IS 733 Lesson 7

Supervised Learning (Continued)

Slides based on those from Data Mining by I. H. Witten, E. Frank, M. A. Hall and C. J. Pal, Data Mining: Concepts and Techniques by Han et al., and Vandana Janeja and James Foulds

In the naive Bayes classifier, it is assumed that the attributes are _____, given the class. This means that their probabilities can be _____ together to obtain their combined probability.

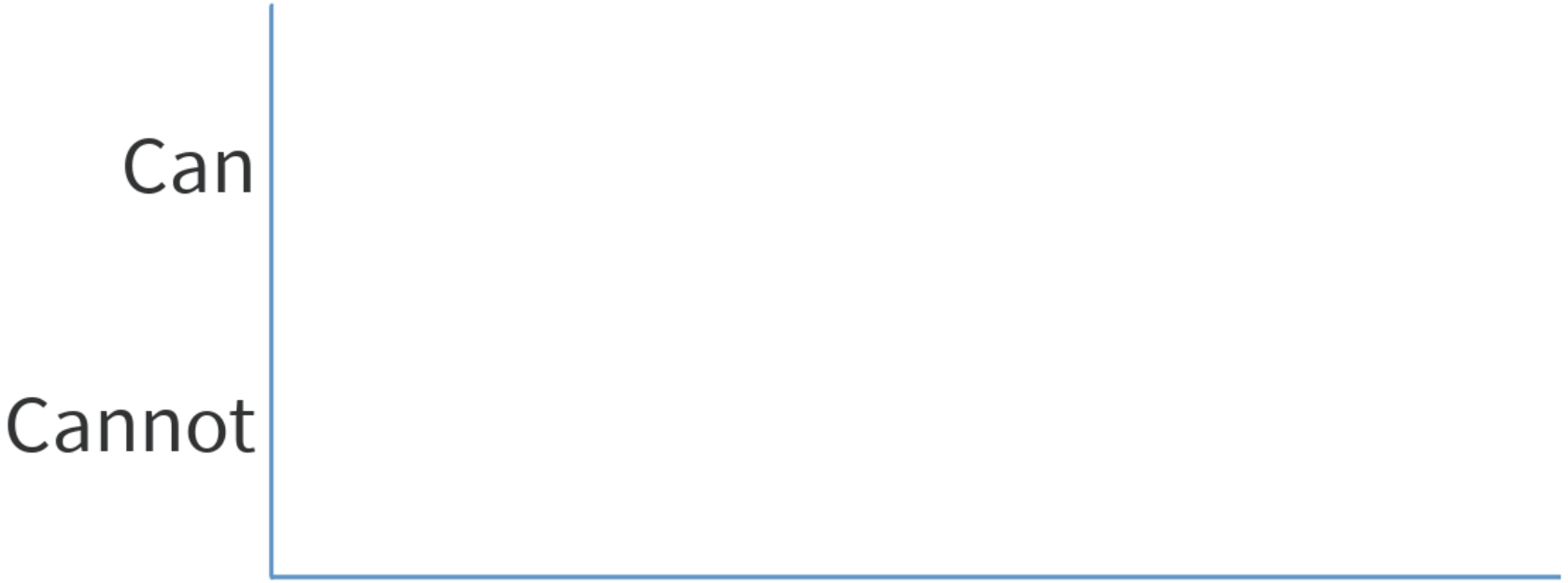
Independent, added

Not independent, added

Independent, multiplied

Not independent, multiplied

Sets of data points that cannot be separated by a single hyperplane _____ be discriminated correctly by logistic regression.



For support vector machines, the instances with the ____ distance to the _____ margin hyperplane are called the support vectors.

Maximum, maximum

Maximum, minimum

Minimum, maximum

Minimum, minimum

Learning outcomes

By the end of the lesson, you should be able to:

- **Perform** the steps of the training algorithms for the **naïve Bayes classifier**, given a small dataset
- **Explain** the assumptions made by naïve Bayes models
- **Discuss** how **logistic regression** and **support vector machines** (**SVMs**) extend the simple linear regression model into powerful techniques for classification

Simple probabilistic modeling: Naïve Bayes classifier

- Remember how 1R uses only one attribute?

  Naïve Bayes is "opposite" of 1R: use all the attributes

- Two assumptions: Attributes are
  - ***equally important***
  - ***statistically independent*** (given the class value)
    - This means knowing the value of one attribute tells us nothing about the value of another takes on (if the class is known)

Simple probabilistic modeling: Naïve Bayes classifier

Offer	Cheap	Buy	Sale	…	Spam?
1	0	0	1		1

Naïve Bayes assumption:

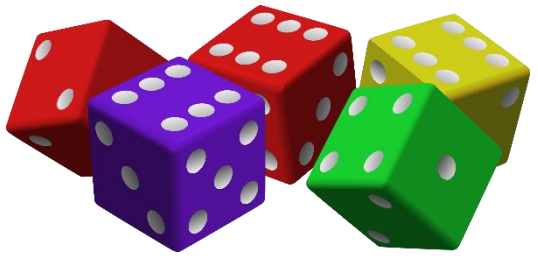
Knowing, e.g., *Offer = 1* does not affect the probability that *Sale = 1*, given we know that the class *Spam = 1*, etc.

Simple probabilistic modeling: Naïve Bayes classifier

- The Naïve Bayes independence assumption is almost never correct!!!
- But … this scheme often works surprisingly well in practice (why?)
- Easy to implement in a program and very fast

Probability

- A framework for reasoning with uncertainty
- Boolean logic expresses that statements are either necessarily true, or necessarily false
- Probability theory relaxes Boolean logic to allow for the expression of uncertainty in such statements

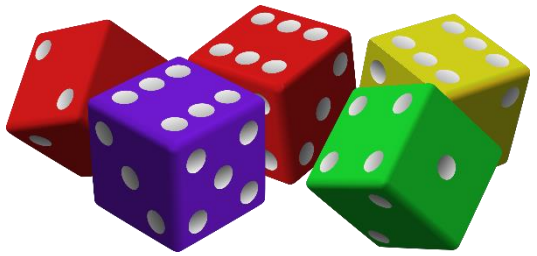


Probability

- A variable which is subject to random chance, or is uncertain, is called a **random variable**
  - E.g. *x* = the value of the roll of a die

- Each value that the random variable can take is called an **outcome**
  - E.g. for a die roll, the value *x* = 6

- **Probabilities** P(x) are values >= 0, assigned to each value of the sample space, which **sum to 1**



Conditional Probability

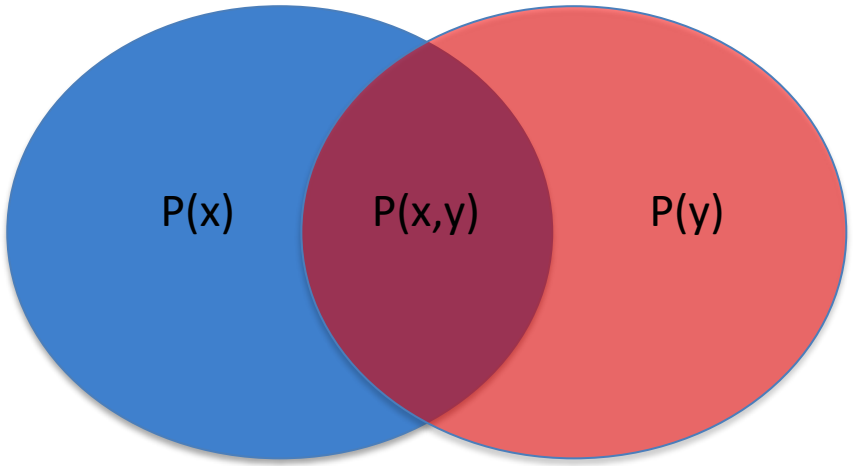
Probability of *x*, given that we know that *y* takes a particular value

Conditional probability

$$P(x=a_i \mid y=b_j) \equiv \frac{P(x=a_i, y=b_j)}{P(y=b_j)} \text{ if } P(y=b_j) \neq 0. \qquad (2.5)$$

[If $P(y=b_j)=0$ then $P(x=a_i \mid y=b_j)$ is undefined.]

We pronounce $P(x=a_i \mid y=b_j)$ 'the probability that x equals a_i, given y equals b_j'.



Bayes' rule

- **Our goal:**

  To infer the probability of a hypothesis H, given evidence E.
  This probability distribution is called the **posterior distribution,** $Pr(H|E)$

- **We have:**
  Prior beliefs about H, encoded as a probability distribution,
  the **prior distribution** $Pr(H)$

  A model for how the data were generated, given the hypothesis, the
  likelihood $Pr(E|H)$

Bayes' rule

- Bayes' rule allows us to **combine observed evidence and prior beliefs** to obtain the posterior,

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

$$Pr(H|E) = \frac{Pr(E|H)Pr(H)}{Pr(E)}$$

A normalizing constant

Deconstructing Bayes' rule

$$Pr(H|E) = \frac{Pr(H,E)}{Pr(E)}$$

Definition of conditional probability

$$Pr(H,E) = Pr(E|H)Pr(H)$$

Product rule

$$Pr(H|E) = \frac{Pr(E|H)Pr(H)}{Pr(E)}$$

Plug in equation for joint

$$Pr(E) = \sum_{H} Pr(E,H)$$

Sum rule

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables

- Given a record with attributes (A_1, A_2,…,A_n)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)

- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability P(C | A_1, A_2, …, A_n) for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    P(C | A_1, A_2, …, A_n)

  - Equivalent to choosing value of C that maximizes
    P(A_1, A_2, …, A_n|C) P(C)

- How to estimate P(A_1, A_2, …, A_n | C)?

Naïve Bayes Classifier

- Naïve assumption: evidence splits into parts (i.e., attributes) that are conditionally *independent*

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$

  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.

- This means, given *n* attributes, we can write Bayes' rule using a product of per-attribute probabilities

Probabilities for weather data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/ 14	5/ 14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Probabilities for weather data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/	5/
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5	14	14
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

$$P(yes \mid E) = P(Outlook = Sunny \mid yes)$$

$$P(Temperature = Cool \mid yes)$$

$$P(Humidity = High \mid yes)$$

$$P(Windy = True \mid yes)$$

$$P(yes) / P(E)$$

Probabilities for weather data

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/	5/
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5	14	14
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

P(Outlook=sunny|C)

P(C)

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes"|E) = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no"|E) = 0.0206 / (0.0053 + 0.0206) = 0.795

Weather data example

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

← ***Evidence E***

Probability of class "yes"

$$P(yes \mid E) = P(Outlook = Sunny \mid yes)$$

$$P(Temperature = Cool \mid yes)$$

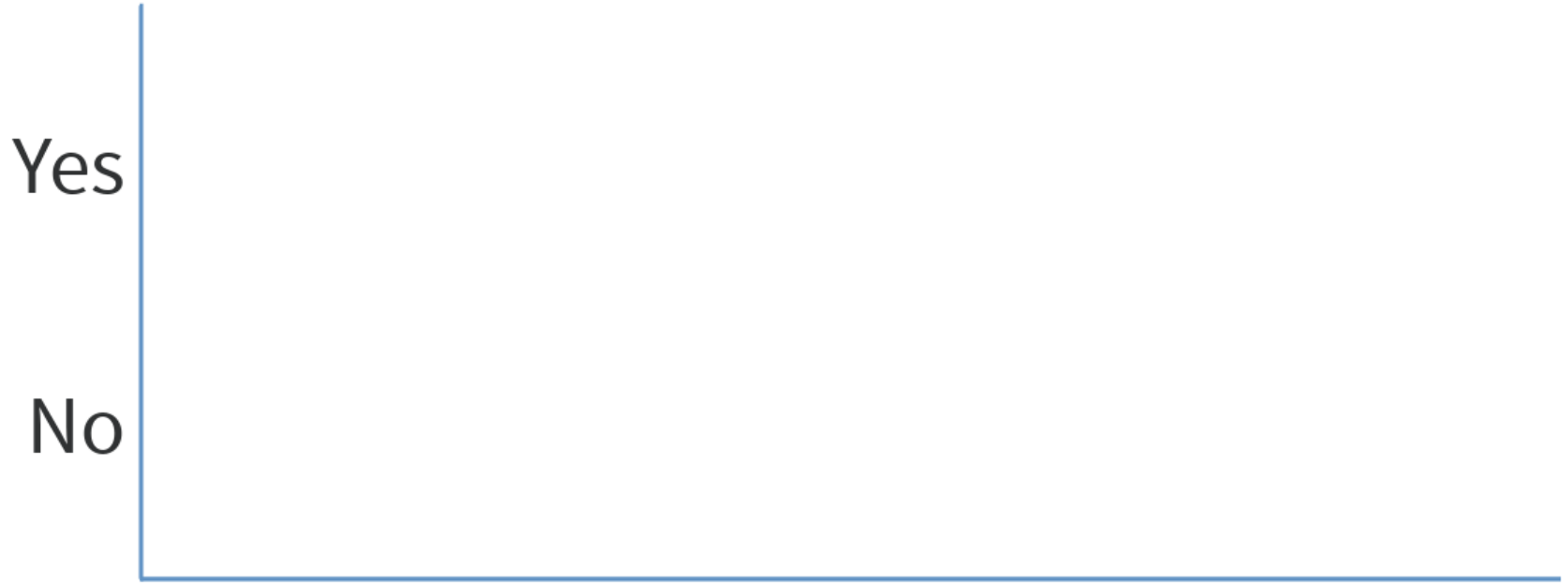
$$P(Humidity = High \mid yes)$$

$$P(Windy = True \mid yes)$$

$$P(yes) / P(E)$$

$$= \frac{2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14}{P(E)}$$

Today Outlook = Rainy, Temperature = Cool, Humidity = Normal, Windy = False. Should we play today, according to the naive Bayes classifier?



Outlook	Temp.	Humidity	Windy	Play
Rainy	Cool	Normal	False	?

← ***Evidence E***

$$P(yes \mid E) = P(Outlook = Rainy \mid yes)$$
$$P(Temperature = Cool \mid yes)$$
$$P(Humidity = Normal \mid yes)$$
$$P(Windy = False \mid yes)$$
$$P(yes)/P(E)$$

Probability of class "yes" given E

Prob. of evidence, given "yes"

Probability of class "yes"

$$= \frac{3/9 \times 3/9 \times 6/9 \times 6/9 \times 9/14}{P(E)} = 0.0317$$

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp.	Humidity	Windy	Play
Rainy	Cool	Normal	False	?

← ***Evidence E***

$$P(no \mid E) = P(Outlook = Rainy \mid no)$$
$$P(Temperature = Cool \mid no)$$
$$P(Humidity = Normal \mid no)$$
$$P(Windy = False \mid no)$$
$$P(no) / P(E)$$

Probability of class "no"

$$= \frac{2/5 \times 1/5 \times 1/5 \times 2/5 \times 5/14}{P(E)} = 0.0023$$

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp.	Humidity	Windy	Play
Rainy	Cool	Normal	False	?

← ***Evidence E***

$$P(yes \mid E) = \frac{3/9 \times 3/9 \times 6/9 \times 6/9 \times 9/14}{P(E)} = 0.0317$$

$$P(no \mid E) = \frac{2/5 \times 1/5 \times 1/5 \times 2/5 \times 5/14}{P(E)} = 0.0023$$

$$P(E) = 0.0317 + 0.0023$$

$$P(yes \mid E) = 0.0317/(0.0317 + 0.0023) = 0.93$$

Yes, we should play today!

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/	5/
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5	14	14
Rainy	3/9	2/5	Cool	3/9	1/5								

The "zero-frequency problem"

- What if an attribute value does not occur with every class value?
  (e.g., "Humidity = high" for class "yes")
  - Probability will be zero: $P(Humidity = High \mid yes) = 0$
  - *A posteriori* probability will also be zero: $P(yes \mid E) = 0$
    (Regardless of how likely the other values are!)
- Remedy: **add 1 to the count for every attribute value-class combination (*Laplace estimator)***
- Result: probabilities will never be zero
- Additional advantage: stabilizes probability estimates computed from small samples of data

Intuition: these extra counts encode "previous observations," representing prior knowledge

Laplace estimator example

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Add 1 to counts, compute probabilities using new counts. *(Must sum to one!)*

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	3	4	Hot	3	3	High	4	5	False	7	3	10	6
Overcast	5	1	Mild	5	3	Normal	7	2	True	4	4		
Rainy	4	3	Cool	4	2								
Sunny	3/12	4/8	Hot	3/12	3/8	High	4/11	5/7	False	7/11	3/7	10/16	6/16
Overcast	5/12	1/8	Mild	5/12	3/8	Normal	7/11	2/7	True	4/11	4/7		
Rainy	4/12	3/8	Cool	4/12	2/8								

Missing values

- Training: instance is not included in frequency count for attribute value-class combination
- Classification: attribute will be omitted from calculation
- Example:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Likelihood of "yes" = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

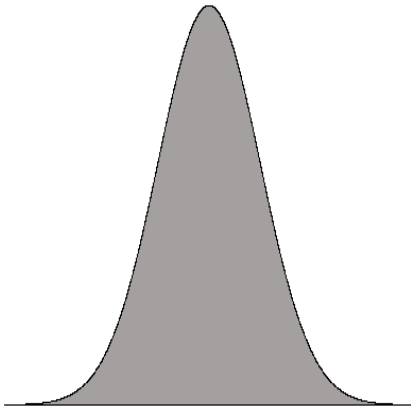
Likelihood of "no" = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

P("yes"|E) = 0.0238 / (0.0238 + 0.0343) = 41%

P("no"|E) = 0.0343 / (0.0238 + 0.0343) = 59%

Numeric attributes

- Usual assumption: attributes have a *normal* or *Gaussian* probability distribution (given the class)
- The *probability density function* for the normal distribution is defined by two parameters:
  - *Sample mean* $$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$
  - *Standard deviation* $$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu)^2}$$
  - Then the density function *f(x) is* $$f(x) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Statistics for weather data

Outlook			Temperature		Humidity		Windy			Play	
	Yes	*No*	*Yes*	*No*	*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	64, 68,	65,71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72,80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, …	85, …	80, …	95, …					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/	5/
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5	14	14
Rainy	3/9	2/5									

- Example density value:

$$f(temperature = 66|yes) = \frac{1}{\sqrt{2\pi}\cdot 6.2} e^{-\frac{(66-73)^2}{2\cdot 6.2^2}} = 0.0340$$

Classifying a new day

- A new day:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of "yes" = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of "no" = $3/5 \times 0.0221 \times 0.0381 \times 3/5 \times 5/14 = 0.000108$

P("yes") = 0.000036 / (0.000036 + 0. 000108) = 25%

P("no") = 0.000108 / (0.000036 + 0. 000108) = 75%

- Missing values during training are not included in calculation of mean and standard deviation

Naïve Bayes: discussion

- Naïve Bayes works surprisingly well even if independence assumption is clearly violated

- Why? Because classification does not require accurate probability estimates ***as long as maximum probability is assigned to the correct class***

- However: adding too many redundant attributes will cause problems (e.g., identical attributes)

Naïve Bayes

- **Representation**
  - Probabilistic model, with naïve Bayes conditional independence assumption
- **Objective function for training**
  - Log probability of training data under the model, log Pr(C,A).
- **Search algorithm**
  - Estimate all probabilities required for Bayes rule, under naïve Bayes assumption, according to frequencies on training data

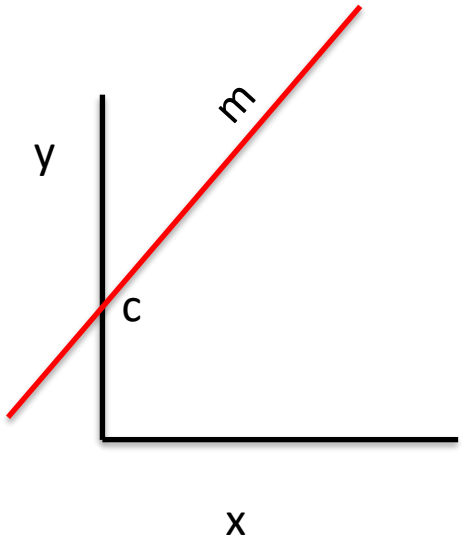
Linear functions

- Remember the equation for a line?

$$y = mx + c$$

slope

intercept term



Linear functions

- Remember the equation for a line?

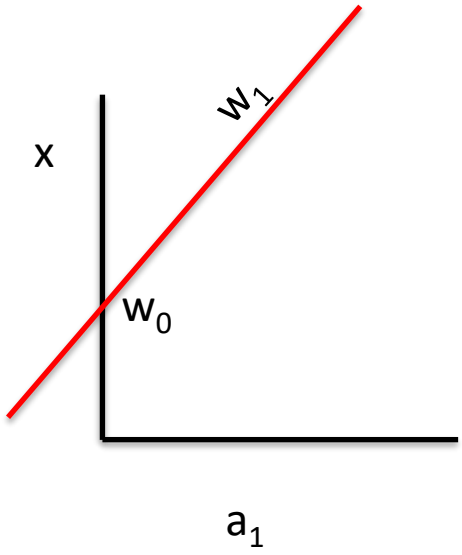
$$y = mx + c$$

slope

intercept term

Different notation for machine learning:

$$x = w_1 a_1 + w_0$$



Linear functions

- Remember the equation for a line?

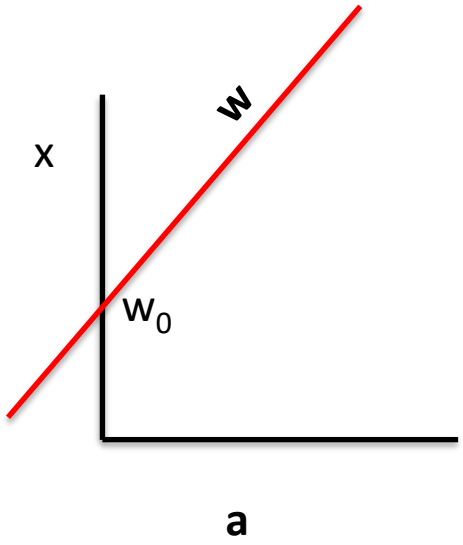
$$y = mx + c$$

slope

intercept term

Different notation for machine learning:

$$x = w_1 a_1 + w_0$$



Generalize to multiple attributes:

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k$$

Linear models: linear regression

- Work most naturally with numeric attributes
- Standard technique for numeric prediction
  - Outcome is linear combination of attributes

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k$$

- Weights are calculated from the training data

Linear models: linear regression

- We can simplify and get rid of the intercept term w_0 by assuming each instance is extended with a constant attribute with value 1.

- Predicted value for first training instance $\mathbf{a}^{(1)}$:

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \cdots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

Minimizing the squared error

- Choose *k* +1 coefficients to minimize the squared error on the training data
- Squared error: $$\sum_{i=1}^{n}\left(x^{(i)}-\sum_{j=0}^{k}w_j a_j^{(i)}\right)^2$$

- Coefficients can be derived using standard matrix operations
  - Can be done if there are more instances than attributes (roughly speaking)
  - Minimizing *absolute error* is more difficult, but can be done

Classification

- *Any* regression technique can be used for classification
  - Training: perform a regression for each class, setting the output to 1 for training instances that belong to class, and 0 for those that don't
  - Prediction: predict class corresponding to model with largest output value (*membership value*)
- For linear regression this method is also known as ***multi-response linear regression***

Classification

- *Any* regression technique can be used for classification
  - Training: perform a regression for each class, setting the output to 1 for training instances that belong to class, and 0 for those that don't
  - Prediction: predict class corresponding to model with largest output value (*membership value*)
- For linear regression this method is also known as ***multi-response linear regression***
- Problem: membership values are not in the [0,1] range, so they cannot be considered proper probability estimates
  - In practice, they are often simply clipped into the [0,1] range and normalized to sum to 1

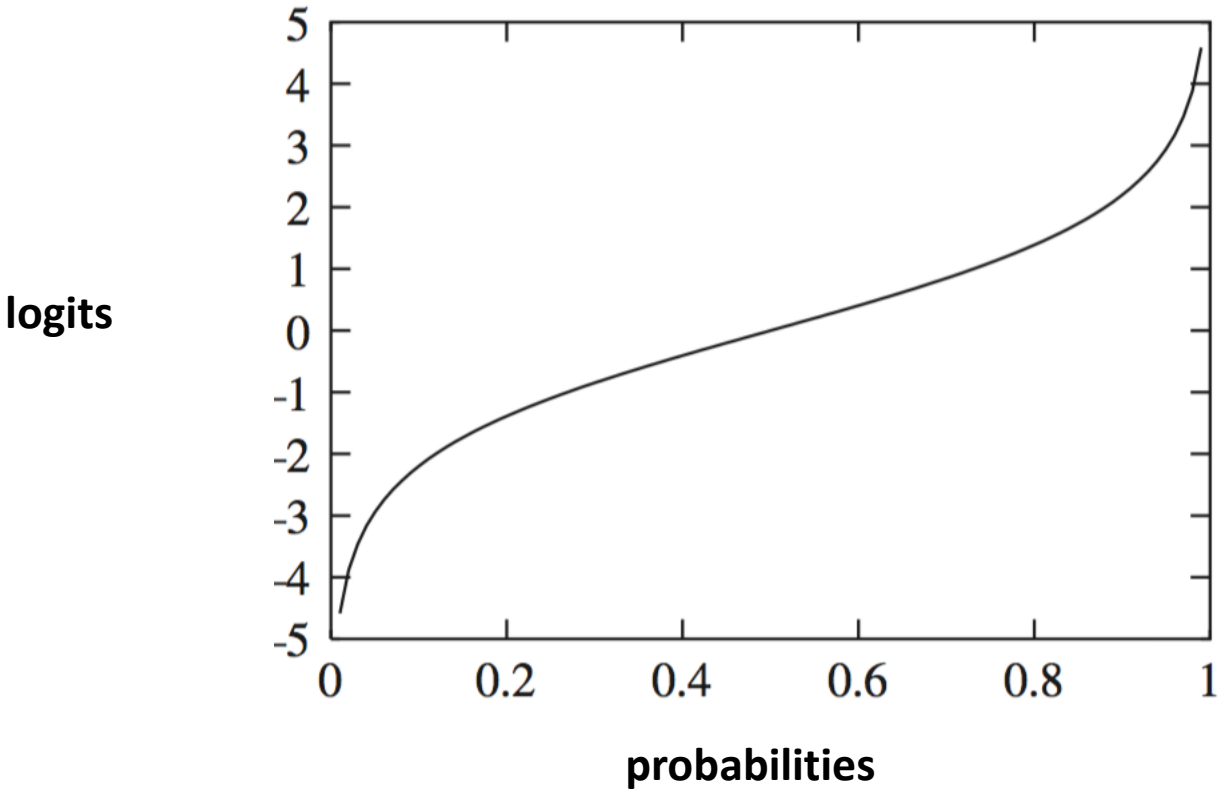
Linear models: logistic regression

- Can we do better than using linear regression for classification?
- Yes, we can, by applying **logistic regression**:
  - Builds a **linear model for a transformed target variable**
- Assume we have two classes *(multi-class also possible)*
- Logistic regression replaces the target $\Pr[1|a_1, a_2, \ldots, a_k]$

  by this target: $\log[\Pr[1|a_1, a_2, \ldots, a_k]/(1 - \Pr[1|a_1, a_2, \ldots, a_k])$

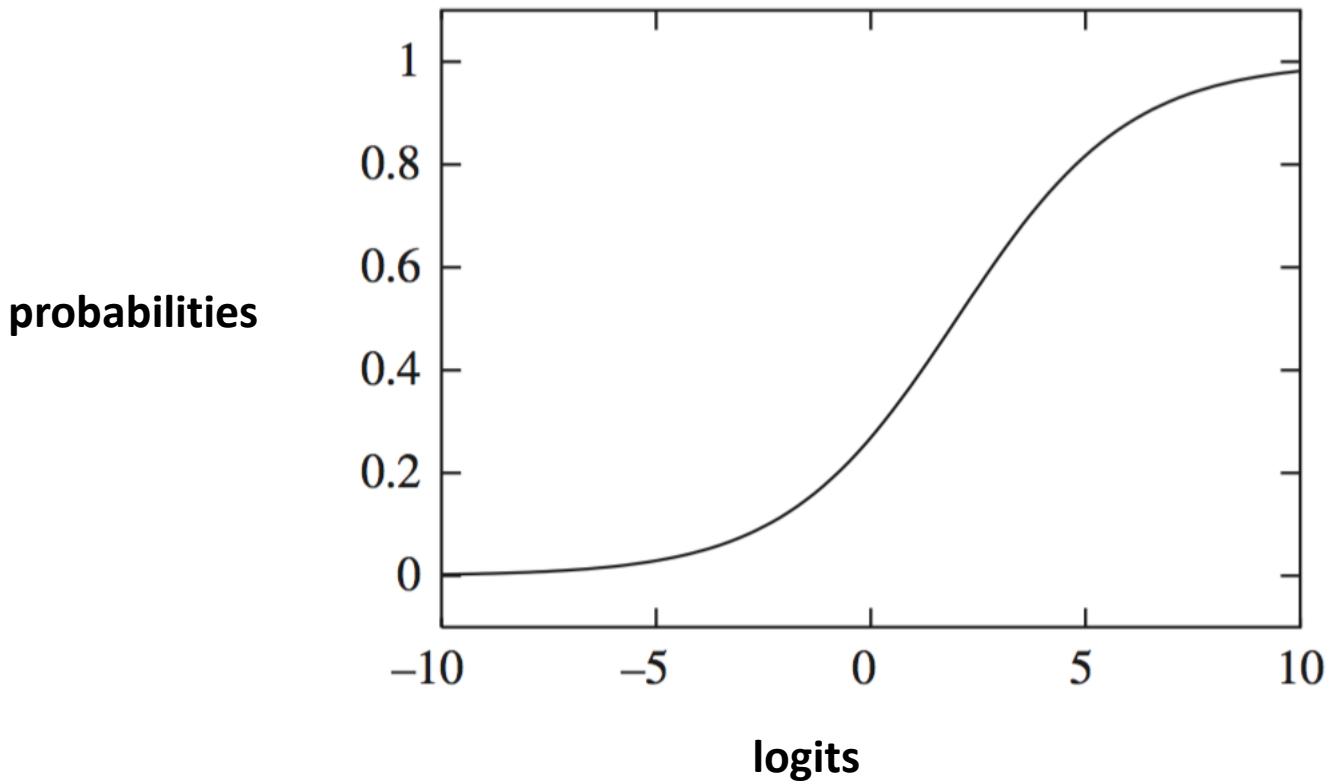
- This ***logit transformation*** maps [0,1] to (-∞ , +∞), i.e., the new target values are no longer restricted to the [0,1] interval

Logit transformation



- Probabilities, between 0 and 1, are corresponded with numbers anywhere on the real line
- To obtain probabilities from the regression, we reverse this mapping

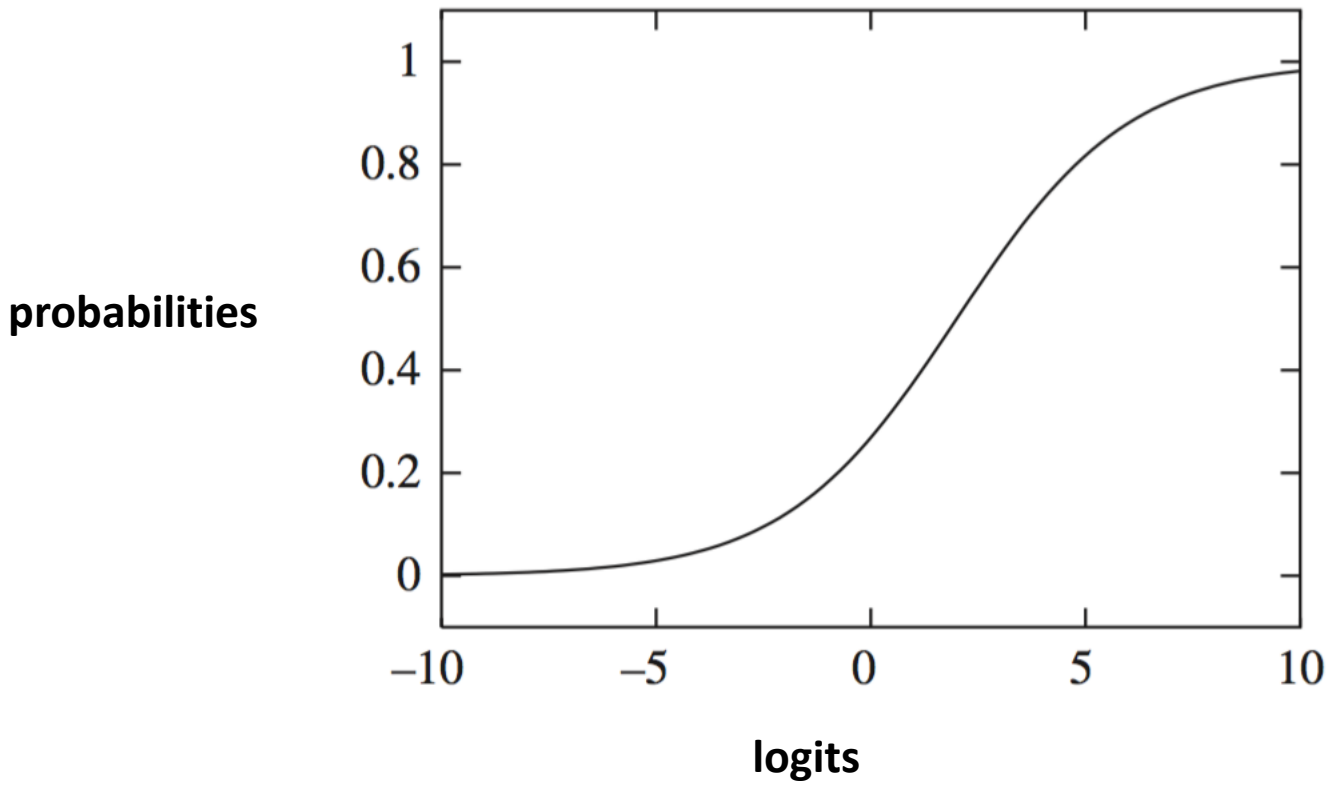
The inverse logit function



- Aka the **logistic function**, or the **sigmoid function** ("S" shaped).
- "Squashes" real numbers to probabilities

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Logistic regression model



$$p(c = 1|a_1, a_2, \ldots, a_n) = \sigma(w_0 + w_1 a_1 + \ldots + w_n a_n)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

linear model

"squashes" reals to probabilities

Maximum likelihood estimation

- Aim: **maximize probability of observed training data** with respect to final parameters of the logistic regression model

- We can use logarithms of probabilities and maximize conditional *log-likelihood* instead of product of probabilities:

$$\sum_{i=1}^{n}(1-x^{(i)})\log(1-\Pr[1|a_1^{(i)},\ a_2^{(i)},\ \ldots,\ a_k^{(i)}])+x^{(i)}\log(\Pr[1|a_1^{(i)},a_2^{(i)},\ldots,a_k^{(i)}])$$

  where the class values $x^{(i)}$ are either 0 or 1

- **Weights w_i need to be chosen to maximize log-likelihood**

Maximum likelihood estimation

- Training Algorithms:
  - Iteratively re-weighted least squares (**IRLS**). *E.g. Matlab's mnrfit function*
    - Until converged
      - Approximate objective as a (weighted) least squares linear regression problem, based on current guess
      - Solve least squares problem, update parameter estimates
    - Converges in few iterations, but each iteration is $O(k^3)$, due to a matrix inversion
    - Equivalent to Newton's method
  - Quasi-Newton methods.
    - Approximate the above, faster per iteration
    - **BFGS**. Approximate $O(k^3)$ step with a diagonal matrix. *E.g. WEKA's functions -> Logistic*
    - **Truncated Newton**. Iterative procedure to approximate Newton step. E.g. *LIBLINEAR*
  - Stochastic gradient descent (SGD)
    - Process 1 data point at a time, so low memory requirements.
    - Used for large-scale learning at Google. *WEKA's functions -> SGD*

Multiple classes

- What do we do when have a problem with *k* classes?
- Could perform logistic regression independently for each class (like in multi-response linear regression)

- **Problem**: the probability estimates for the different classes will generally not sum to one

Multiple classes

- A better approach: train *k*-1 coupled linear models by maximizing likelihood over all classes simultaneously

- The linear models' outputs are mapped to probabilities over all k classes, using the **softmax function**

- This is known as **multi-class logistic regression**, a.k.a. **multinomial logistic regression**

Linear models are hyperplanes

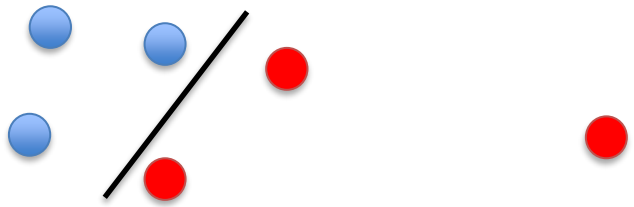
- Decision boundary for two-class logistic regression is where probability equals 0.5:

$$\Pr[1|a_1, a_2, \ldots, a_k] = 1/(1 + \exp(-w_0 - w_1 a_1 - \cdots - w_k a_k)) = 0.5$$

which occurs when $-w_0 - w_1 a_1 - \cdots - w_k a_k = 0$

- **Thus logistic regression can only separate data that can be separated by a hyperplane**

- Multi-response linear regression has the same problem.



Logistic Regression

- **Representation**
  - Linear model. Logistic function maps a linear function to probabilities.
- **Objective function for training**
  - Log probability of class labels under the model, log Pr(C|A).
- **Search algorithm**
  - Iteratively reweighted least squares, or gradient-based methods

Relationship between naïve Bayes and logistic regression

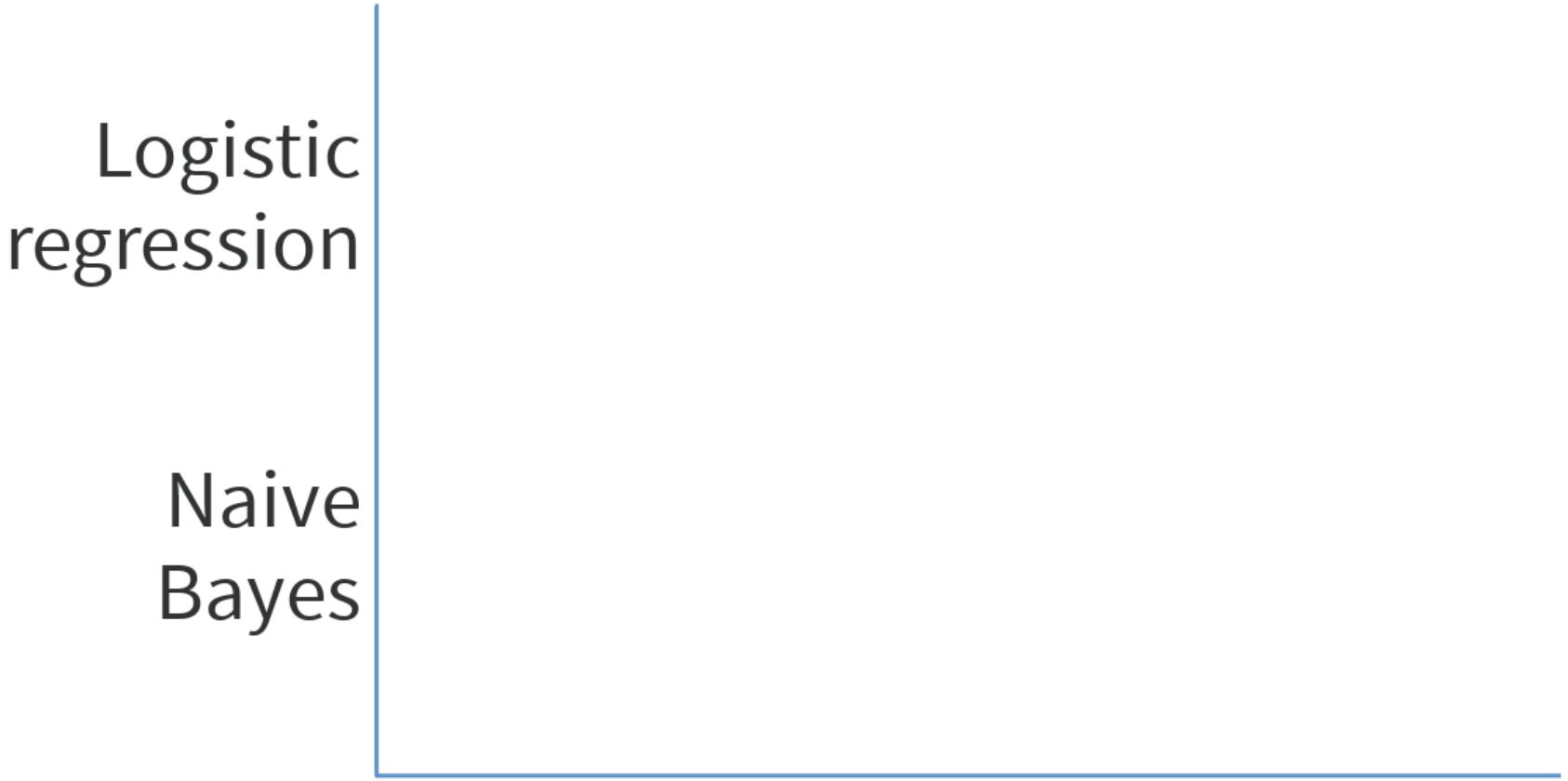
- Both naïve Bayes and logistic regression are trained to maximize the log-likelihoood
- Naïve Bayes: a generative classifier
  - Models the attributes as well as the class
  - Maximize log P(C,A) = log P(A|C) + log P(C) = log P(C|A) + log P(A)

- Logistic regression: A discriminative classifier
  - Models the class given the attributes, but not attributes
    Maximize log P(C|A)

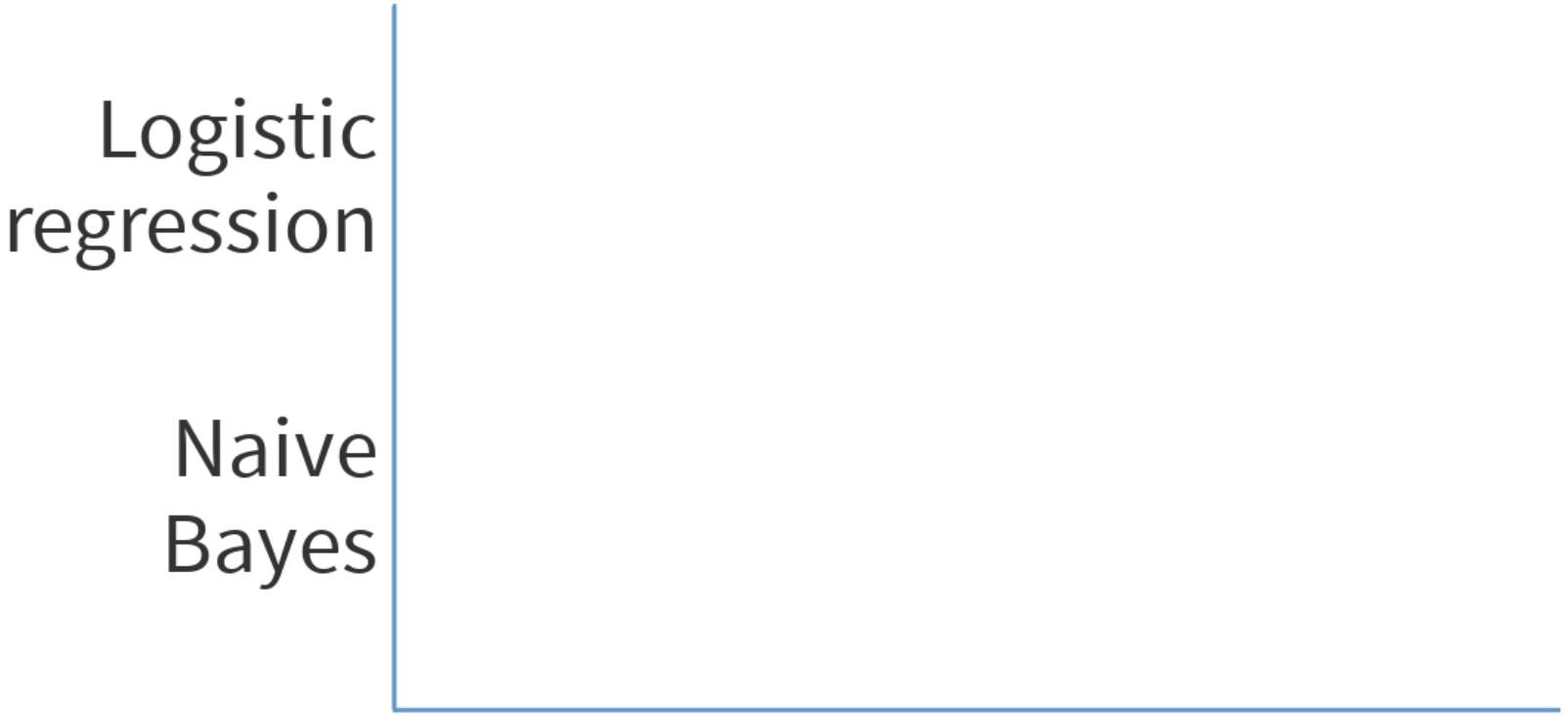
- Naïve Bayes, trained to maximize log P(C|A), is equivalent to logistic regression! (Both with Gaussian assumption, or discrete data. Proof on bonus slide at end)

- Naïve Bayes and logistic regression are a "generative-discriminative pair"

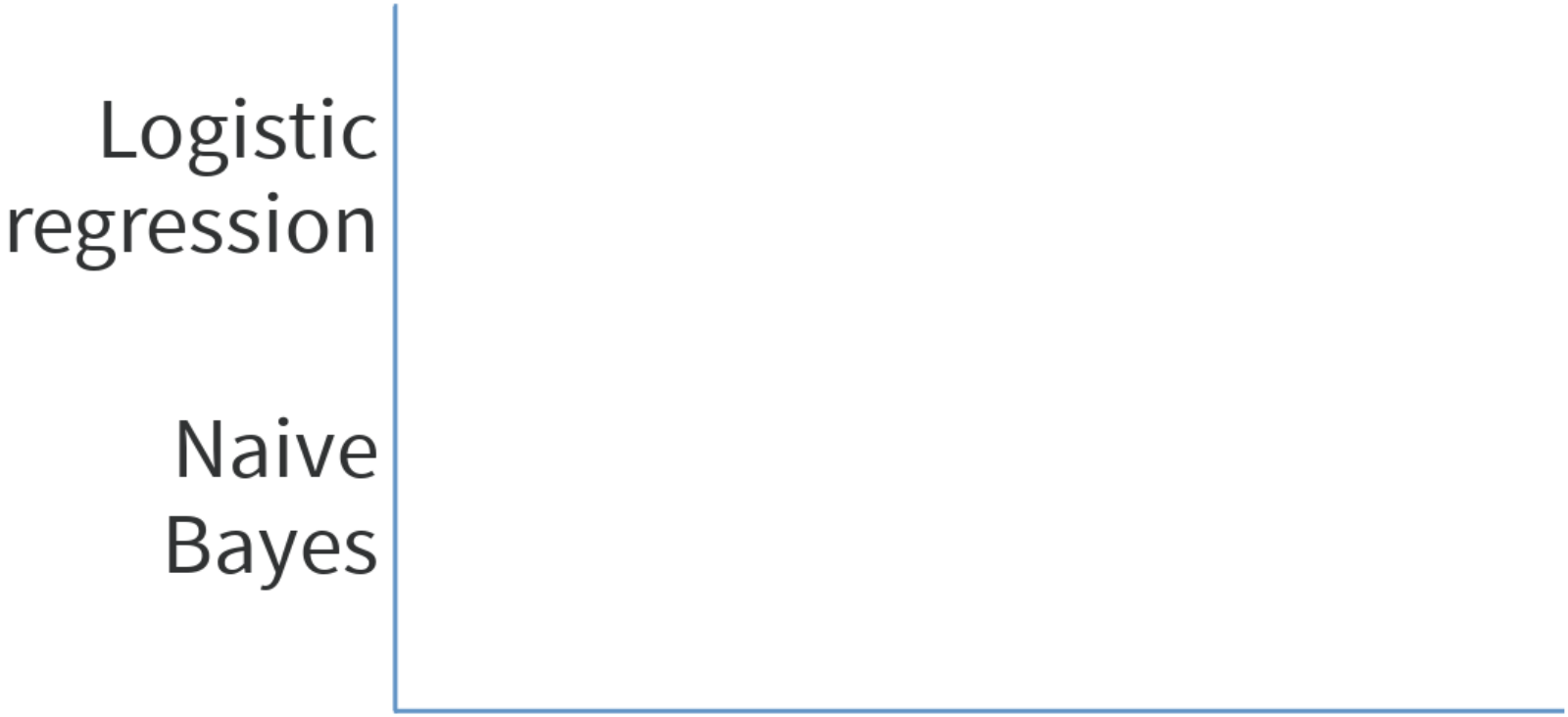
Which can be trained more computationally efficiently on a large dataset?



Which is typically more accurate at predicting held-out data, when trained on a very large dataset?



Which is typically more accurate at predicting held-out data, when trained on a small dataset?



Intuition

- Naïve Bayes needs to model the instances P(A|C)
  - This requires strong assumptions
  - These assumptions *make the estimates stable for small data*, but *limit the achievable performance* with a lot of data

Intuition

- The reverse is true for logistic regression, which only needs to model the classes given the instances, P(C|A)

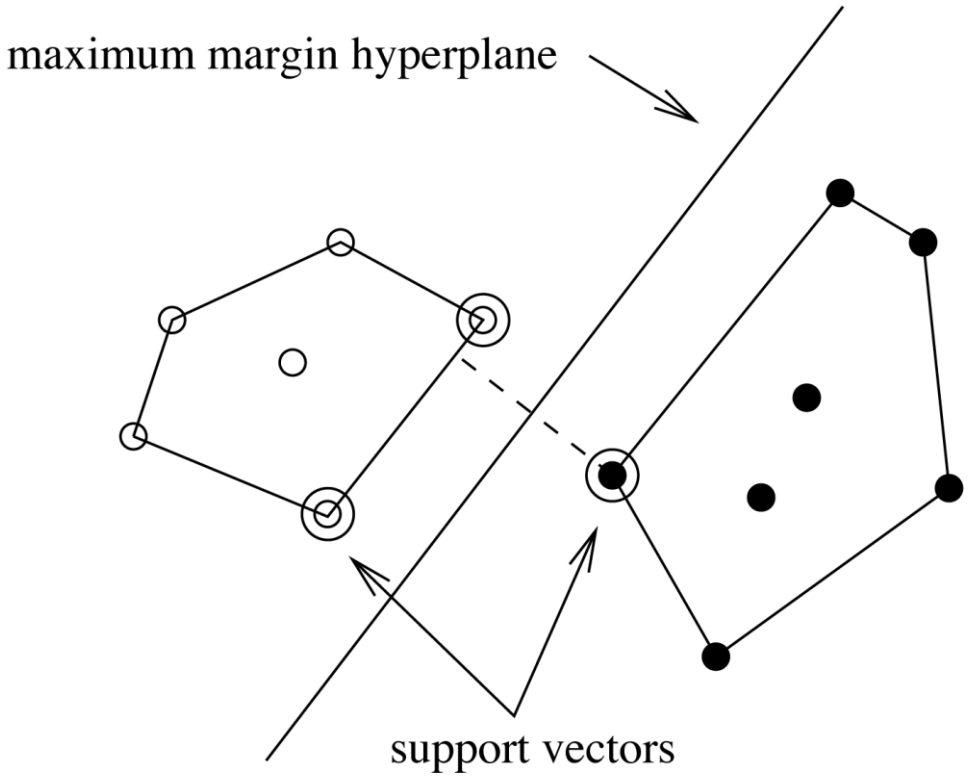
- This is called the **bias-variance trade-off**
  - more next lesson!
  - These intuitions hold more generally for generative and discriminative models.

Support vector machines

- ***Support vector machines*** are algorithms for learning linear classifiers
- Resilient to overfitting because they learn a particular linear decision boundary:
  - The ***maximum margin hyperplane***

- They can also **learn non-linear classifiers**, using a certain "trick"
  - Use a mathematical trick to avoid creating "pseudo-attributes"
  - The nonlinear space is created implicitly

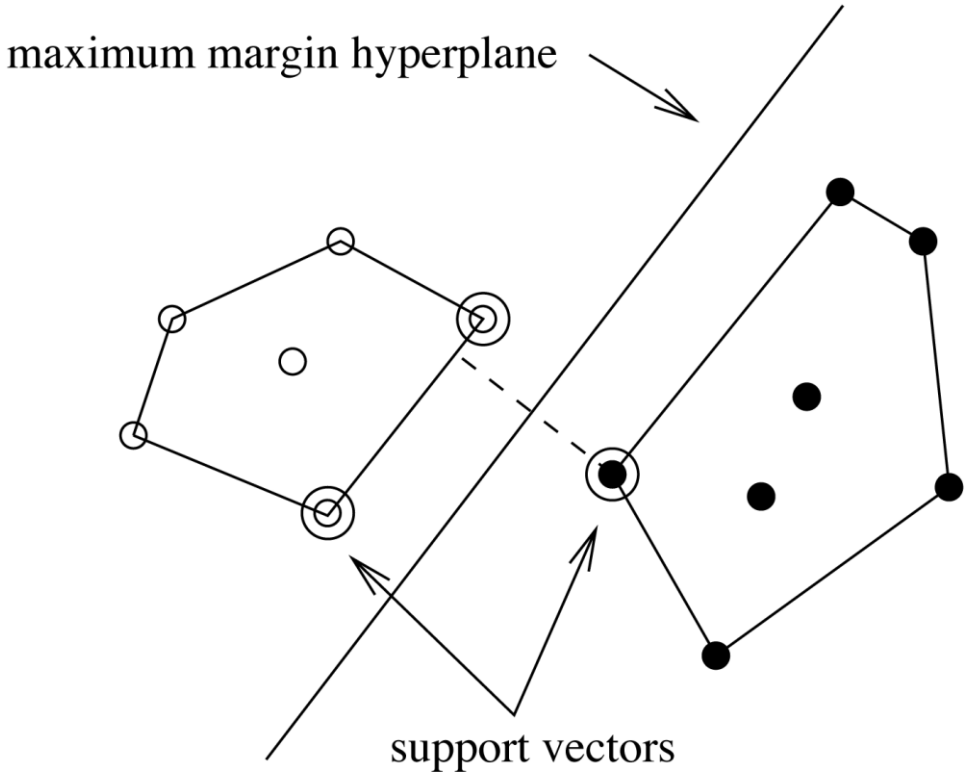
The maximum margin hyperplane



- The instances closest to the maximum margin hyperplane are called *support vectors*

Support vectors

- The support vectors define the maximum margin hyperplane
- All other instances can be deleted without changing its position and orientation



- The hyperplane

$$x = w_0 + w_1 a_1 + w_2 a_2$$

can be written as

$$x = b + \sum_{i \text{ is a supp. vector}} \alpha_i y_i \vec{a}(i) \cdot \vec{a}$$

Finding support vectors

$$x = b + \sum_{i \text{ is a supp. vector}} \alpha_i y_i \vec{a}(i) \cdot \vec{a}$$

- Support vector: training instance for which $\alpha_i > 0$
- Determining α_i and *b* ?

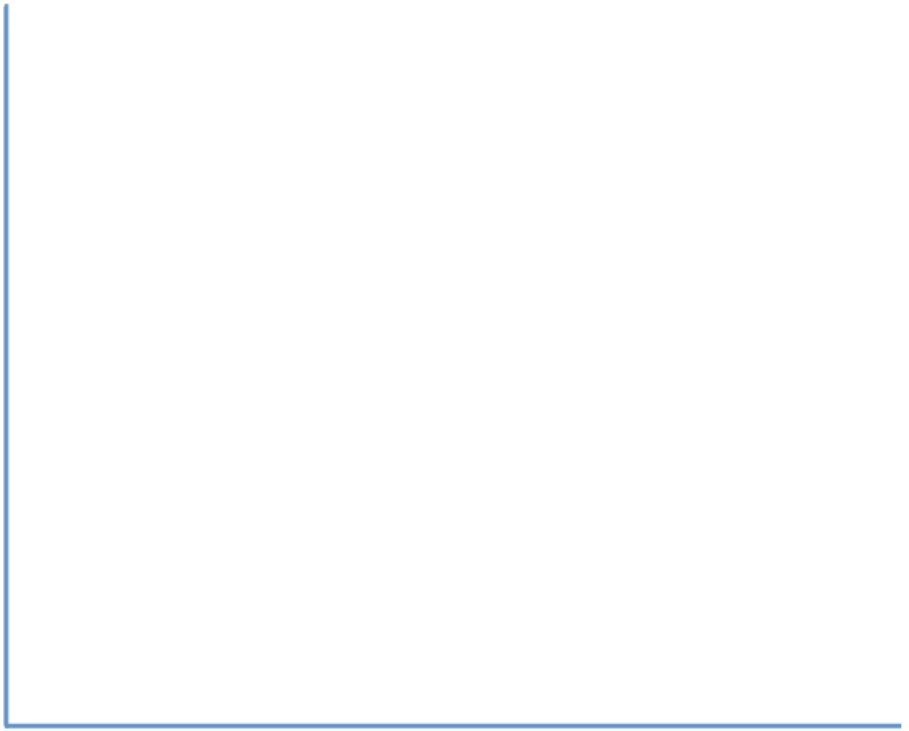
  A *constrained quadratic optimization* problem
  - Off-the-shelf tools for solving these problems
  - However, special-purpose algorithms are faster
  - Example: Platt's *sequential minimal optimization* (SMO) algorithm

Which of the following would allow you to create classifier with a non-linear decision boundary?

Use a linear classifier on a modified set of instances, where we add extra features $a_1 a_2, a_1 a_3, \ldots a_{d-1} a_d$

Apply the logit transformation to map real-valued predictions from a linear regression model into class probabilities

Train a modified version of naive Bayes for numeric data, which optimizes $\log P(C|A)$ instead of $\log P(A, C)$



Nonlinear SVMs

- We can create a nonlinear classifier by creating new "pseudo" attributes from the original attributes in the data
  - "Pseudo" attributes represent attribute combinations
  - E.g.: all polynomials of degree 2 that can be formed from the original attributes

- The linear SVM in the extended space is a non-linear classifier in the original attribute space
- **Overfitting** often **not** a significant problem with this approach because the maximum margin hyperplane is stable
  - There are often comparatively few support vectors relative to the size of the training set
- **Computation time still an issue**

A mathematical trick

- Avoid computing the "pseudo attributes"
- Compute the dot product before doing the nonlinear mapping
- Example: $$x = b + \sum_{i \text{ is a supp. vector}} \alpha_i y_i (\vec{a}(i) \cdot \vec{a})^n$$

- Corresponds to a map into the instance space spanned by all products of *n* attributes

Other kernel functions

- Mapping is called a "kernel function"
- Polynomial kernel $$x = b + \sum_{i \text{ is a supp. vector}} \alpha_i y_i (\vec{a}(i) \cdot \vec{a})^n$$
- We can use others: $$x = b + \sum_{i \text{ is a supp. vector}} \alpha_i y_i K(\vec{a}(i) \cdot \vec{a})$$
- Only requirement: $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$

K() can be written as a dot product in a feature space create by the implicit feature mapping *Φ*()

Noise

- Have assumed that the data is separable (in original or transformed space)
- Can apply SVMs to noisy data by introducing a "noise" parameter *C*
  - Also known as *regularization* parameter
- *C* bounds the influence of any one training instance on the decision boundary
  - Based on the following constraint: $0 \leq \alpha_i \leq C$
- A "soft" margin is maximized based on this constraint

Support Vector Machines (SVMs)

- **Representation**
  - Linear model. The “kernel trick” implicitly maps instances to higher-dimensional spaces, leading to non-linear decision boundaries
- **Objective function for training**
  - The margin (distance from hyperplane to closest instances for each class)
- **Search algorithm**
  - Quadratic optimization, often solved by special-purpose algorithms

Think-pair-share: Naïve Bayes Assumption

- The Naïve Bayes classifier assumes that the attributes are independent, given the class: knowing the value of one attribute will not inform you of the others (if you know the class). Is this assumption valid for the following?
  - Class: **acceptance to UMBC IS graduate program** (yes/no).
    Attributes: GRE verbal, quantitative, and writing scores
  - Class: the **number of coins coming up heads is even**, from two coin tosses (yes/no).
    Attributes: Coin 1 (heads/tails), Coin 2 (heads/tails)
  - Class: **play sports game today**? (yes/no).
    Attributes: Outlook, temperature, humidity, wind (all nominal)

Can you think of another scenario where the assumption is valid, and another where it isn't?

Bonus slide: Naïve Bayes is a linear model, Generative version of logistic regression

$$
\begin{aligned}
P(C=1|A) &= \frac{P(A|C=1)P(C=1)}{P(A)} \\
&= \frac{P(A|C=1)P(C=1)}{P(A|C=1)P(C=1)+P(A|C=0)P(C=0)} \\
&= \frac{1}{1+\frac{P(A|C=0)P(C=0)}{P(A|C=1)P(C=1)}} \\
&= \frac{1}{1+\exp(-\log\frac{P(A|C=1)P(C=1)}{P(A|C=0)P(C=0)})} \\
&= \sigma\Big(\log\frac{P(A|C=1)P(C=1)}{P(A|C=0)P(C=0)})\Big) \\
&= \sigma\Big(\sum_{i=1}^{k}\log\frac{P(A_i|C=1)}{P(A_i|C=0)}+\log\frac{P(C=1)}{P(C=0)}\Big)
\end{aligned}
$$

Linear if discrete, or Gaussian.

So naïve Bayes is equivalent to logistic regression, trained generatively!