis h
  Reflects background knowledge; before data is observed. If no information - uniform distribution.

- $P(D)$ - The probability that this sample of the Data is observed. (No knowledge of the hypothesis)

- $P(D|h)$: The probability of observing the sample D, given that hypothesis h is the target

- $P(h|D)$: The posterior probability of h. The probability that h is the target, given that D has been observed.

Bayes Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h|D)$ increases with $P(h)$ and with $P(D|h)$
- $P(h|D)$ decreases with $P(D)$

Learning Scenario

$$P(h|D) = P(D|h)\,P(h)/P(D)$$

- The learner considers a set of candidate hypotheses H (models), and attempts to find the most probable one $h \in H$, given the observed data.

- Such maximally probable hypothesis is called maximum a posteriori hypothesis (MAP); Bayes theorem is used to compute it:

$$h_{MAP} = argmax_{h \in H}\, P(h|D) = argmax_{h \in H}\, P(D|h)\,P(h)/P(D)$$

$$= argmax_{h \in H}\, P(D|h)\,P(h)$$

Learning Scenario (2)

$$h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h) P(h)$$

- We may assume that a priori, hypotheses are equally probable: $P(h_i) = P(h_j) \; \forall \; h_i, h_j \in H$

- We get the Maximum Likelihood hypothesis:

$$h_{ML} = argmax_{h \in H} P(D|h)$$

- Here we just look for the hypothesis that best explains the data

Examples

- $h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h)\, P(h)$

- A given coin is either fair or has a 60% bias in favor of Head.
- Decide what is the bias of the coin [This is a learning problem!]
- Two hypotheses: h_1: $P(H) = 0.5$; h_2: $P(H) = 0.6$

  Prior: $P(h)$: $P(h_1) = 0.75$ $P(h_2) = 0.25$

  Now we need Data. 1st Experiment: coin toss is H.

  – $P(D|h)$:

  $$P(D|h_1) = 0.5\ ;\ P(D|h_2) = 0.6$$

  – $P(D)$:

  $$P(D) = P(D|h_1)P(h_1) + P(D|h_2)P(h_2)$$
  $$= 0.5 \times 0.75 + 0.6 \times 0.25 = 0.525$$

  – $P(h|D)$:

  $$P(h_1|D) = P(D|h_1)P(h_1)/P(D) = 0.5 \times 0.75/0.525 = 0.714$$
  $$P(h_2|D) = P(D|h_2)P(h_2)/P(D) = 0.6 \times 0.25/0.525 = 0.286$$

Examples(2)

- $h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h)\, P(h)$

- A given coin is either fair or has a 60% bias in favor of Head.
- Decide what is the bias of the coin [This is a learning problem!]

- Two hypotheses: h_1: $P(H) = 0.5$; h_2: $P(H) = 0.6$
  - Prior: $P(h)$: $P(h_1) = 0.75$ $P(h_2) = 0.25$

- After 1st coin toss is H we still think that the coin is more likely to be fair

- If we were to use Maximum Likelihood approach (i.e., assume equal priors) we would think otherwise. The data supports the biased coin better.

- Try: 100 coin tosses; 70 heads.
- You will believe that the coin is biased.

Examples(2)

$$h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h)\, P(h)$$

- A given coin is either fair or has a 60% bias in favor of Head.
- Decide what is the bias of the coin [This is a learning problem!]

- Two hypotheses: h_1: $P(H) = 0.5$; h_2: $P(H) = 0.6$
  - Prior: $P(h)$: $P(h_1) = 0.75$ $P(h_2) = 0.25$
- Case of 100 coin tosses; 70 heads.

$$P(D) = P(D|h_1)\, P(h_1) + P(D|h_2)\, P(h_2) =$$
$$= 0.5^{100} \times 0.75 + 0.6^{70} \times 0.4^{30} \times 0.25 =$$
$$= 7.9 \times 10^{-31} \times 0.75 + 3.4 \times 10^{-28} \times 0.25$$

$$0.0057 = P(h_1|D) = P(D|h_1)\, P(h_1)/P(D) << P(D|h_2)\, P(h_2)\, /P(D) = P(h_2|D) = 0.9943$$

In this example we had to choose one hypothesis out of two possibilities; realistically, we could have infinitely many to choose from. We will still follow the maximum likelihood principle.

Maximum Likelihood Estimate

- Assume that you toss a $(p, 1-p)$ coin m times and get k Heads, $m-k$ Tails. What is p?

- If p is the probability of Head, the probability of the data observed is:

$$P(D|p) = p^k (1-p)^{m-k}$$

- The log Likelihood:

$$L(p) = \log P(D|p) = k \log(p) + (m-k)\log(1-p)$$

- To maximize, set the derivative w.r.t. p equal to 0:

$$\frac{dL(p)}{dp} = \frac{k}{p} - \frac{m-k}{1-p}$$

- Solving this for p, gives: $p = \frac{k}{m}$

1. The model we assumed is binomial. You could assume a different model! Next we will consider other models and see how to learn their parameters.

2. In practice, smoothing is advisable – deriving the right smoothing can be done by assuming a prior.

Probability Distributions

- Bernoulli Distribution:
  - Random Variable X takes values $\{0, 1\}$ s.t $P(X = 1) = p = 1 - P(X = 0)$
  - (Think of tossing a coin)
- Binomial Distribution:
  - Random Variable X takes values $\{1, 2, \dots, n\}$ representing the number of successes $(X = 1)$ in n Bernoulli trials.
  - $P(X = k) = f(n, p, k) = C_n^k \, p^k \, (1 - p)^{n-k}$
  - Note that if $X \sim Binom(n, p)$ and $Y \sim Bernoulli\ (p)$, $X = \sum_{i=1}^{n} Y$
  - (Think of multiple coin tosses)

Probability Distributions(2)

- Categorical Distribution:
  - Random Variable X takes on values in $\{1,2,\ldots k\}$ s.t $P(X = i) = p_i$ and $\sum_1^k p_i = 1$
  - (Think of a dice)
- Multinomial Distribution:
  - Let the random variables X_i $(i = 1, 2, \ldots, k)$ indicates the number of times outcome i was observed over the n trials.
  - The vector $X = (X_1, \ldots, X_k)$ follows a multinomial distribution (n, p) where $p = (p_1, \ldots, p_k)$ and $\sum_1^k p_i = 1$
  - $f(x_1, x_2, \ldots x_k, n, p) = P(X_1 = x_1, \ldots X_k = x_k)$
    $= \frac{n!}{x_1! \ldots x_k!} p_1^{x_1} \ldots p_k^{x_k}$ *when* $\sum_{i=1}^k x_i = n$
  - (Think of n tosses of a k sided dice)

A Multinomial Bag of Words

How do we model it?
What is the learning problem?

Our eventual goal will be: Given a document, predict whether it's "good" or "bad"

- We are given a collection of documents written in a three word language $\{a, b, c\}$. All the documents have exactly n words (each word can be either a, b or c).
- We are given a labeled document collection $\{D_1, D_2 \ \dots \ , D_m\}$. The label y_i of document D_i is 1 or 0, indicating whether D_i is "good" or "bad".
- Our generative model uses the multinominal distribution. It first decides whether to generate a good or a bad document (with $P(y_i = 1) = \eta$). Then, it places words in the document; let a_i (b_i, c_i, resp.) be the number of times word a (b, c, resp.) appears in document D_i. That is, we have $a_i + b_i + c_i = |D_i| = n$.
- In this generative model, we have:

$$P(D_i|y = 1) = n!/(a_i!\ b_i!\ c_i!)\ \alpha_1^{a_i}\ \beta_1^{b_i}\ \gamma_1^{c_i}$$

where α_1 (β_1, γ_1 resp.) is the probability that a (b , c) appears in a "good" document.

- Similarly, $P(D_i|y = 0) = n!/(a_i!\ b_i!\ c_i!)\ \alpha_0^{a_i}\ \beta_0^{b_i}\ \gamma_0^{c_i}$
- Note that: $\alpha_0 + \beta_0 + \gamma_0 = \alpha_1 + \beta_1 + \gamma_1 = 1$

Unlike the discriminative case, the "game" here is different:

- ❑ We make an assumption on how the data is being generated.
  - ❑ (multinomial, with $\eta, \alpha_i, \beta_i, \gamma_i$)
- ❑ We observe documents, and estimate these parameters (that's the learning problem).
- ❑ Once we have the parameters, we can predict the corresponding label.

A Multinomial Bag of Words (2)

- We are given a collection of documents written in a three word language $\{a, b, c\}$. All the documents have exactly n words (each word can be either a, b *or* c).
- We are given a labeled document collection $\{D_1, D_2 \ \ldots \ , D_m\}$. The label y_i of document D_i is 1 or 0, indicating whether D_i is "good" or "bad".

- The classification problem: given a document D, determine if it is good or bad; that is, determine $P(y|D)$.

- This can be determined via Bayes rule: $P(y|D) \ = \ P(D|y)\, P(y)/P(D)$

- But, we need to know the parameters of the model to compute that.

A Multinomial Bag of Words (3)

- How do we estimate the parameters?
- We derive the most likely value of the parameters defined above, by maximizing the log likelihood of the observed data.
- $PD = \prod_i P(y_i, Di) = \prod_i P(D_i | y_i) P(y_i) =$

Labeled data, assuming that the examples are independent

  - We denote by $P(y_i = 1) = \eta$ the probability that an example is "good" ($y_i = 1$; otherwise y_i 0). Then:

$$\prod_i P(y, D_i) = \prod_i [\left(\eta \frac{n!}{a_i!\ b_i!\ c_i!} \alpha_1^{a_i} \beta_1^{b_i} \gamma_1^{c_i}\right)^{y_i} \cdot \left((1-\eta) \frac{n!}{a_i!\ b_i!\ c_i!} \alpha_0^{a_i} \beta_0^{b_i} \gamma_0^{c_i}\right)^{1-y_i}$$

Notice that this is an important trick to write down the joint probability without knowing what the outcome of the experiment is. The ith expression evaluates to $p(D_i, y_i)$ (Could be written as a sum with multiplicative y_i but less convenient)

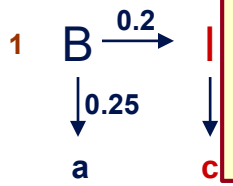
- We want to maximize it with respect to each of the parameters. We first compute $\log(PD)$ and then differentiate:
- $\log(PD) = \sum_i y_i [\log(\eta) + C + a_i \log(\alpha_1) + b_i \log(\beta_1) + c_i \log(\gamma_1)] + (1 - y_i) [\log(1-\eta) + C' + a_i \log(\alpha_0) + b_i \log(\beta_0) + c_i \log(\gamma_0)]$
- $\frac{d log PD}{\eta} = \sum_i [\frac{y_i}{\eta} - \frac{1-y_i}{1-\eta}] = 0 \rightarrow \sum_i (y_i - \eta) = 0 \rightarrow \eta = \sum_i \frac{y_i}{m}$

Makes sense?

- The same can be done for the other 6 parameters. However, notice that they are not independent: $\alpha_0 + \beta_0 + \gamma_0 = \alpha_1 + \beta_1 + \gamma_1 = 1$ and also $a_i + b_i + c_i = |D_i| = n$.

Other Examples (HMMs)

- Consider data over 5 characters, $x = a, b$
  - Think about chunking a sentence to phrases
  - (Or: think about being in one of two rooms,
- We generate characters according to:
- Initial state prob: $p(B) = 1;\ p(I) = 0$
- State transition prob:
  - $p(B \rightarrow B) = 0.8\ p(B \rightarrow I) = 0.2$
  - $p(I \rightarrow B) = 0.5\ p(I \rightarrow I) = 0.5$
- Output prob:
  - $p(a|B) = 0.25, p(b|B) = 0.10, p(c|B$
  - $p(a|I) = 0.25, p(b, I) = 0, \ldots$
- Can follow the generation process to ge



- We can do the same exercise we did before.
- Data: $\{(x_1, x_2, \ldots x_m, s_1, s_2, \ldots s_m)\}_1^n$
- Find the most likely parameters of the model:
  $P(x_i|s_i), P(s_{i+1}|s_i), p(s_1)$
- Given an unlabeled example (observation)
  $x = (x_1, x_2, \ldots x_m)$
- use Bayes rule to predict the label $l = (s_1, s_2, \ldots s_m)$:
  $l^* = argmax_l\ P(l|x) = argmax_l\ P(x|l)\ P(l)/P(x)$
- The only issue is computational: there are 2^m possible values of l (labels)
- This is an HMM model (but nothing was hidden; the s_i were given in training. It's also possible to solve when the $s_1, s_2, \ldots s_m$ are hidden, via EM.

Bayes Optimal Classifier

- How should we use the general formalism?
- What should H be?
- H can be a collection of functions. Given the training data, choose an optimal function. Then, given new data, evaluate the selected function on it.
- H can be a collection of possible predictions. Given the data, try to directly choose the optimal prediction.
- Could be different!

Bayes Optimal Classifier

- The first formalism suggests to learn a good hypothesis and use it.
- (Language modeling, grammar learning, etc. are here)

$$h_{MAP} = argmax_{h \in H} P(h|D) = argmax_{h \in H} P(D|h)P(h)$$

- The second one suggests to directly choose a decision.[it/in]:
- This is the issue of "thresholding" vs. entertaining all options until the last minute. (Computational Issues)

Bayes Optimal Classifier: Example

- Assume a space of 3 hypotheses:
  - $P(h_1|D) = 0.4;\ P(h_2|D) = 0.3;\ P(h_3|D) = 0.3 \rightarrow h_{MAP} = h_1$
- Given a new instance x, assume that
  - $h_1(\boldsymbol{x}) = 1 \qquad h_2(\boldsymbol{x}) = 0 \qquad h_3(\boldsymbol{x}) = 0$
- In this case,
  - $P(f(\boldsymbol{x}) = 1) = 0.4\ ;\ P(f(\boldsymbol{x}) = 0) = 0.6 \quad but \quad h_{MAP}(\boldsymbol{x}) = 1$

- We want to determine the most probable classification by combining the prediction of all hypotheses, weighted by their posterior probabilities
- Think about it as deferring your commitment – don't commit to the most likely hypothesis until you have to make a decision (this has computational consequences)

Bayes Optimal Classifier: Example(2)

- Let V be a set of possible classifications

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i, D)P(h_i|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Bayes Optimal Classification:

$$v = argmax_{v_j \in V} P(v_j|D) = argmax_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- In the example:

$$P(1|D) = \sum_{h_i \in H} P(1|h_i)P(h_i|D) = 1 * 0.4 + 0 * 0.3 + 0 * 0.3 = 0.4$$

$$P(0|D) = \sum_{h_i \in H} P(0|h_i)P(h_i|D) = 0 * 0.4 + 1 * 0.3 + 1 * 0.3 = 0.6$$

- and the optimal prediction is indeed 0.
- The key example of using a "Bayes optimal Classifier" is that of the Naïve Bayes algorithm.

Bayesian Classifier

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

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

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

$$v_{MAP} = argmax_{v_j \in V} P(v_j | x) = argmax_{v_j \in V} P(v_j | x_1, x_2, \ldots x_n)$$

$$v_{MAP} = argmax_{v_j \in V} \frac{P(x_1, x_2, \ldots, x_n | v_j) P(v_j)}{P(x_1, x_2, \ldots, x_n)} = argmax_{v_j \in V} P(x_1, x_2, \ldots, x_n | v_j) P(v_j)$$

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

Bayesian Classifier

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

- 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

$$V_{MAP} = argmax_v P(x_1, x_2, \dots, x_n \mid v)P(v)$$

$$P(x_1, x_2, \dots, x_n | v_j) = P(x_1 | x_2, \dots, x_n, v_j) P(x_2, \dots, x_n | v_j)$$
$$= P(x_1 | x_2, \dots, x_n, v_j)\ P(x_2 | x_3, \dots, x_n, v_j) P(x_3, \dots, x_n | v_j)$$
$$= \dots$$
$$= P(x_1 | x_2, \dots, x_n, v_j)\ P(x_2 | x_3, \dots, x_n, v_j) P(x_3 | x_4, \dots, x_n, v_j) \dots P(x_n | v_j)$$
$$= \prod_{i=1}^{n} P(x_i | v_j)$$

- Assumption: feature values are independent given the target value

Naive Bayes (2)

$$V_{MAP} = argmax_v P(x_1, x_2, \dots, x_n \mid v) P(v)$$

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

  $P(x_1 = b_1, x_2 = b_2, \dots, x_n = b_n \mid v = v_j) \prod_{i=1}^{n} (x_i = b_i \mid v = v_j)$
- Generative model:
- First choose a value $v_j \in V$ according to $P(v)$
- For each v_j : choose $x_1 x_2, \dots, x_n$ according to $P(x_k \mid v_j)$

Naive Bayes (3)

$$V_{MAP} = argmax_v P(x_1, x_2, \dots, x_n \mid v)P(v)$$

- Assumption: feature values are independent given the target value

$$P(x_1 = b_1, x_2 = b_2, \dots, x_n = b_n | v = v_j) \prod_{i=1}^{n} (x_i = b_i | v = v_j)$$

- 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 $l = f(x,y) = x \wedge y$ over the product distribution:
  $$p(x=0) = p(x=1) = \frac{1}{2} \quad \text{and} \quad p(y=0) = p(y=1) = \frac{1}{2}$$
  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 $(l=0)$	¼ $(l=0)$
$Y=1$	¼ $(l=0)$	¼ $(l=1)$

- But, given that $l = 0$:
  $$p(x=1 \mid l=0) = p(y=1 \mid l=0) = \frac{1}{3}$$
  while: $p(x=1, y=1 \mid l=0) = 0$
  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: $p(l = 0) = p(l = 1) = \frac{1}{2}$

	$X = 0$	$X = 1$
$Y = 0$	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

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

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

- $v_{NB} = argmax_{v \in \{yes,no\}} \mathrm{P}(\mathrm{v}) \prod_i P(x_i = observation|v)$
- How do we estimate $P(observation|v)$?

Example

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

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

Example

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

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

- Given a new instance:

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

- Predict: $PlayTennis = ?$

Example

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

- Given: (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)
- $P(PlayTennis = yes) = 9/14 = 0.64)$ $P(PlayTennis = no) = 5/14 = 0.36$

- $P(outlook = sunny \mid yes) = 2/9$ $P(outlook = sunny \mid no) = 3/5$
- $P(temp = cool \mid yes) = 3/9$ $P(temp = cool \mid no) = 1/5$
- $P(humidity = hi \mid yes) = 3/9$ $P(humidity = hi \mid no) = 4/5$
- $P(wind = strong \mid yes) = 3/9$ $P(wind = strong \mid no) = 3/5$

- $P(yes, \ldots..) \sim 0.0053$ $P(no, \ldots..) \sim 0.0206$

Example

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

- Given: (Outlook=sunny; Temperature=cool; Humidity=high; Wind=strong)
- $P(PlayTennis = yes) = 9/14 = 0.64)$ $P(PlayTennis = no) = 5/14 = 0.36$

- $P(outlook = sunny \mid yes) = 2/9$ $P(outlook = sunny \mid no) = 3/5$
- $P(temp = cool \mid yes) = 3/9$ $P(temp = cool \mid no) = 1/5$
- $P(humidity = hi \mid yes) = 3/9$ $P(humidity = hi \mid no) = 4/5$
- $P(wind = strong \mid yes) = 3/9$ $P(wind = strong \mid no) = 3/5$

- $P(yes, \ldots..) \sim 0.0053$ $P(no, \ldots..) \sim 0.0206$
- $P(no|instance) = 0.0206/(0.0053 + 0.0206) = 0.795$

What if we were asked about Outlook=OC ?

Estimating Probabilities

$$v_{NB} = argmax_v \prod_i P(x_i|v)P(v)$$

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

$$\mathrm{P}(x_i = 1|v) = \frac{\#(x_i{=}1 \; in \; training \; in \; an \; example \; labeld \; v)}{\#(v \; labeled \; examples)} = \frac{n_i}{n}$$

- Sparsity of data is a problem
  - if n is small, the estimate is not accurate
  - if n_i 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

$$v_{NB} = argmax_{v \in \{like, dislike\}} \mathrm{P(v)} \prod_i P(x_i = word_i | v)$$

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

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

Here:

- n_k is # of occurrences of the k-th feature given the label 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:

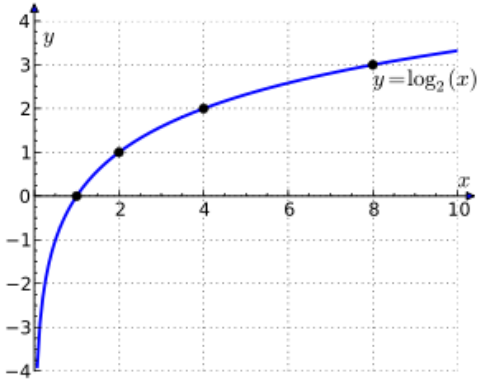
$$P(x_k|v) = \frac{n_k + mp}{n + m}$$

- Common values:
  - Laplace Rule for the Boolean case, $p = \frac{1}{2}$, $m = 2$

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

Another comment on estimation

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



- **In practice, we will typically work in the log space:**
- $v_{NB} = argmax \log P(v_j) \prod_i P(x_i \mid v_j) =$

$$= argmax \left[\log P(v_j) + \sum_i \log P(x_i | v_j) \right]$$

Naïve Bayes: Two Classes

Why do things work?

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

- 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:

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

Naïve Bayes: Two Classes

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

- 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:

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

Denote $p_i = P(x_i = 1|v = 1), q_i = P(x_i = 1|v = 0)$

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

Naïve Bayes: Two Classes

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

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

Naïve Bayes: Two Classes

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

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

Take logarithm; we predict $v = 1$ iff

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

Naïve Bayes: Two Classes

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

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

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

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

- 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:

$$\log \frac{P(v_j = 1|x)}{P(v_j = 0|x)} = \sum_i \boldsymbol{w}_i \boldsymbol{x}_i - b$$

- but since

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

- We get:

$$P(v_j = 1|x) = \frac{1}{1 + \exp(-\sum_i \boldsymbol{w}_i \boldsymbol{x}_i + b)}$$

- 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:

$$Exp(-C) = B/A =$$
$$= (1 - A)/A = 1/A - 1$$
$$= 1 + Exp(-C) = 1/A$$
$$A = 1/(1 + Exp(-C))$$

Why Does it Work?

- **Learning Theory**
  - Probabilistic predictions are done via Linear Models
  - The low expressivity explains Generalization + Robustness

$$Err_s(h) = \frac{|\{x \in S \,|h(x) \neq 1\}|}{|S|}$$

- **Expressivity (1: Methodology)**
  - Is there a reason to believe that these hypotheses minimize the empirical error?
    - In General, No. (Unless some probabilistic assumptions happen to hold).
  - But: if the hypothesis does not fit the training data,
    - Experimenters will augment set of features (in fact, add dependencies)
  - But now, this actually follows the Learning Theory Protocol:
    - Try to learn a hypothesis that is consistent with the data
    - Generalization will be a function of the low expressivity
- **Expressivity (2: Theory)**
  - (a) Product distributions are "dense" in the space of all distributions.
    - Consequently, the resulting predictor's error is actually close to optimal classifier.
  - (b) NB classifiers cannot express all linear functions; but they converge faster to what they can express.

What's Next?

(1) If probabilistic hypotheses are actually like other linear functions, can we interpret the outcome of other linear learning algorithms probabilistically?

- Yes

(2) If probabilistic hypotheses are actually like other linear functions, can you train them similarly (that is, discriminatively)?

- Yes.
- Classification: Logistics regression/Max Entropy
- HMM: can be learned as a linear model, e.g., with a version of Perceptron (Structured Models class)

Recall: Naïve Bayes, Two Classes

- In the case of two classes we have:

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

- but since

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

- We get (plug in (2) in (1); some algebra):

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

- Which is simply the logistic (sigmoid) function used in the
- neural network representation.

Conditional Probabilities

- (1) If probabilistic hypotheses are actually like other linear functions, can we interpret the outcome of other linear learning algorithms probabilistically?
  - Yes
- General recipe
  - Train a classifier f using your favorite algorithm (Perceptron, SVM, Winnow, etc). Then:
  - Use $\Pr(y = 1|x) \approx P_{A,B}(f) \equiv \frac{1}{1+\exp(Af+B)}$, where $f = \mathrm{f}(x)$.
  - A, B can be tuned using a held out that was not used for training.
  - Done in LBJava, for example

Logistic Regression (1)

- (2) If probabilistic hypotheses are actually like other linear functions, can you actually train them similarly (that is, discriminatively)?
- The logistic regression model assumes the following model:

$$P(y = +1 \mid \boldsymbol{x}, \boldsymbol{w}) = [1 + \exp(-y(\boldsymbol{w}^T\boldsymbol{x} + b)]^{-1}$$

- This is the same model we derived for naïve Bayes, only that now we will **not** assume any independence assumption.

  We will directly find the best $\boldsymbol{w}$.

- Therefore training will be more difficult. However, the weight vector derived will be more expressive.
  - It can be shown that the naïve Bayes algorithm cannot represent all linear threshold functions.
  - On the other hand, NB converges to its performance faster.

How?

Logistic Regression (2)

- Given the model:

$$P(y = +1 \mid \boldsymbol{x}, \boldsymbol{w}) = [1 + \exp(-y(\boldsymbol{w}^T\boldsymbol{x} + b)]^{-1}$$

- The goal is to find the $(\boldsymbol{w}, b)$ that maximizes the log likelihood of the data: $\{x_1, x_2 \ldots x_m\}$.
- We are looking for $(\boldsymbol{w}, b)$ that minimizes the negative log-likelihood

$$\min_{\boldsymbol{w},b} \sum_1^m \log P(y = +1 \mid x, w) = \min_{\boldsymbol{w},b} \sum_1^m \log[1 + \exp(-y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)]$$

- This optimization problem is called Logistics Regression.

- Logistic Regression is sometimes called the Maximum Entropy model in the NLP community (since the resulting distribution is the one that has the largest entropy among all those that activate the same features).

Logistic Regression (3)

- Using the standard mapping to linear separators through the origin, we would like to minimize:

$$\min_{\boldsymbol{w},b} \sum_1^m \log P(y = +1 \mid x, w) = \min_{\boldsymbol{w},b} \sum_1^m \log[1 + \exp(-y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)]$$

- To get good generalization, it is common to add a regularization term, and the regularized logistics regression then becomes:

$$min_w f(w) = \underbrace{½\, w^T w}_{\text{Regularization term}} + \underbrace{C \sum_1^m \log[1 + \exp(-y_i(w^T x_i)]}_{\text{Empirical loss}},$$

Where C is a user selected parameter that balances the two terms.

Comments on Discriminative Learning

- $min_w f(w) = \frac{1}{2} w^T w + C \sum_1^m \log[1 + \exp(-y_i(w^T x_i)],$

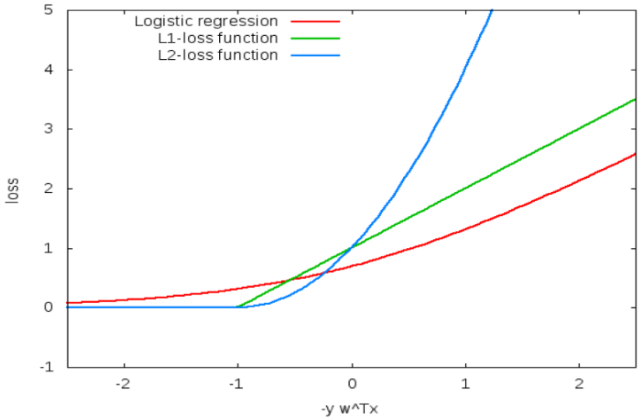
  Regularization term — Empirical loss

  Where C is a user selected parameter that balances the two terms.

- Since the second term is the loss function
- Therefore, regularized logistic regression can be related to other learning methods, e.g., SVMs.
- L_1 SVM solves the following optimization problem:

  $min_w f_1(w) = \frac{1}{2} w^T w + C \sum_1^m \max(0, 1 - y_i(w^T x_i))$
- L_2 SVM solves the following optimization problem:

  $min_w f_2(w) = \frac{1}{2} w^T w + C \sum_1^m (\max(0, 1 - y_i w^T x_i))^2$



A few more NB examples

Example: Learning to Classify Text

- 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?

$$v_{NB} = argmax_{v \in V} P(v) \prod_i P(x_i|v)$$

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: $2 \times N \times n$ $(2 \times 100 \times 50{,}000 \sim 10^7)$
- 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
- $argmax_y P(y|D) = argmax_y P(D|y)\, P(y)/P(D) =$
- $= argmax_y P(D|y)P(y)$

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

- 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 $p(w\,|y)$, irrespective of anything else.
   III. If the size of the dictionary is $|V| = n$, we can then write

- $P(d|y) = \prod_1^n P(w_i = 1|y)^{b_i} P(w_i = 0|y)^{1-b_i}$
- 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
- $argmax_y P(y|D) = argmax_y P(D|y)\, P(y)/P(D) =$
- $= argmax_y P(D|y)P(y)$

Parameters:
1. Priors: $P(y = 0/1)$
2. $\forall\, w_i \in Dictionary$ $p(w_i = 0/1\,|y = 0/1)$
   N dictionary items are chosen into D

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

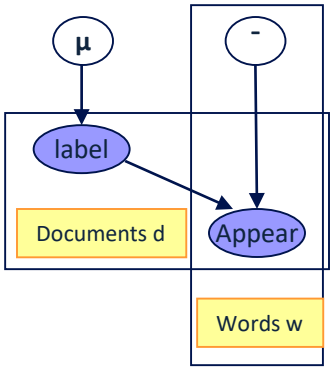
2. Multinomial
   I. To generate a document, first decide if it's good $(y = 1)$ or bad $(y = 0)$.
   II. Given that, place N words into d, such that w_i is placed with probability $P(w_i|y)$, and $\sum_i^N P(w_i|y) = 1$.
   III. The Probability of a document is:

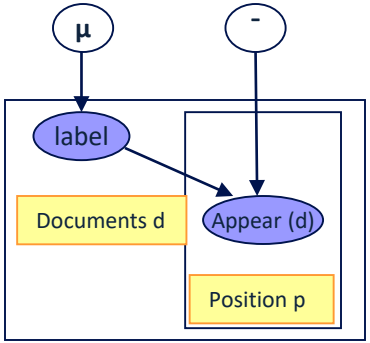
$$P(d|y) = N!/n_1! \ldots n_k!\; P(w_1|y)^{n_1} \ldots p(w_k|y)^{n_k}$$

- 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).

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 μ, its topic (label), and the variable in the intersection of the plates is governed by μ and the Bernoulli parameter ¯ 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 ¯, 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

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

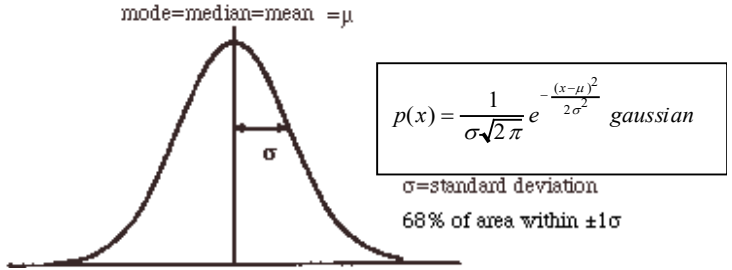
Naïve Bayes classifier:

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

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



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 $f(x) = N(x,\ \mu, \sigma^2)$. $f(x)$ behaves as a probability distribution in the sense that $8\ x, f(x)\ ,\ 0$ 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	0.4	1
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
  - $\mu_i = E[X_i|Y = y]$
  - $\sigma_i^2 = E[(X_i - \mu_i)^2|Y = y]$
  - $\mu_1 = E[X_1|Y = 1] = \frac{2+(-1.2)+2.2}{3} = 1$
  - $\sigma_1^2 = E[(X_1 - \mu_1)|Y = 1] = \frac{(2-1)^2+(-1.2-1)^2+(2.2-1)^2}{3} = 2.43$

X_1	X_2	X_3	Y
2	3	1	1
−1.2	2	0.4	1
1.2	0.3	0	0
2.2	1.1	0	1

Recall: Naïve Bayes, Two Classes

- In the case of two classes we have that:

$$\log \frac{P(v=1|x)}{P(v=0|x)} = \sum_i \boldsymbol{w}_i \boldsymbol{x}_i - b$$

- but since

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

- We get:

$$P(v=1|x) = \frac{1}{1+\exp(-\sum_i \boldsymbol{w}_i \boldsymbol{x}_i + b)}$$

- 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**

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

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

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

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

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

$$= \sum_i \log\exp(\frac{(x_i-\mu_{i1})^2 - (x_i-\mu_{i0})^2}{2\,\sigma_i^2})$$

$$= \sum_i (\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}x_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2})$$