Artificial Intelligence 2

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Introduction

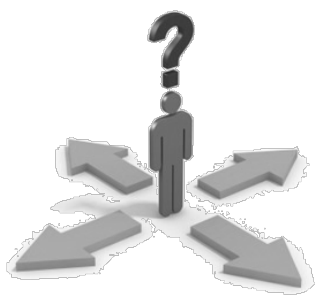
There is prevalence of **uncertainty** in real environments.

Agents can handle uncertainty by using the methods of **probability**, such as Bayesian networks.

How can we learn the proper model (how to fill up the conditional probability tables)?

We will described methods for learning probability models

- statistical (Bayesian) learning
  - calculating the probability of each hypothesis
- learning naïve Bayes models
- learning with hidden variables



Model example

Consider candy coming in two flavors – cherry and lime – that the manufacturer wraps in the same opaque wrapper.

The candy is sold in very large bags, of which there are known to be five kinds:

h_1: 100% cherry
h_2: 75% cherry + 25% lime
h_3: 50% cherry + 50% lime
h_4: 25% cherry + 75% lime
h_5: 100% lime

The **random variable** H (for hypothesis) denotes the type of the bag (H is not directly observable).

As the pieces of candy are opened and inspected, **data** are revealed $D_1,\ldots,D_N$, where each D_i is a random variable with possible values cherry and lime.

The basic task is to **predict the flavor of the next piece of candy**.

Bayesian learning

It calculates the probability of each hypothesis, given the data and makes predictions on that basis.

The predictions are made by **using all the hypothesis**, weighted by their probabilities, rather than by using just a single "best" hypothesis.

Formally $P(h_i|\mathbf{d}) = \alpha\ P(\mathbf{d}|h_i)\ P(h_i)$

where **d** are the observed values

A prediction about an unknown quantity X is made using:

$P(X|\mathbf{d}) = \Sigma_i\ P(X|\mathbf{d},h_i).\ P(h_i|\mathbf{d}) = \Sigma_i\ P(X|h_i).\ P(h_i|\mathbf{d})$

Predictions are weighted averages over the predictions of the individual hypothesis.

The hypotheses themselves are "intermediaries" between the raw data and the predictions.

The key quantities in the Bayesian approach are:

- the hypothesis prior $P(h_i)$
- the likelihood of the data under each hypothesis $P(\mathbf{d}|h_i)$

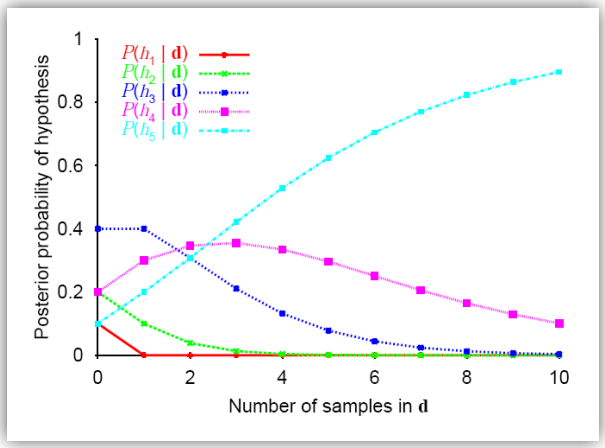
So which candy is next?

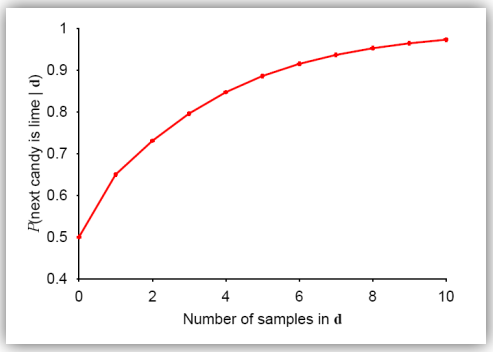
Let the prior distribution over the hypotheses space is given ⟨0.1; 0.2; 0.4; 0.2; 0.1⟩

Under the assumption of independent and identically distributed samples (big bags):

$P(\mathbf{d}|h_i) = \Pi_j P(d_j|h_i)$

After 10 lime candies in a row we get $P(\mathbf{d}|h_3) = 0.5^{10}$.





Properties of Bayesian learning

The Bayesian prediction eventually **agrees** with the the hypothesis.

The posterior probability of any false hypothesis will eventually vanish.

The Bayesian prediction is **optimal**, whether the data set be small or large.

Any other prediction is expected to be correct less often.

For real learning problems, the **hypothesis space is usually very large** or infinite.

We must resort to approximate or simplified methods.

MAP learning

A very common approximation is to make predictions based on a single most probable hypothesis (**maximum a posteriori hypothesis**).

$P(X|d) \approx P(X|h_{MAP})$

In our candy example, after three lime candies in a row, we get $h_{MAP} = h_5$.

It predicts that the fourth candy is lime with probability 1.0 – a much more dangerous prediction than the Bayesian prediction of 0.8.

Finding MAP hypotheses requires **solving an optimization problem** instead of a large summation (or integration) problem.

Overfitting

We saw that **overfitting** can occur when the hypothesis space is too expressive, so that it contains many hypotheses that fit the data set well.

Bayesian and MAP learning methods use the prior to **penalize complexity**.

- more complex hypotheses have a lower prior probability
- the hypothesis prior embodies a tradeoff between the complexity of a hypothesis and its degree of fit to the data

If H contains only deterministic hypotheses ($P(\mathbf{d}|h_i) = 1$, if h_i is consistent and 0 otherwise) then h_{MAP} is the simplest logical theory that is consistent with data.

Therefore, MAP learning provides a natural embodiment of Ockham's razor.

MDL learning

Choosing h_{MAP} means to maximize $P(\mathbf{d}|h_i)\ P(h_i)$

This is equivalent to minimizing "$-\log_2 P(\mathbf{d}|h_i) - \log_2 P(h_i)$"

- "$-\log_2 P(h_i)$" term equals the number of bits required to specify the hypothesis h_i
- "$-\log_2 P(\mathbf{d}|h_i)$" is the additional number of bits required to specify the data, given the hypothesis (consider that no bits are required if the hypothesis predicts the data exactly $-\log_2 1 = 0$)

Hence, MAP learning is choosing the hypothesis that provides maximum compression of data.

The same task is addressed more directly by the **minimum description length** (MDL) learning method.

- MDL counts the bits in a binary encoding of the hypotheses and data and looks for hypothesis with the smallest number of bits.

ML learning

A final simplification is provided by assuming a **uniform prior** over the space of hypotheses.

In that case, MAP learning reduces to choosing an h_i that maximizes $P(\mathbf{d}|h_i)$. This is called **maximum-likelihood (ML) hypothesis.**

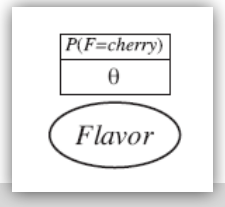
- suppresses the subjective nature of hypothesis priors
- it provides a good approximation to Bayesian and MAP learning when the data set is large (data swamps the prior distribution over hypotheses)
- it has a problem with small data sets

How to find the numerical parameters for a probability model whose structure is fixed?

- assume a Bayesian network with a given structure
- we are interested in learning the conditional probabilities – **parameter learning**
- let us assume that we have **complete data**
  - each data point contains values for every variable in the probability model being learned

Suppose we buy a bag of lime and cherry candy from a new manufacturer whose lime-cherry proportions are completely unknown.

- The parameter θ in this case is the proportion of cherry candies.
  - hypothesis is h_θ
  - the Bayesian network consists of a single node



- After unwrapping N candies, of which c are cherries and l (= N-c) are limes, the likelihood of this data is:

$$P(\mathbf{d}|h_\theta) = \Pi_j P(d_j|h_\theta) = \theta^c (1-\theta)^l$$

- Application of ML learning is appropriate there as prior probabilities of all hypotheses are identical.
  - maximizing $P(\mathbf{d}|h_\theta)$ is the same as maximizing

$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \Sigma_j \log P(d_j|h_\theta) = c \log\theta + l \log(1-\theta)$$

  - We differentiate L with respect to θ and set the resulting expression to zero:

$$\partial L(\mathbf{d}|h_\theta) / \partial\theta = c/\theta - l/(1-\theta) = 0 \quad \Rightarrow \quad \theta = c/(c+l) = c/N$$

  - h_{ML} asserts that the actual proportion of cherries in the bag is equal to the observed proportion in the candies unwrapped so far!

Maximum-likelihood parameter learning

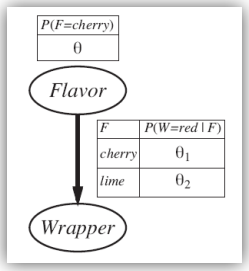
The standard method for maximum-likelihood parameter learning:

- write down an expression for the likelihood of the data as a function of the parameter(s)
- write down the derivative of the log likelihood with respect to each parameter
- find the parameter values such that the derivatives are zero
  - this is the trickiest step and in many cases we need to resort to iterative solution algorithms or other numerical optimization techniques

ML parameter learning problem for a Bayesian network decomposes into separate learning problems, one for each parameter.

Extended example

Suppose the candy manufacturer wants to give a little hint to the consumer and uses candy wrappers colored red and green (the wrapper for each candy is selected probabilistically).



- the Bayesian network now has three parameters $\theta, \theta_1, \theta_2$
  - θ - cherry candy
  - θ_1 – cherry candy has a red wrapper
  - θ_2 – lime candy has a red wrapper

$P(\text{Flavor=cherry, Wrapper=green} \mid h_{\theta,\theta_1,\theta_2})$

$= P(\text{Flavor=cherry} \mid h_{\theta,\theta_1,\theta_2}) \, P(\text{Wrapper=green} \mid \text{Flavor=cherry}, h_{\theta,\theta_1,\theta_2})$

$= \theta \, (1-\theta_1)$

We unwrap N candies
(c cherries, l limes,
r_c cherries with red wrappers, g_c cherries with green wrappers,
r_l lime with red wrappers, g_l lime with green wrappers)

$$P(\mathbf{d} \mid h_{\theta,\theta_1,\theta_2}) = \theta^c (1-\theta)^l \, \theta_1^{r_c} (1-\theta_1)^{g_c} \, \theta_2^{r_l} (1-\theta_2)^{g_l}$$

$$L = c.\log \theta + l.\log(1-\theta) + r_c.\log \theta_1 + g_c.\log(1-\theta_1) + r_l.\log \theta_2 + g_l.\log(1-\theta_2)$$

$$\partial L / \partial \theta = c / \theta - l / (1-\theta) \quad \Rightarrow \quad \theta = c / (c+l)$$

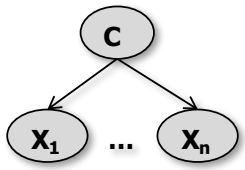
$$\partial L / \partial \theta_1 = r_c / \theta_1 - g_c / (1-\theta_1) \quad \Rightarrow \quad \theta_1 = r_c / (r_c + g_c)$$

$$\partial L / \partial \theta_2 = r_l / \theta_2 - g_l / (1-\theta_2) \quad \Rightarrow \quad \theta_2 = r_l / (r_l + g_l)$$

Naive Bayes models

Probably the most common Bayesian network model used in machine learning is the **naïve Bayes model**.

- the "class" variable C (which is to be predicted) is the root
- the "attribute" variables X_i are the leaves (the model assumes that the attributes are conditionally independent of each other, given the class)



Assuming Boolean variables, the **parameters** are:

θ = P(C=true)

θ_{i1} = P(X_i=true | C=true)

θ_{i2} = P(X_i=true | C=false)

Once the model has been trained, it can be used to classify new examples for which the class variable C is unobserved (we observe attribute values $x_1,\ldots,x_N$)

$P(C \mid x_1,\ldots,x_N) = \alpha\, P(C) \prod_i P(x_i \mid C)$

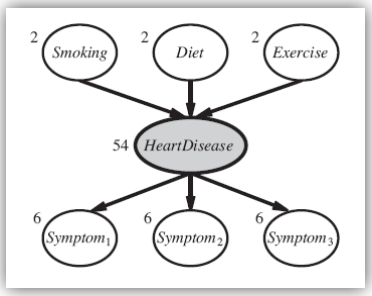
Properties:

- **scales well** to very large problems (with n Boolean attributes, there are just 2n+1 parameters to be learnt)
- **no difficulty with noisy or missing data**
- **gives probabilistic predictions**

Hidden variables

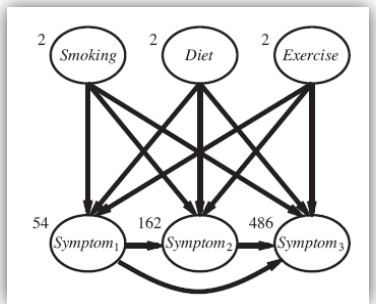
Many real-world problems have **hidden variables**, which are not observable in the data that are available for learning.

For example, medical records often include the observed symptoms, the diagnosis, and the treatment applied, but they seldom contain a direct observation of the disease itself!



If a variable is not observed, why not constructing a model without it?

- The hidden variables can dramatically reduce the number of parameters required to specify a Bayesian network.
- Assume three possible values for each variable. Then the model without the hidden variable requires 708 parameters, while the model with the hidden variable requires only 78 parameters.



EM algorithm

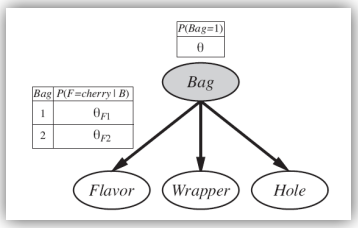
How to learn conditional distribution for the hidden variable if the value of that variable is not given in examples?

- **pretend that we know the parameters** of the model
- **infer the expected values** of hidden variables to ”complete” the data (E-step, expectation)
- **update the parameters** to maximize likelihood of the model (M-step, maximization)
- iterate until convergence

Model example

Consider that some candies have a hole in the middle and some do not and that there are two bags of candies that have been mixed together.

The distribution of candies in each bag is described by naïve Bayes model, where the Bag is a hidden variable



– parameters
  - θ candy comes from Bag 1
  - θ_{F1}, θ_{F2} the flavor is cherry given the bag 1 or 2
  - θ_{W1}, θ_{W2} the wrapper is red given the bag 1 or 2
  - θ_{H1}, θ_{H2} the candy has a hole given the bag 1 or 2

– we generated 1000 samples from a model with the following distribution:

	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	273	93	104	90
F=lime	79	100	94	167

Application or the EM algorithm

We start by initializing the parameters

$\theta^{(0)} = \theta^{(0)}_{F1} = \theta^{(0)}_{W1} = \theta^{(0)}_{H1} = 0.6$

$\theta^{(0)}_{F2} = \theta^{(0)}_{W2} = \theta^{(0)}_{H2} = 0.4$

Because the bag is a hidden variable, we calculate the expected counts instead (using any inference algorithm for Bayesian networks):

$N(Bag=1) = \Sigma_j P(Bag=1 \mid flavor_j, wrapper_j, holes_j)$

We update the parameters (N is the number of examples)

$\theta^{(0)} = N(Bag=1) / N$

– a general principle of parameter update

- let $\theta^{(0)}_{i,j,k}$ be the parameter $P(X_i = x_{ij} \mid \mathbf{U}_i = \mathbf{u}_{ik})$ for X_i with parents $\mathbf{U}_i$
- $\theta_{ijk} \leftarrow N(X_i = x_{ij}, \mathbf{U}_i = \mathbf{u}_{ik}) / N(\mathbf{U}_i = \mathbf{u}_{ik})$

$\theta^{(1)} = 0.6124$, $\theta^{(1)}_{F1} = 0.6684$, $\theta^{(1)}_{W1} = 0.6483$, $\theta^{(1)}_{H1} = 0.6558$

$\theta^{(1)}_{F2} = 0.3887$, $\theta^{(1)}_{W2} = 0.3817$, $\theta^{(1)}_{H2} = 0.6558$

EM algorithm in general

The EM algorithm works in two steps

– calculate expected values of hidden variables for each example (expectation)

– then recompute the parameters (maximization)

At a glance

– **x** are all the observed values in all the examples

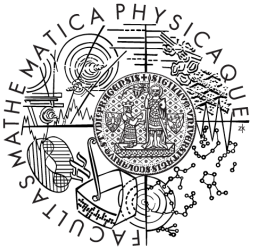
– **Z** are all the hidden variables

– θ are all the parameters for the probability model

$$\theta^{(i+1)} \leftarrow \text{argmax}_{\theta(i)} \Sigma_{\mathbf{z}} P(\mathbf{Z}=\mathbf{z} \mid \mathbf{x}, \theta^{(i)}) \, L(\mathbf{x}, \mathbf{Z}=\mathbf{z} \mid \theta^{(i)})$$

M-step (maximization)

E-step (expectation)



Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz