How does CRF work?

Sandeep Aparajit

Agenda

- Probability *refresher*
- Naïve Bayes Model
- Maximum Entropy Model
- Hidden Markov Model
- Conditional Random Field
- Appendix
    - Markov Chain | Markov Process | Stochastic Process | Concave Equation
    - Likelihood | Log Likelihood | Maximum Likelihood
    - Entropy | Maximum Entropy Principle

★ Indicates an important concept that will be used in the forthcoming slides.

Probability

- Probability is a measure or estimation of how likely it is that something will happen or that a statement is true.

- Probabilities are given a value between 0 (0% chance or will not happen) and 1 (100% chance or will happen). The higher the degree of probability, the more likely the event is to happen, or, in a longer series of samples, the greater the number of times such event is expected to happen.

- For our purposes, events are expressions about random variables, such as Two heads in 6 coin tosses.

- Two events are mutually exclusive if they cannot both be true. For example the events “The coin is heads” and “The coin is tails” are mutually exclusive.

Basics of Probability

- The probability $p(X = x)$ of variable X being in state x is represented by a value between 0 and 1
- $p(\boldsymbol{X} = x) = 1$ means that we are certain X is in state x.

- The summation of the probability over all the states is 1:

$$\sum_{\mathrm{x} \in \mathrm{dom}(x)} p(x = \mathrm{x}) = 1$$

★

- This is called the **normalization condition**. We will usually more conveniently write

$$\textstyle\sum_x p(x) = 1$$

- Two variables x and y can interact through

$$p(x = a \text{ } or \text{ } y = b) = p(x = a) + p(y = b) - p(x = a \text{ } and \text{ } y = b)$$
$$p(x \text{ } or \text{ } y) = p(x) + p(y) - p(x \text{ } and \text{ } y)$$

Marginals

- Given a joint distribution $p(x, y)$ the distribution of a single variable is given by

$$p(x) = \sum_y p(x, y)$$ ★

- Here $p(x)$ is termed a marginal of the joint probability distribution $p(x, y)$. The process of computing a marginal from a joint distribution is called **marginalisation**. More generally, we have

$$p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \sum_{x_i} p(x_1, \ldots, x_n)$$

Conditional Probability

- The probability of event x conditioned on knowing event y (or more shortly, the probability of x given y) is defined as

$$p(x|y) \equiv \frac{p(x,y)}{p(y)}$$ ★

- If $p(y) = 0$ then $p(x|y)$ is not defined. From this definition and $p(x,y) = p(y,x)$ we immediately arrive at **Bayes' rule**

$$\boxed{p(x|y) = \frac{p(y|x)p(x)}{p(y)}}$$ ★

Joint Probability (for discrete variables)

- The joint probability mass function of two discrete random variables is equal to:

$$p(X = x \text{ and } Y = y) = p(Y = y|X = x) \cdot p(X = x)$$
$$i.e.\quad p(x, y) = p(y|x) \cdot p(x)$$

- In general, the joint probability distribution of n, discrete random variables $x_1, x_2, x_3 \ldots x_n$ is equal to:

$$p(x_1, x_2, \ldots, x_n) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_2, x_1) \ \ldots p(x_n|x_1, x_2 \ldots x_{n-1})$$

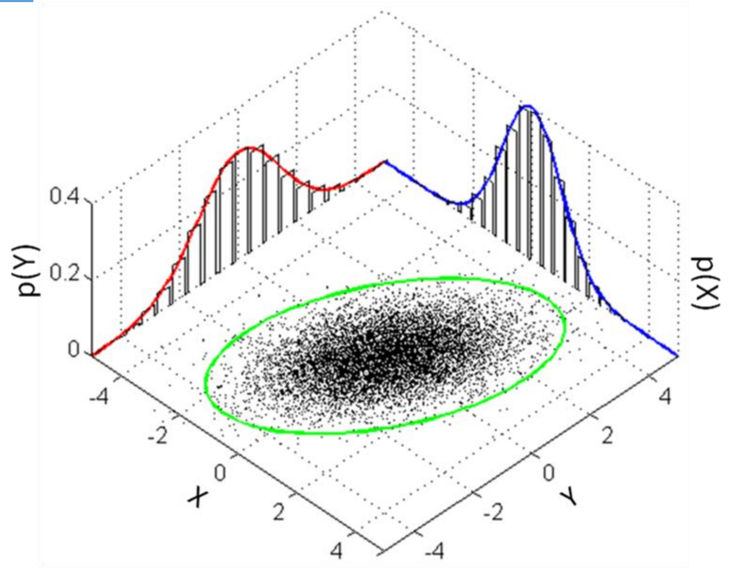
  This is called the **chain rule of probability**.

- **Thus for larger N, computation of joint probability is complex.**

Many sample observations (black) are shown

from a joint probability distribution.

Read more



Independence

- Variables x and y are independent if knowing the state (or value in the continuous case) of one variable gives no extra information about the other variable. Mathematically, this is expressed by

$$p(x, y) = p(x)\, p(y) \quad ★$$

- If $p(x|y) = p(x)$ for all states of x and y, then the variables x and y are said to be independent. If

$$p(x, y) = kf(x) \cdot g(y)$$

for some constant k, and positive functions $f(\cdot)$ and $g(\cdot)$ then x and y are independent and we write $x \perp\!\!\!\perp y$.

Conditional Independence

- $X \perp\!\!\!\perp Y|Z$ denotes that the two sets of variables X and Y are independent of each other provided we know the state of the set of variables Z . For conditional independence, X and Y must be independent given all states of Z. Formally, this means that

$$p(\mathcal{X}, \mathcal{Y}|\mathcal{Z}) = p(\mathcal{X}|\mathcal{Z})p(\mathcal{Y}|\mathcal{Z})$$

for all states of X , Y , Z . In case the conditioning set is empty we may also write $X \perp\!\!\!\perp Y$ for $X\ Y|\Phi$; in which case X is (*unconditionally*) independent of Y .

- If X and Y are not conditionally independent, they are conditionally dependent. This is written

$$X \top\!\!\!\top Y|Z$$

- Similarly $X \top\!\!\!\top Y|\Phi$; can be written as $X \top\!\!\!\top Y$.

- Intuitively, if x is conditionally independent of y given z, this means that, given z, y contains no additional information about x. Similarly, given z, knowing x does not tell me anything more about y.

Statistical Modeling

- Statistical modeling addresses the problem of constructing a stochastic model to predict behavior of a random process.

- **Tasks**:
  - First task is to determine a set of statistics that captures the behavior of a random process
  - Given this statistics, the second task is to corral these facts into an accurate model of the process – a model capable of predicting the future output of the process

Naïve Bayes Classifier

Thomas Bayes
1702 - 1761

Naïve Bayes Classifier

- A Naïve Bayes model is a probability distribution $p(y|\bar{x})$ with an input vector $\bar{} = (x_1, x_2, \dots . xm)$, where x_i $(1 \leq i \leq m)$ are features and y is the class variable to be predicted. That probability can be formulated with Bayes' law:

$$\boxed{p(y|x) = \frac{p(x|y)p(y)}{p(x)}}$$

- The denominator $p(x)$ is not important for classification as it can be understood as a normalization constant which can be computed by considering all possible values of y.
- The numerator can be written as joint probability also:

$$\boxed{p(x|y)p(y) = p(y, x)}$$

which can be too complex to be computed directly (especially for large x).

Naïve Bayes contd.

- A general decomposition can be formulated using the chain rule as:

$$p(y, x) = p(y) \prod_{i=2}^{m} p(x_i, x_{i-1}, \dots, x_1, y)$$

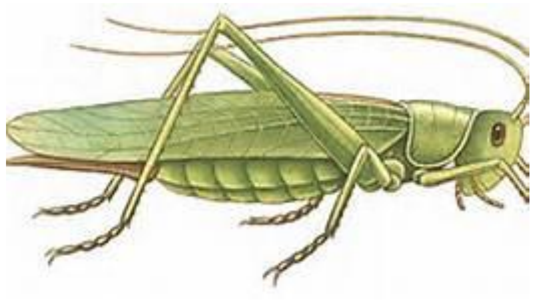
- *In practice, it is often assumed that all input variables x_i are conditionally independent of each other which is known as Naïve Bayes assumption*. That means $p\left(x_i \middle| y, x_j\right) = p(x_i|y)$ holds for all $i \neq j$.
- Based on this the Naïve Bayes classifier is formulates as:

$$\boxed{p(y|\vec{x}) \propto p(y, \vec{x}) = p(y) \prod_{i=1}^{m} p(x_i|y) \, .}$$

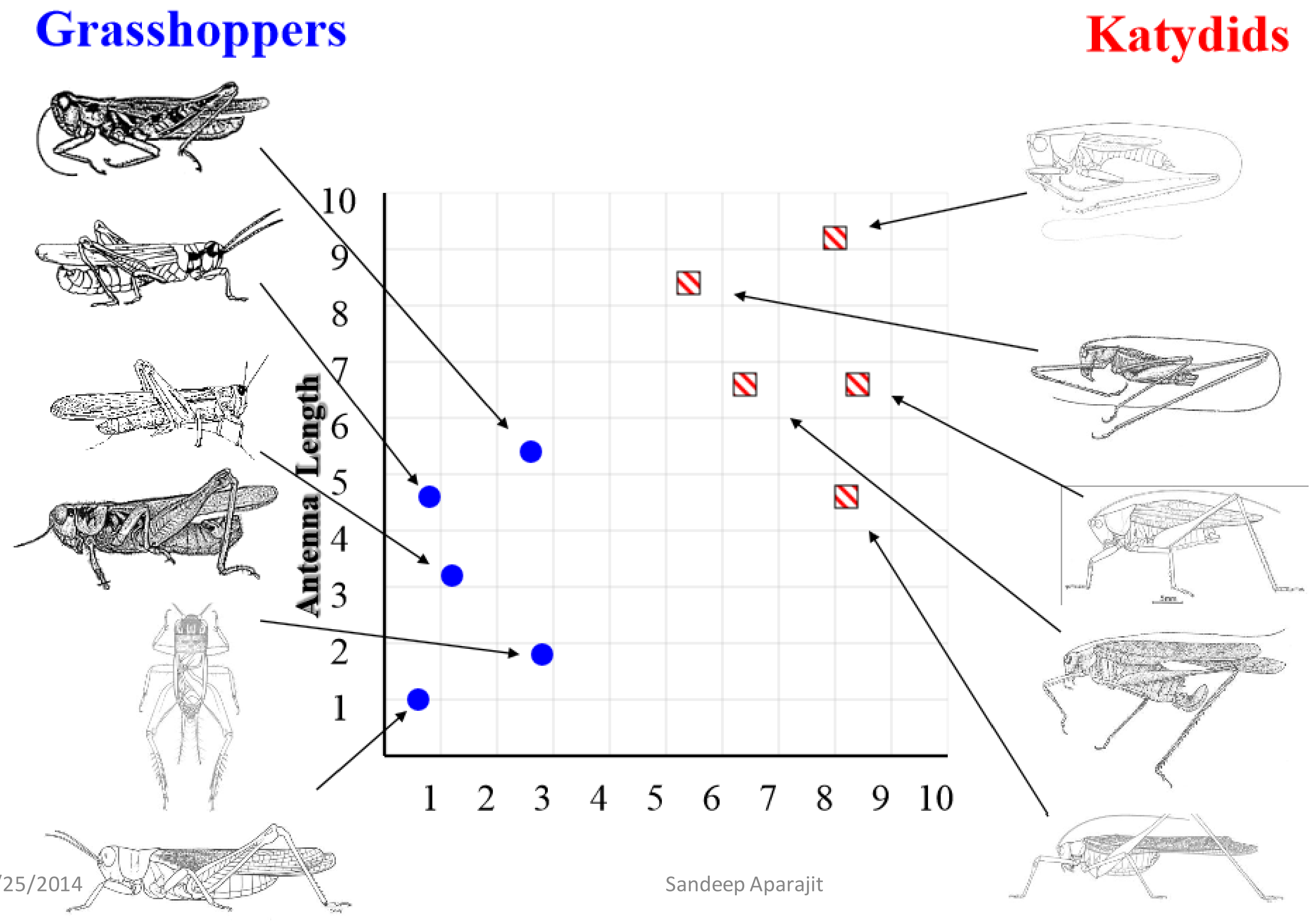
★

- Dependencies of the input variable x are not modeled, probably leading to an imperfect representation of the real world. Nevertheless, it performs surprisingly well in many real world applications such as email classification etc.

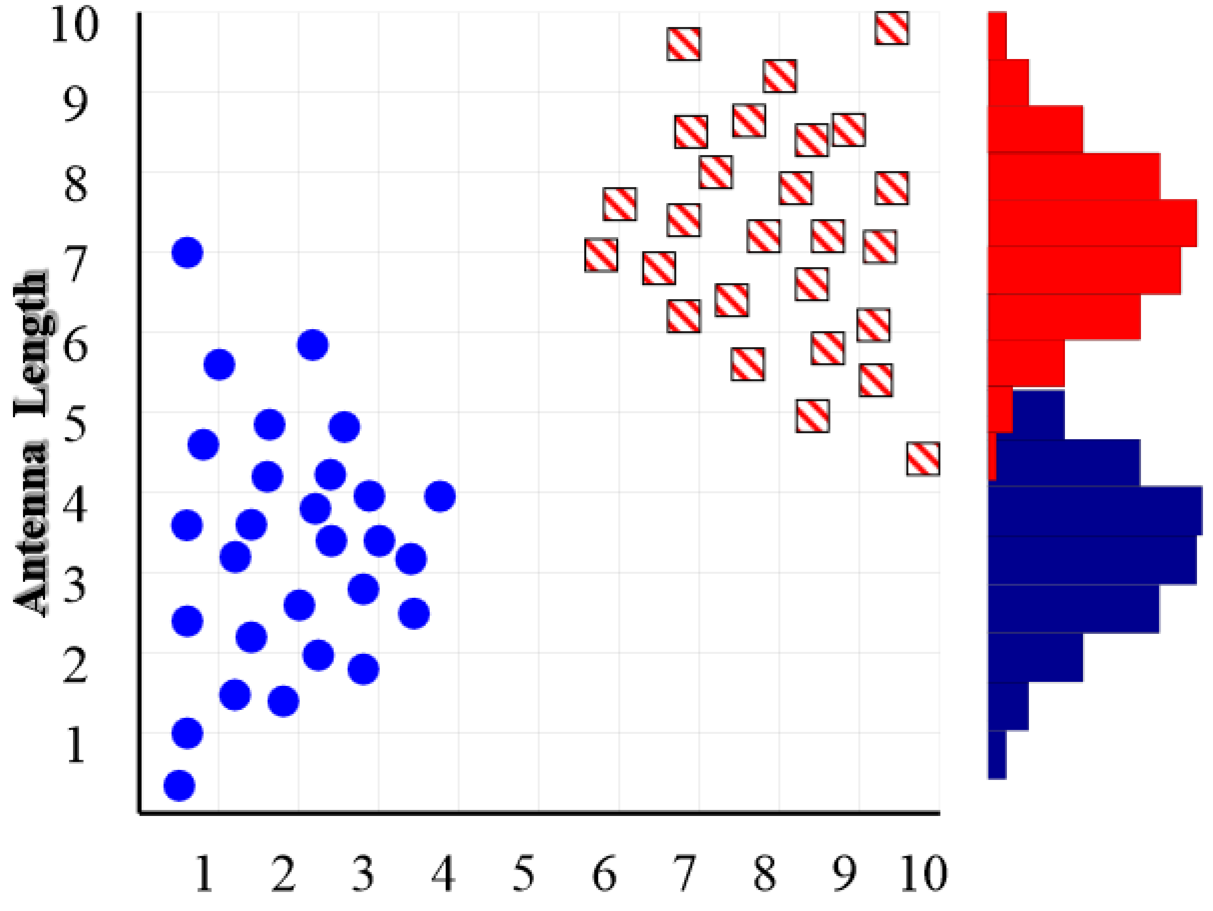
Example: Grasshopper Or Katydid?



Distribution:



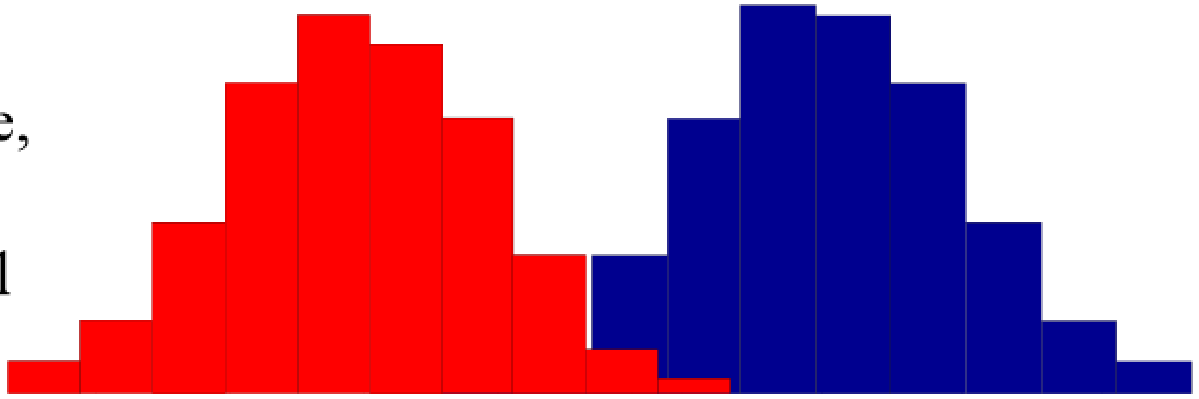
Histogram for Antenna Length



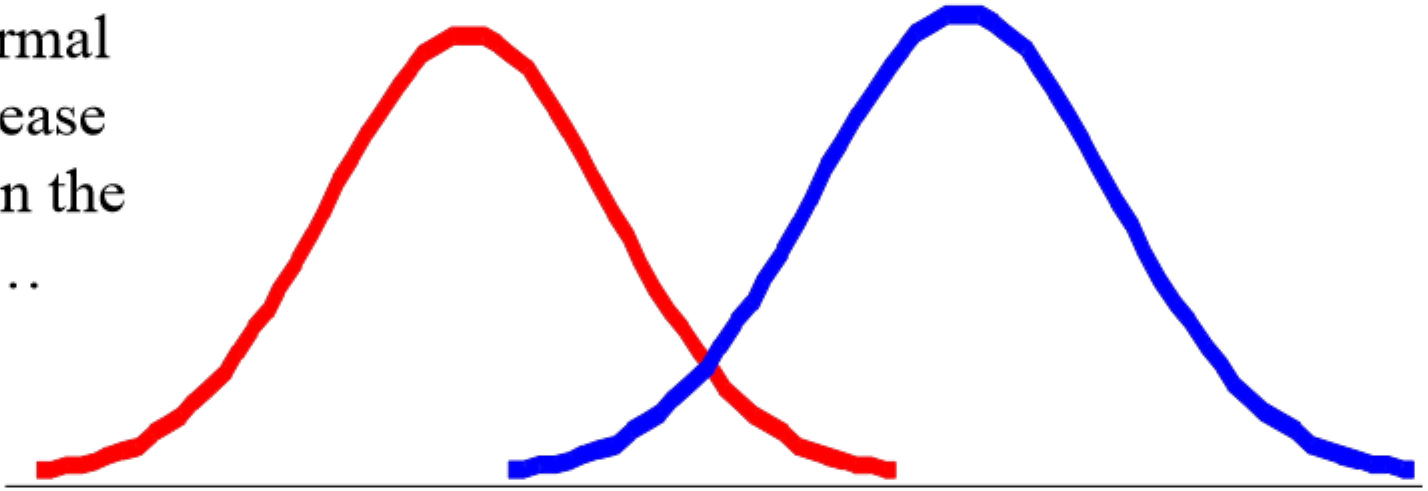
▨ Katydids
● Grasshoppers

Simplified Histogram

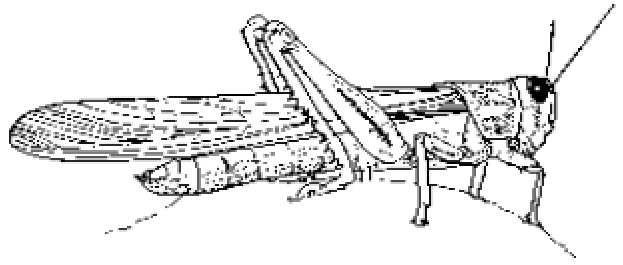
We can leave the histograms as they are, or we can summarize them with two normal distributions.



Let us us two normal distributions for ease of visualization in the following slides…



How do we classify this insect?

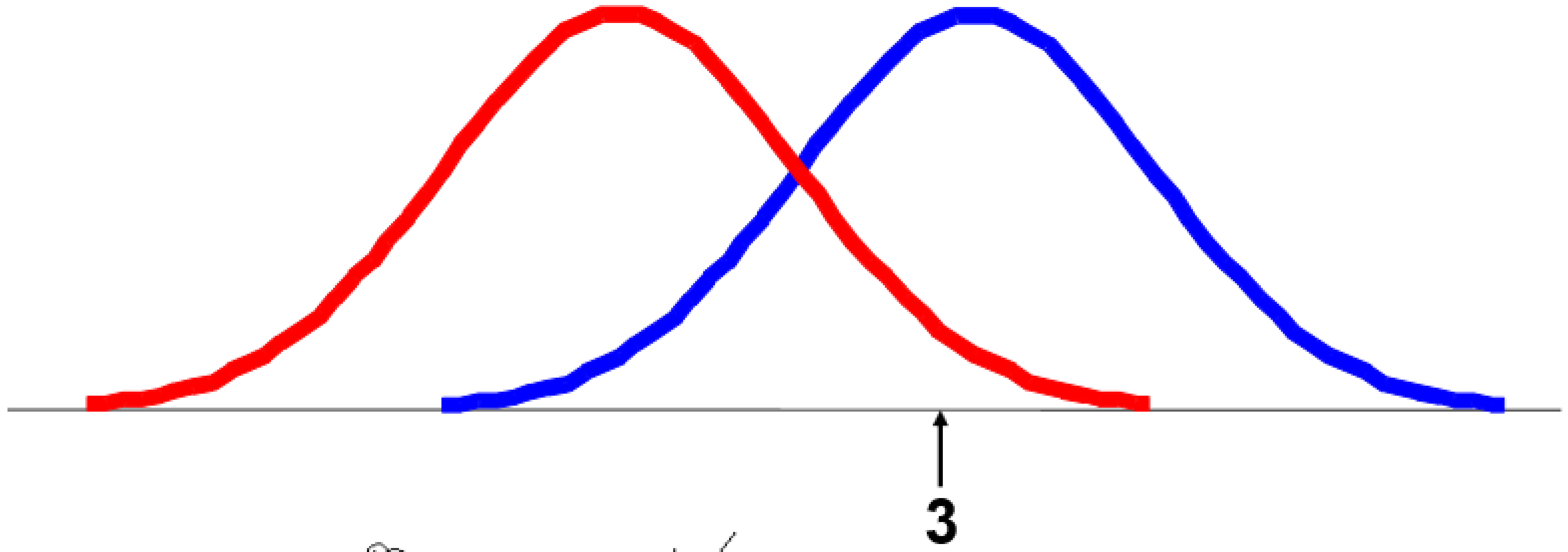


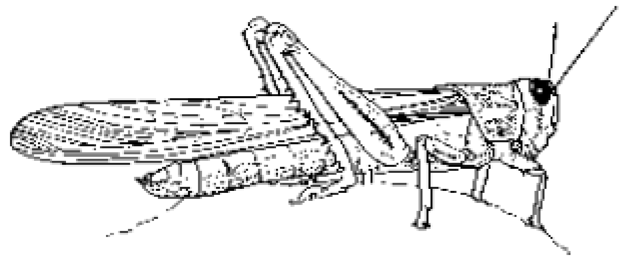
- We found this insect with its antennae length 3 units.
- How can we classify it?

We can just ask ourselves, given the distributions of antennae lengths we have seen, is it more probable that our insect is a Grasshopper or a Katydid??

Compute the probability

- $p(y_j|x)$ = probability of class y_j given the observation x





Antennae length is **3**

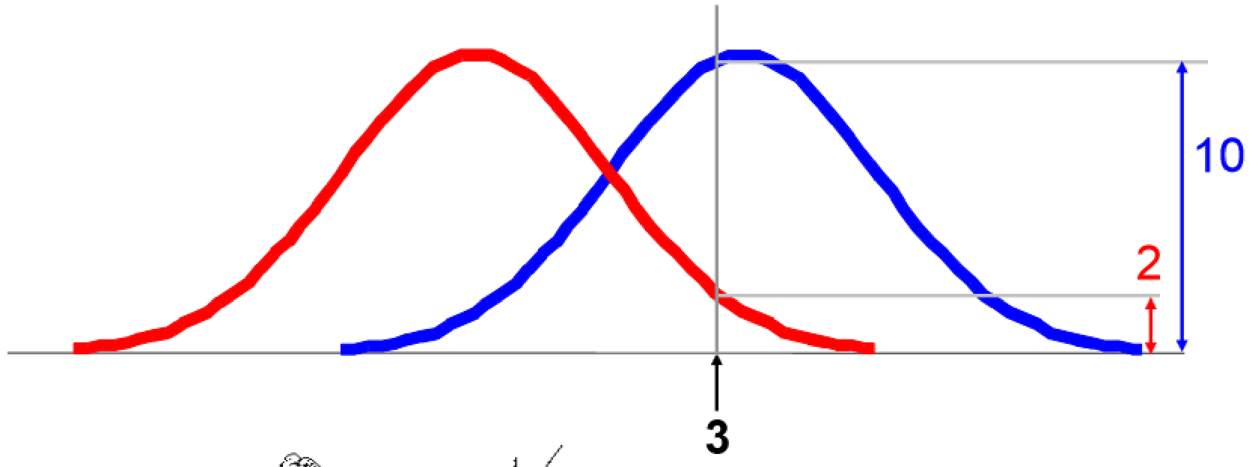
- Compute:
  - $p(Grasshopper|3)$
  - $p(Katydid|3)$

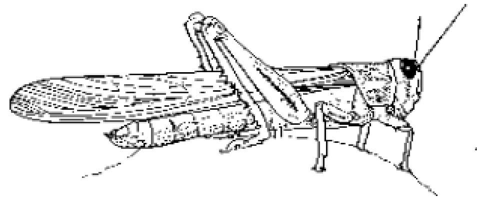
Compute the probability

- $p(y_j|x)$ = probability of class y_j given the observation x

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

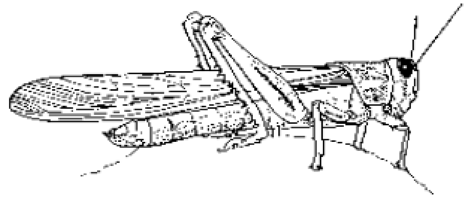
P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166





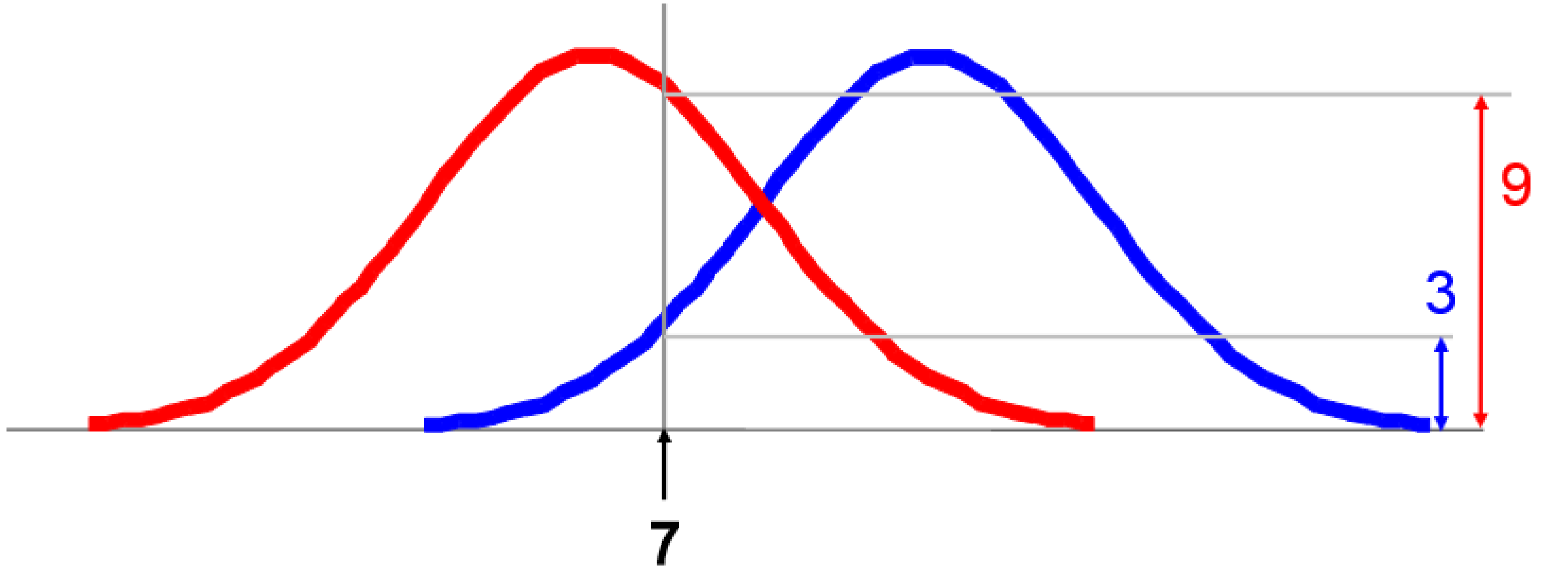
Antennae length is **3**

New insect? No worries!

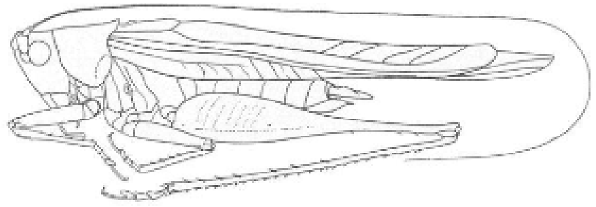


P(**Grasshopper** | **7**) = 3 / (3 + 9) = 0.250

P(**Katydid** | **7**) = 9 / (3 + 9) = 0.750



Antennae length is 7



Example: 2

Note: Drew can be a female or a male

Assume that we have two classes

c_1 = male, and c_2 = female.

We have a person whose sex we do not know, say "*drew*" or *d*.

Classifying *drew* as male or female is equivalent to asking is it more probable that *drew* is male or female, I.e which is greater $p(\text{male} \mid drew)$ or $p(\text{female} \mid drew)$

Drew Barrymore

Drew Carey

Bayes Rule:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

What is the probability of being called "drew" given that you are a male?

What is the probability of being a male?

$$p(\text{male} \mid drew) = \frac{p(drew \mid \text{male})\, p(\text{male})}{p(drew)}$$

What is the probability of being name "drew"?

(actually irrelevant since it is same for all classes)

- This is the famous officer who arrested the gangster. Is the officer drew male or female?

Officer Drew

- We have a small database of people names and sex…
- We can apply Bayes rules using this database

$$\boxed{p(y|x) = \frac{p(y)p(x|y)}{p(x)}}$$

Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

$$\boxed{p(y|x) = \frac{p(y)p(x|y)}{p(x)}}$$

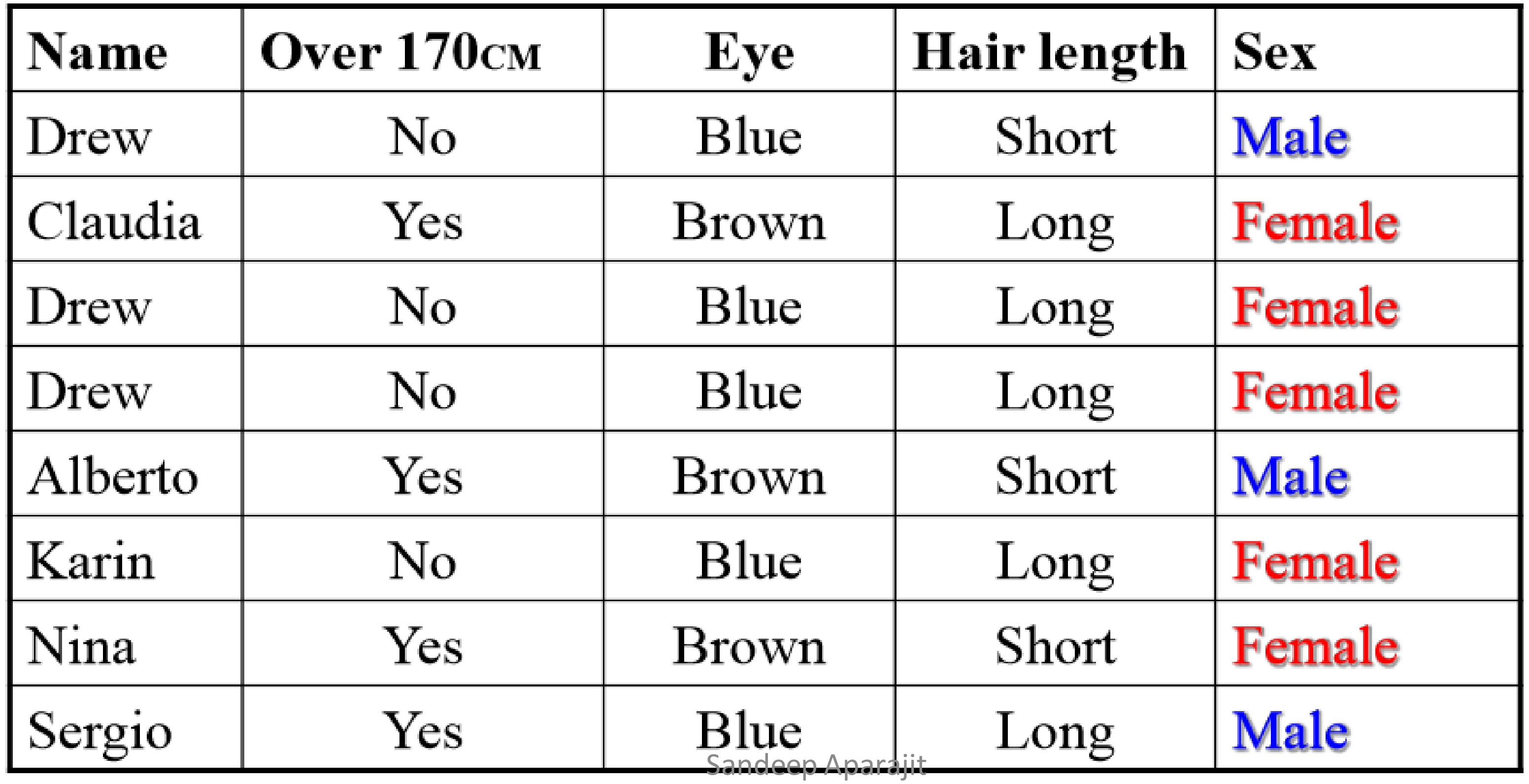
Name	Sex
Drew	Male
Claudia	Female
Drew	Female
Drew	Female
Alberto	Male
Karin	Female
Nina	Female
Sergio	Male

- $p(male|drew) = \frac{p(drew|male)p(male)}{p(drew)}$
- $p(male|drew) = \frac{\frac{1}{3} * \frac{3}{8}}{\frac{3}{8}} = \boxed{\frac{0.125}{3/8}}$
- $p(female|drew) = \frac{p(drew|female)p(female)}{p(drew)}$
- $p(female|drew) = \frac{\frac{2}{3} * \frac{5}{8}}{\frac{3}{8}} = \boxed{\frac{0.250}{3/8}}$

Offier drew is more likely to be a *female*.

What if we have more that one features?

- So far we have only considered Bayes Classification when we have one attribute (the "antennae length", or the "name"). But we may have *many features*!



Name	Over 170CM	Eye	Hair length	Sex
Drew	No	Blue	Short	Male
Claudia	Yes	Brown	Long	Female
Drew	No	Blue	Long	Female
Drew	No	Blue	Long	Female
Alberto	Yes	Brown	Short	Male
Karin	No	Blue	Long	Female
Nina	Yes	Brown	Short	Female
Sergio	Yes	Blue	Long	Male

- In the previous slide, we have see the Bayes assumption
  - *In practice, it is often assumed that all input variables x_i are conditionally independent of each other which is known as Naïve Bayes assumption.*

- Thus our equation becomes:
- $p(x|y_j) = p(x_1|y_j) \ * \ p(x_2|y_j) \ * \ p(x_1|y_j) * \ldots\ldots * \ p(x_n|y_j)$

Probability of class y_j generating instance d equals…

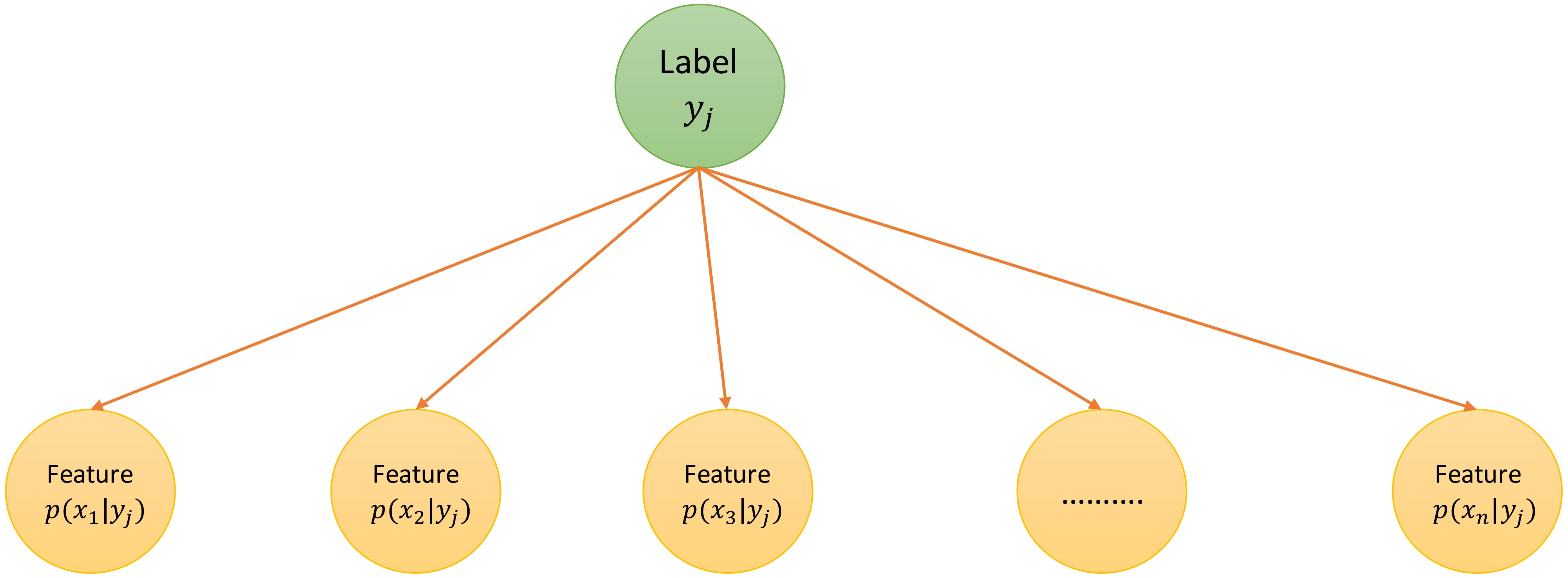
Probability of class y_j generating the observed Value for feature 1 Multiplied by…

Probability of class y_j generating the observed Value for feature 2 Multiplied by…

……………

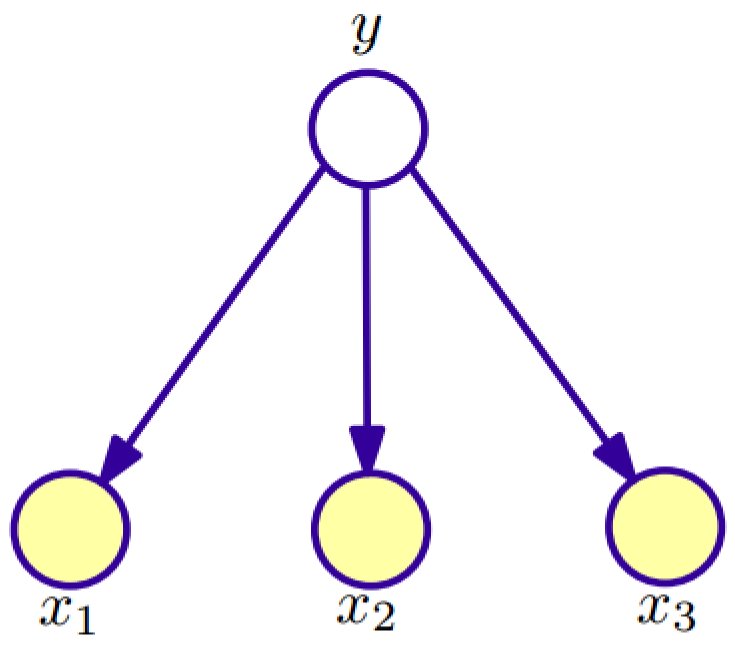
Probability of class y_j generating the observed Value for feature n

Graphical Representation

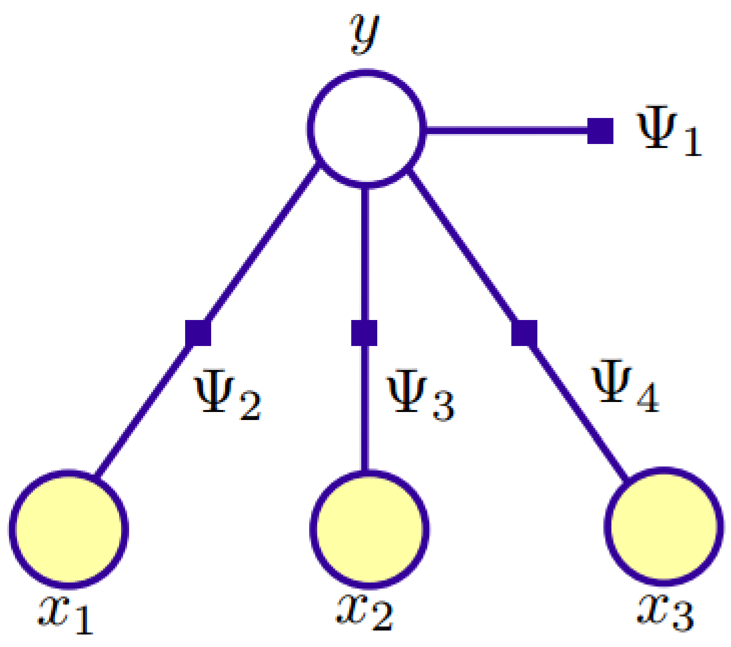


- Note the direction of the arrows, which state that each class causes certain features, with a certain probability

Factor Graph



(a) Independency graph



(b) Factor graph

Naïve Bayes classifier

Advantages/Disadvantages of Naïve Bayes

- Advantages:
  - Fast to train (single scan). Fast to classify
  - Not sensitive to irrelevant features
  - Handles real and discrete data
  - Handles streaming data well

- Disadvantages:
  - Assumes independence of features

Maximum Entropy/Logistic Regression

- Suppose we wish to model an expert translator's decisions concerning the proper French rendering of the English word *in*.

- Our model p of the expert's decisions assigns to each French word or phrase f an estimate, $p(f)$, of the probability that the expert would choose f as a translation of *in*.

- To guide us in developing p, we collect a large sample of instances of the expert's decisions.

- Our goal is to extract a set of facts about the decision-making process from the sample (*the first task of modeling*) that will aid us in constructing a model of this process (*the second task*).

- One obvious clue we might glean from the sample is the list of allowed translations:
- Assuming that the expert translator always chooses among the following five French phrases: *{dans, en, à, au cours de, pendant}*
- With this information, we can impose our first constraint:

$$p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1$$

This equation represents our first statistic of the process; we can now proceed to search for a suitable model that obeys this equation.

- Of course, there are an infinite number of models *p* for which this identity holds. One model satisfying the above equation is

$p(dans) = 1$; in other words, the model always predicts *dans*.

- Another model obeying this constraint predicts *pendant* with a probability of 1/2, and *à* with a probability of 1/2.

- Each seems to be making rather bold assumptions, with no empirical justification. Put another way, these two models **assume more than we actually know** about the expert's decision-making process.
- The expert chose exclusively from among these five French phrases; given this, the most intuitively appealing mode is:

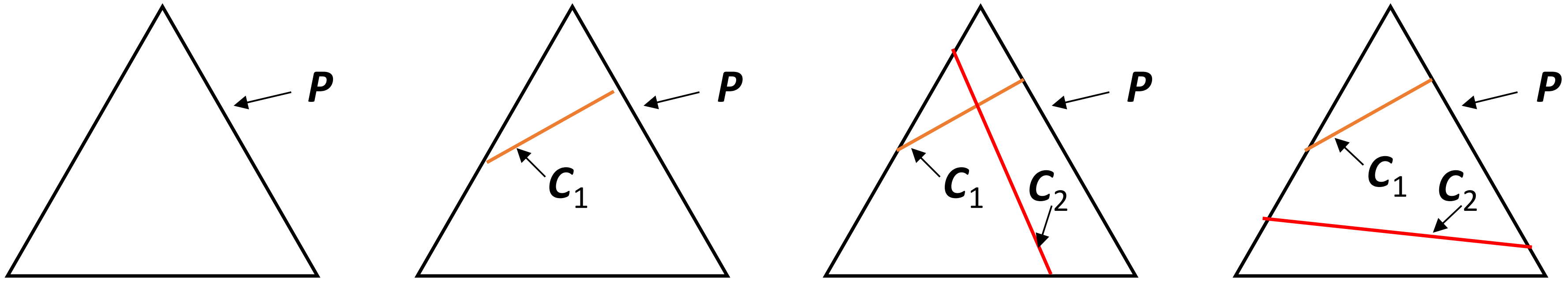
$$p(dans) = \frac{1}{5} \qquad p(en) = \frac{1}{5} \quad p(\text{à}) = \frac{1}{5} \quad p(au\ cours\ de) = \frac{1}{5} \qquad p(pendant) = \frac{1}{5}$$

- Suppose we notice that the expert chose either $dans$ or en 30% of the time. We could apply this knowledge to update our model as:
  - $p(dans) + p(en) = \frac{3}{10}$
  - $p(dans) + p(en) + p(\text{à}) + p(au\ cours\ de) + p(pendant) = 1$

- There are many probability distributions consistent with these two constraints. A reasonable choice for p is again the *most uniform*:

$$p(dans) = \frac{3}{20} \quad p(en) = \frac{3}{20} \quad p(\text{à}) = \frac{7}{30} \quad p(au\ cours\ de) = \frac{7}{30} \quad p(pendant) = \frac{7}{30}$$

- Another interesting fact is, in half the cases, the expert chose either $dans$ or à. Thus the model has **3** constraints:
  - $p(dans) + p(en) = \frac{3}{10}$
  - $p(dans) + p(en) + p(\text{à}) + p(au\ cours\ de) + p(pendant) = 1$
  - $p(dans) + (\text{à}) = \frac{1}{2}$



(a) (b) (c) (d)

a) If we impose no constraints, then all probability models are allowable

b) Imposing one linear constraint $\boldsymbol{C}_1$ restricts us to those $p \in \boldsymbol{P}$ which lie on the region defined by $\boldsymbol{C}_1$

c) A second linear constraint could determine p exactly, if the two constraints are satisfiable, where the intersection of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ is non-empty. $p \in \boldsymbol{C}_1 \cap \boldsymbol{C}_2$

d) Alternatively, a second linear constraint could be inconsistent with the first (i.e. $\boldsymbol{C}_1 \cap \boldsymbol{C}_2 = \varnothing$); no $p \in \boldsymbol{P}$ can satisfy them both

- As we have added complexity, we have encountered two difficulties:
  - First, what exactly is meant by uniform? How do we measure it?
  - Second, having determined a suitable answer to this, how do we go about finding the most uniform model subject to a set of constraints?

- The maximum entropy method e questions!

Entropy is a measure of the uncertainty in a random variable.

- Why **maximum** entropy?
- Maximize entropy = Minimize commitment

- Model all that is known and assume nothing about what is unknown.
  - Model all that is known: satisfy a set of constraints that must hold

  - Assume nothing about what is unknown:
    choose the most "uniform" distribution
    ➔ choose the one with maximum entropy

Maximum Entropy Modeling

- We consider a random process that produces an output value y, a member of a finite set Y.
- For the translation example just considered, the process generates a translation of the word *in*, and the output y can be any word in the set {*dans*, *en*, *à*, *au cours de*, *pendant*}.

- In generating y, the process may be influenced by some contextual information x, a member of a finite set X.
- In the present example, this information could include the words in the English sentence surrounding in.

- Our task is to construct a *stochastic* model that *accurately* represents the behavior of the random process. Such **a model is a method of estimating the conditional probability that, given a context x, the process will output y**. We will denote by $p(y|x)$ the probability that the model assigns to y in context x.

Training Data

- We observe the behavior of the random process for some time, collecting a large number of samples $(x_1, y_1), (x_2, y_2)$ ..... (x_n, y_n).
- In the example we have been considering, each sample would consist of a phrase x containing the words surrounding in, together with the translation y of *in* that the process produced.
- We can summarize the training sample in terms of its empirical probability distribution $\bar{p}$, defined by:

$$\bar{p}(x, y) \equiv \frac{1}{N} * number\ of\ times\ that\ (x, y) occurs\ in\ the\ sample$$

Typically, a particular pair (x, y) *will either not occur at all in the sample, or will occur at most a few times.*

- Our goal is to construct a statistical model of the process that generates the training sample $\bar{p}(x, y)$.

- The Maximum Entropy Model is a **conditional probability model**. It is based on the Principle of Maximum Entropy which states that:

> If incomplete information about a probability distribution is available, the only unbiased assumption that can be made is a distribution which is as uniform as possible given the available information.

- Under this assumption, the proper probability distribution is the one which maximizes the entropy given the constraints from the training material.

- For the conditional model $p(y|x)$ the conditional entropy $H(y|x)$ is applied, which is defined as-

$$H(y|x) = -\sum_{(x,y)\in z} p(y,x)\log p(y|x)$$

- The set $\mathcal{Z} = \mathcal{X}\ X\ \mathcal{Y}$consists of $\mathcal{X}$, the set of all possible input variables x, and $\mathcal{Y}$, the set of all possible output variables y. Note that $\mathcal{Z}$ contains not only the combinations of x and y occurring in the training data, but all possible combinations.

- The basic idea behind Maximum Entropy Models is to find the model $p^*(y|x)$ which on the one hand has the largest possible conditional entropy but is on the other hand still consistent with the information from the training material. The objective function, later referred to as ***primal problem***, is thus-

$$\boxed{p^*(y|x) = \operatorname*{argmax}_{p(y|x)\in P} H(y|x)}$$

where P is the set of all models consistent with the training material. What is meant with “consistent" will be explained in detail later.

- The building block for this model will be a set of statistics of the training sample. For instance, we might notice that, in the training sample, if $April$ is the word following in, then the translation of in is en with frequency 9/10.
- Here, these are defined as binary-valued functions $f_i(x, y) \in \{0,1\}(1 \leq i \ \leq m)$ which depend on both the input variable x and the class variable y. An example for such a function is:

$$f_i(x,y) = \begin{cases} 1 \ if \ y = en \ and \ April \ follws \ in \\ 0 \ otherwise \end{cases}$$

- Such functions are called ***feature functions*** or ***features***
- The expected value of each feature f_i is estimated from the empirical distribution $\bar{p}(x, y)$. The empirical distribution is obtained by simply counting *how often the different values of the variables occur in the training data*:

$$\bar{E}(f_i) = \sum_{(x,y)\in Z} \bar{p}(x,y) f_i(x,y)$$

- All possible pairs (x, y) are taken into account here. As the empirical probability for a pair (x, y) which is not contained in the training material is 0, $\bar{E}(f_i)$ can be rewritten as-

$$\bar{E}(f_i) = \frac{1}{N} \sum_{(x,y) \in \mathcal{T}} f_i(x, y)$$

- The size of the training set is $N = |\mathcal{T}|$. Thus, $\bar{E}(f_i)$ can be calculated by counting how often a feature f_i is found with value 1 in the training data and dividing that number by the size N of the training set.
- Analogously to the previous equation, the *expected value of a feature on the model distribution* is defined as-

$$E(f_i) = \sum_{(x,y) \in \mathcal{Z}} p(x, y) f_i(x, y)$$

- Of course, $p(x, y)$ cannot be calculated in general because the number of all possible $x \in X$ can be enormous. This can be addressed by rewriting $E(f_i)$ by –

$$E(f_i) = \sum_{(x,y) \in \mathcal{Z}} p(x) p(y|x) f_i(x, y)$$

Look into the Joint Probability slide to understand why computation of $p(x, y)$ is complex.

$$E(f_i) = \sum_{(x,y)\in\mathcal{Z}} p(x)p(y|x)f_i(x,y)$$

- Substituting $p(x)$ with the empirical distribution $\bar{p}(x)$. This is an approximation to make the calculation of $E(f_i)$ possible (see details) –

$$E(f_i) \approx \sum_{(x,y)\in\mathcal{Z}} \bar{p}(x)p(y|x)f_i(x,y)$$

- Which can be transformed into-

$$E(f_i) = \frac{1}{N}\sum_{x\in\mathcal{T}}\sum_{y\in\mathcal{Y}} p(y|x)f_i(x,y)$$

- Only x values occurring in the training data are considered $(x \in \mathcal{T})$ while all possible y values are taken into account $(y \in \mathcal{Y})$. In many applications the set $\mathcal{Y}$ typically contains only a small number of variables. Thus, summing over y is possible here and $E(f_i)$ can be calculated efficiently.

- The objective function as we saw earlier is $p^*(y|x) = \underset{p(y|x)\in P}{\operatorname{argmax}} H(y|x)$
- Thus, from above, the model $p^*(y|x)$ is consistent with the evidence found in the training material. That means, for each feature f_i its expected value on the empirical distribution must be equal to its expected value on the particular model distribution, these are the first m constraints-

$$E(f_i) = \bar{E}(f_i)$$

- Another constraint is to have a proper conditional probability ensured by

$$p(y|x) \geq 0 \; for \; all \; x,y \qquad \sum_{y\in\mathcal{Y}} p(y|x) = 1 \; for \; all \; x$$

- Finding $p^*(y|x)$ under these constraints can be formulated as a constrained optimization problem. For each constraint a Lagrange multiplier λ_i is introduced. This leads to the following Lagrange function $\Lambda(p,\vec{\lambda})$:

$$\bigwedge(p,\vec{\lambda}) = \underbrace{H(y|x)}_{\textit{Primal Problem}} + \sum_{i=1}^{m} \lambda_i \underbrace{\left(E(f_i) - \bar{E}(f_i)\right)}_{\substack{\textit{Constraint from equation} \\ E(f_i) = \bar{E}(f_i)}} + \lambda_{m+1} \underbrace{\left(\sum_{y\in\mathcal{Y}} p(y|x) - 1\right)}_{\substack{\textit{Constraint from equation} \\ p(y|x) \geq 0 \; for \; all \; x,y \quad \sum_{y\in\mathcal{Y}} p(y|x) = 1 \; for \; all \; x}}$$

- This is maximized to get the model formulation $p^*_{\lambda}(y|x)$.

Detailed Derivation of $\boldsymbol{p^{*}_{\bar{\lambda}}(y|x)}$

Skip to this slide if you are not interested in the derivation.

- This is the general form the model needs to meet the constraints. The MaximumEntropy Model can then be formulated as-

$$p^*_{\vec{\lambda}}(y|x) = \frac{1}{\boxed{\mathcal{Z}_{\vec{\lambda}}(x)}} \boxed{\exp(\sum_{i=1}^{m} \boxed{\lambda_i} f_i(x,y))}$$

Potential

Weight vector

And $\mathcal{Z}_{\vec{\lambda}}(x)$ then is

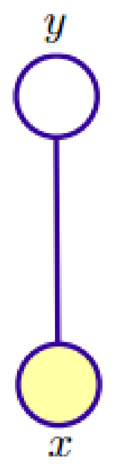
$$\mathcal{Z}_{\vec{\lambda}}(x) = \sum_{y \in \mathcal{Y}} \exp(\sum_{i=1}^{m} \lambda_i f_i(x,y))$$

Normalization constant called *partition function*

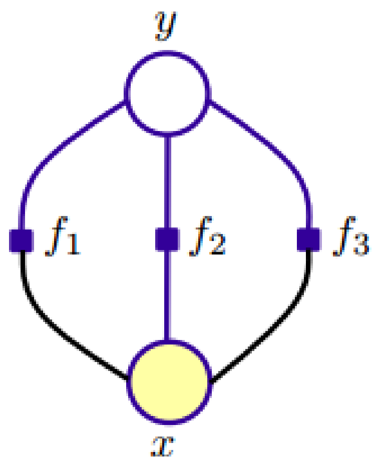
Related to graphical model:

In such log-linear models, potential functions are formulated as the exponential function of weighted features. Such a formulation is frequently used because it fulfills the requirement of strict positivity of the potential functions.

Factor Graph for MEM



(a) Independency graph



(b) Factor graph

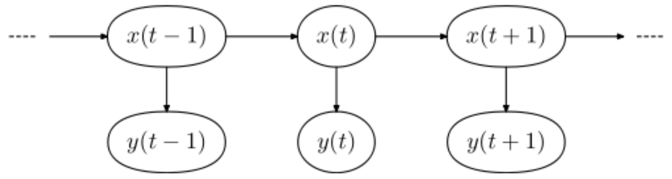
Maximum Entropy Classifier

Hidden Markov Model (HMM)

Hidden Markov Model (HMM)

- The Hidden Markov Model(HMM) is a powerful statistical tool for modeling *generative sequences* that can be characterized by an underlying process generating an observable sequence.

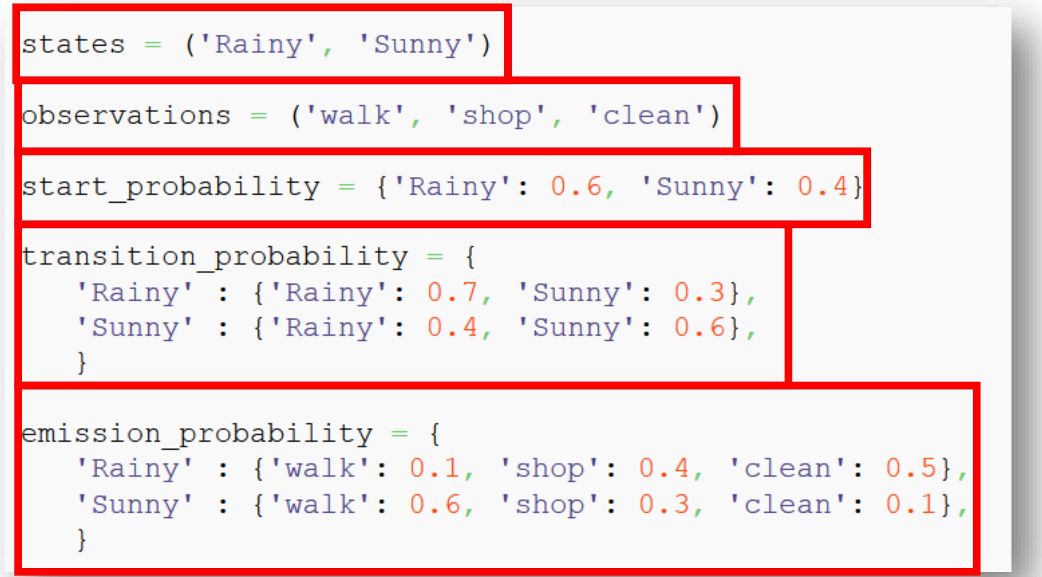
- HMMs have found application in many areas interested in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as part-of-speech tagging, phrase chunking, and extracting target information from documents.



- Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day.
- Bob is only interested in three activities: **walking** in the park, **shopping**, and **cleaning** his apartment. The choice of what to do is determined exclusively by the weather on a given day.
- Alice has no definite information about the weather where Bob lives, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.

- Alice believes that the weather operates as a discrete Markov chain.
- There are two states, "*Rainy*" and "*Sunny*", but she cannot observe them directly, that is, they are *hidden* from her.
- On each day, there is a certain chance that Bob will perform **one** of the following activities, depending on the weather: "*walk*", "*shop*", or "*clean*".

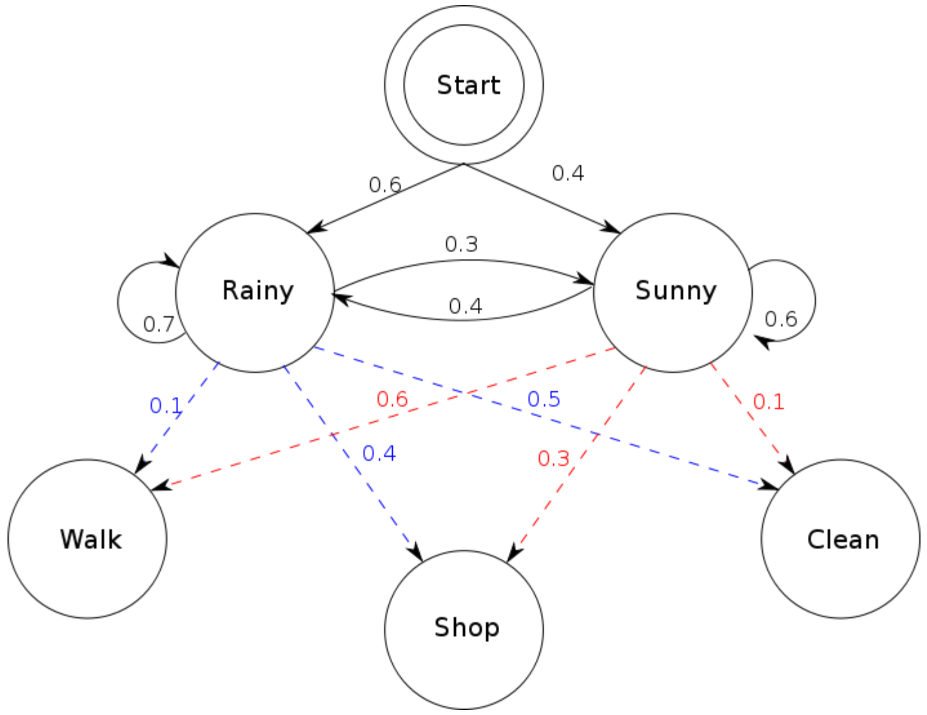
- Since Bob tells Alice about his activities, those are the ***observations***. Alice knows the general weather trends in the area, and what Bob likes to do on average. In other words, the **parameters** of the HMM are known.



```
states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')

start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

transition_probability = {
   'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
   'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},
   }

emission_probability = {
   'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
   'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},
   }
```

- In this piece of code, *start_probability* represents **Alice's belief** about which state the HMM is in when Bob first calls her (all she knows is that it tends to be rainy on average).
- The transition_probability represents the change of the weather in the underlying Markov chain.

- In this example, there is only a 30% chance that tomorrow will be sunny if today is rainy. The emission_probability represents how likely Bob is to perform a certain activity on each day. If it is rainy, there is a 50% chance that he is cleaning his apartment; if it is sunny, there is a 60% chance that he is outside for a walk.



- The entire system is that of a hidden Markov model (HMM). The parameters of the HMM are known here.

Markov Chain Assumptions

- The Markov Assumption
  - Next state is only **dependent** on current state
  - *In our example, tomorrow's weather depends only on today's weather*

- The stationarity assumption
  - Transition probabilities are **independent** of the time the transition takes place

- The output independence assumption
  - Observations are **independent** of previous observations

HMM Formally…

- Thus HMM is given by the joint probability distribution:

$$p(y,x) = \prod_{t=1}^{T} p(y_t|y_{t-1}) \cdot p(x_t|y_t)$$

Each state y_t depends only on its immediate predecessor y_{t-1} and each observation x_t depends only on the current state y_t

- The full HMM is specified using the triplet:

$$\lambda = (A, B, \pi)$$

Weight vector λ

- S is our state alphabet set, and V is the observation alphabet set:

$$S = (s_1, s_2, \dots, s_N)$$
$$V = (v_1, v_2, \dots, v_M)$$

States: Rainy, Sunny

Obs: Walk, Clean, Shop

- We define Q to be a fixed state sequence of length T, and corresponding observations O:

$$Q = q_1, q_2, q_3, \dots, q_T$$
$$O = o_1, o_2, o_3, \dots, o_T$$

Rainy → Rainy → Sunny

Clean → Shop → Walk

- A is a transition array, storing the probability of state j following state i. Note the state transition probabilities are independent of time:

$$A = [a_{ij}], a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$$

Probability of going from Rainy → Rainy

- B is the observation array, storing the probability of observation k being produced from the state j, independent of t:

$$B = [b_i(k)], \qquad b_i(k) = P(x_t = v_k | q_t = s_i)$$

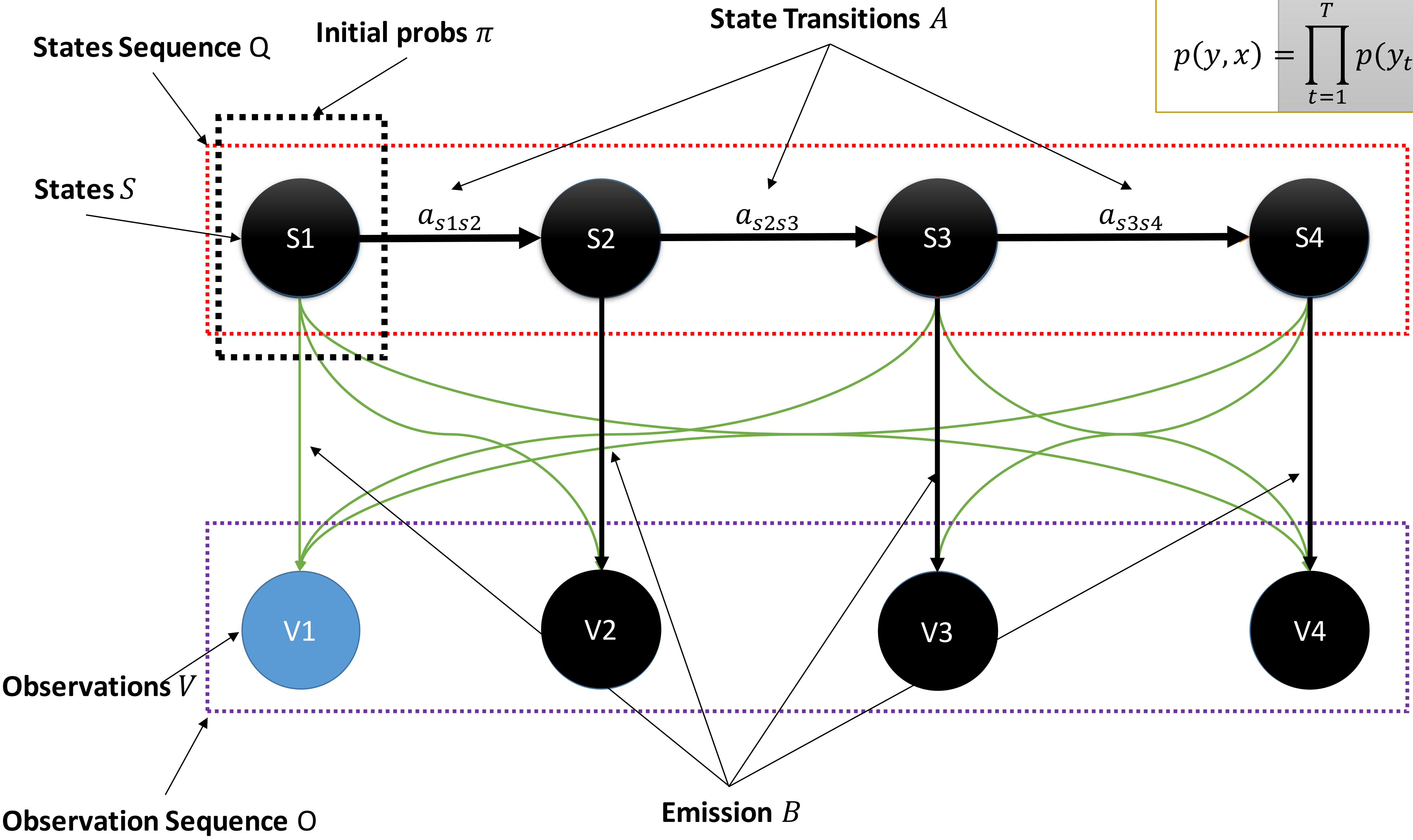
Probability of going from Rainy → Clean

- π is the initial probability array:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i)$$

Probability that the first day is Rainy/Sunny?

$$p(y,x) = \prod_{t=1}^{T} p(y_t|y_{t-1}) \cdot p(x_t|y_t)$$



- Two assumptions are made by the model. The first, called the Markov assumption, states that the current state is dependent only on the previous state, this represents the memory of the model:

$$P(q_t \big| q_1^{t-1}) = P(q_t | q_{t-1})$$

- The independence assumption states that the output observation at time t is dependent only on the current state, it is independent of previous observations and states:

$$P(o_t \big| o_1^{t-1}) = P(o_t | q_t)$$

Evaluation

- Given a HMM, and a sequence of observations, we'd like to be able to compute $P(O|\lambda)$, the probability of the observation sequence given a model.

- This problem could be viewed as one of evaluating how well a model predicts a given observation sequence, and thus allow us to choose the most appropriate model from a set.

- The probability of the observations O for a specific state sequence Q is:

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(o_t|q_t,\lambda) = b_{q1}(o_1) * b_{q2}(o_2) \dots b_{qT}(o_T)$$

and th

Observations are mutually independent, given the hidden states.
Joint distribution of independent variables factorises into marginal distributions of the independent variables.

so we can calculate the probability of the observations given the model as:

$$P(O|\lambda) = \sum_{Q} P(O|Q,\lambda)P(Q|\lambda)$$

$$= \sum \pi_{q_1} b_{q_1}(o_1)\, a_{q_1 q_2} \cdot b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{qT}(o_T)$$

- The above sum is over all state paths
- There are N^T states paths, each ‘costing’ O(T) calculations, leading to $O(TN^T)$ time complexity.
- Let’s look at a better algorithm…

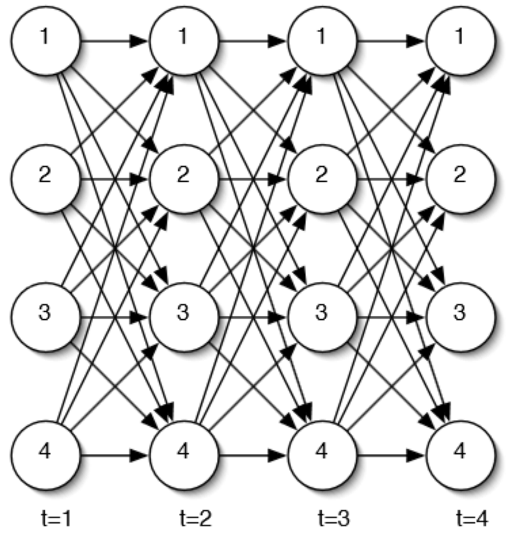
- A better approach is to recognize that many redundant calculations would be made by directly evaluating the previous equation and therefore caching calculations can lead to reduced complexity.
- We implement the cache as a trellis of states at each time step, calculating the cached valued (called α) for each state as a sum over all states at the previous time step.

$$\alpha_t(i) = P(o_1, o_2, o_3, \dots o_t, q_t = s_i | \lambda)$$

$\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1, o_2, o_3, \dots o_t$ such that at time t, state $q_t = s_i$

- So if we work through the trellis filling in the values of α the sum of the final column of the trellis will equal the probability of the observation sequence

A trellis diagram:



Forward Algorithm

- Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad 1 \le i \le N$$

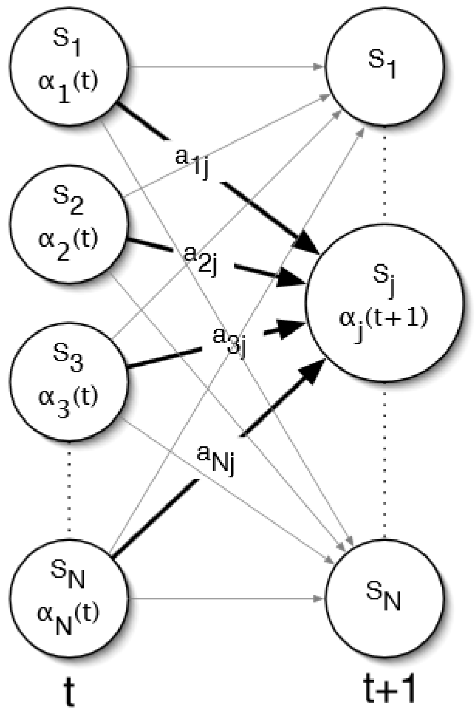
- Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}\right] b_j(o_{t+1}), \qquad 1 \le t \le T-1 \; and \; 1 \le j \le N$$

- Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

- The induction step is the key to the forward algorithm and is depicted in the figure below.
- For each state s_j, $\alpha_j(t)$ stores the probability of arriving in that state having observed the observation sequence up until time t.



- It is apparent that by caching α values the forward algorithm reduces the complexity of calculations involved to N^2Trather than $2TN^T$

- We can also define an analogous backwards algorithm which is the exact reverse of the forwards algorithm with the backwards variable:

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \dots o_T | q_t = s_i, \lambda)$$

as the probability of the partial observation sequence from $t+1$ to T, starting in state s_i

Decoding

- The aim of decoding is to discover the hidden state sequence that was most likely to have produced a given observation sequence.

- One solution to this problem is to use the Viterbi algorithm to find the single best state sequence for an observation sequence.

- The Viterbi algorithm is another trellis algorithm which is very similar to the forward algorithm, except that the transition probabilities are maximized at each step, instead of summed.

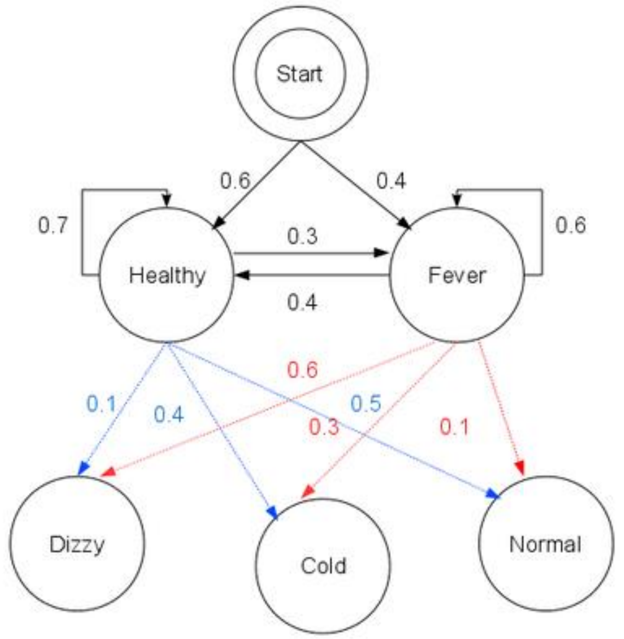
Viterbi Algorithm

- Consider a primitive clinic in a village. People in the village have a very nice property that they are either *healthy* or have a *fever*.
- They can only tell if they have a fever by asking a doctor in the clinic. The wise doctor makes a diagnosis of fever by asking patients how they feel. Villagers only answer that they feel *normal*, *dizzy*, or *cold*.

- Suppose a patient comes to the clinic each day and tells the doctor how she feels. The doctor believes that the health condition of this patient operates as a discrete Markov chain.
- There are two states, "*Healthy*" and "*Fever*", but the doctor cannot observe them directly, that is, they are *hidden* from him. On each day, there is a certain chance that the patient will tell the doctor he has one of the following feelings, depending on his health condition: "*normal*", "*cold*", or "*dizzy*". Those are the *observations*. The entire system is that of a hidden Markov model (HMM).

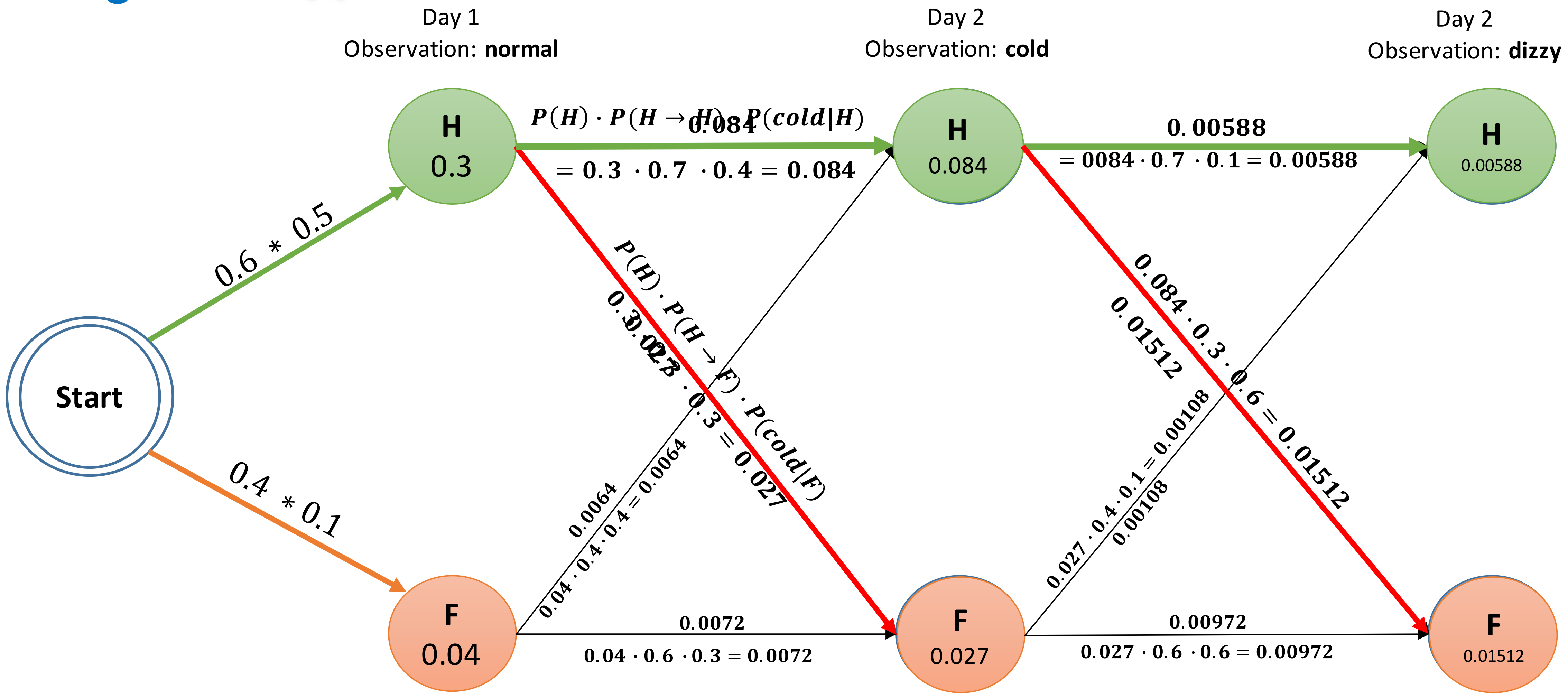
- The doctor knows the villager's general health condition, and what symptoms patients complain of with or without fever on average. In other words, the parameters of the HMM are known.

```
states = ('Healthy', 'Fever')

observations = ('normal', 'cold', 'dizzy')

start_probability = {'Healthy': 0.6, 'Fever': 0.4}

transition_probability = {
   'Healthy' : {'Healthy': 0.7, 'Fever': 0.3},
   'Fever' : {'Healthy': 0.4, 'Fever': 0.6},
   }

emission_probability = {
   'Healthy' : {'normal': 0.5, 'cold': 0.4, 'dizzy': 0.1},
   'Fever' : {'normal': 0.1, 'cold': 0.3, 'dizzy': 0.6},
   }
```



- The patient visits three days in a row and the doctor discovers that on the first day he feels **normal**, on the second day he feels **cold**, on the third day he feels **dizzy**.
- The doctor has a question: what is the most likely sequence of health condition of the patient would explain these observations?

Viterbi Algorithm



Day 1
Observation: **normal**
Day 2
Observation: **cold**
Day 2
Observation: **dizzy**
Start
$0.6 * 0.5$
$0.4 * 0.1$
H
0.3
F
0.04
H
0.084
F
0.027
H
0.00588
F
0.01512
$P(H) \cdot P(H \to H) \cdot P(cold|H)$
0.084
$= 0.3 \cdot 0.7 \cdot 0.4 = 0.084$
$P(H) \cdot P(H \to F) \cdot P(cold|F)$
$0.3 \cdot 0.3 \cdot 0.3 = 0.027$
0.027
0.0064
$0.04 \cdot 0.4 \cdot 0.4 = 0.0064$
0.0072
$0.04 \cdot 0.6 \cdot 0.3 = 0.0072$
0.00588
$= 0084 \cdot 0.7 \cdot 0.1 = 0.00588$
$0.084 \cdot 0.3 \cdot 0.6 = 0.01512$
0.01512
$0.027 \cdot 0.4 \cdot 0.1 = 0.00108$
0.00108
0.00972
$0.027 \cdot 0.6 \cdot 0.6 = 0.00972$

H → H → F are the state transitions

Calculate:
P_start(state) * P_obs("normal")

Calculate:
P(oldState)*P_Trans(OldState→newState) *P(cold|newState)

For each state H/F, select the path with the highest probability

Calculate:
P(oldState)*P_trans(oldState→newState)*P(cold|newState

- First we define:

$$\delta_t(i) = \max_{q_1,q_2,\ldots q_{t-1}} P(q_1, q_{2,\ldots} q_t = s_i, o_1, o_2, \ldots o_t|\lambda)$$

- as the probability of the most probable state path for the partial observation sequence. The Viterbi algorithm is as follows:
- Initialization:

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \le i \le N, \varphi_1(i) = 0$$

- Recursion:

$$\delta_t(j) = \max_{1\le i\le N}\left[\delta_{t-1}(i)\, a_{ij}\right] b_j(o_t), \qquad 2 \le t \le T, \qquad 1 \le j \le N,$$

$$\varphi_t(j) = \arg\max_{1\le i\le N}\left[\delta_{t-1}(i)\, a_{ij}\right], \qquad 2 \le t \le T, 1 \le j \le N$$

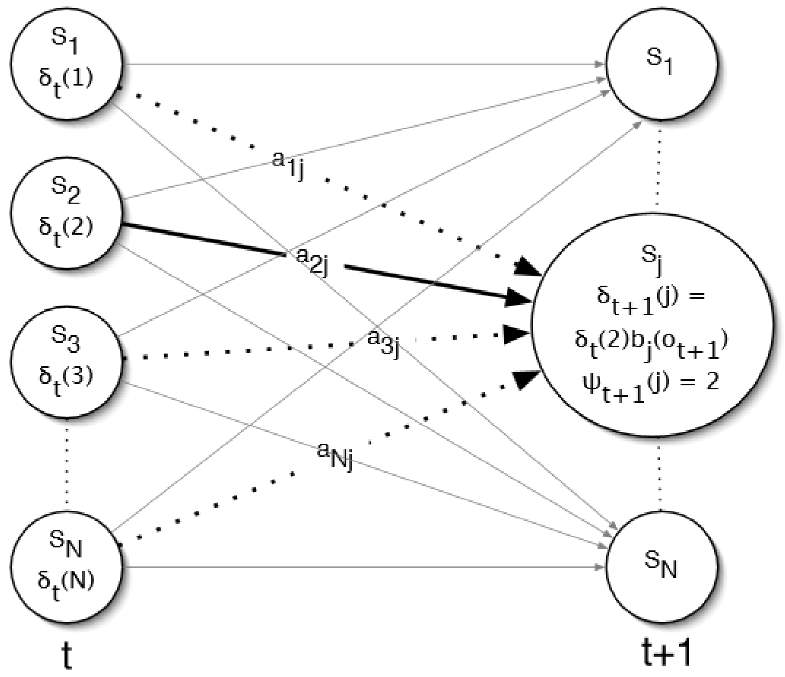
- Termination:

$$P^* = \max_{1\le i\le N}[\delta_T(i)]$$
$$q_t^* = \varphi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1$$

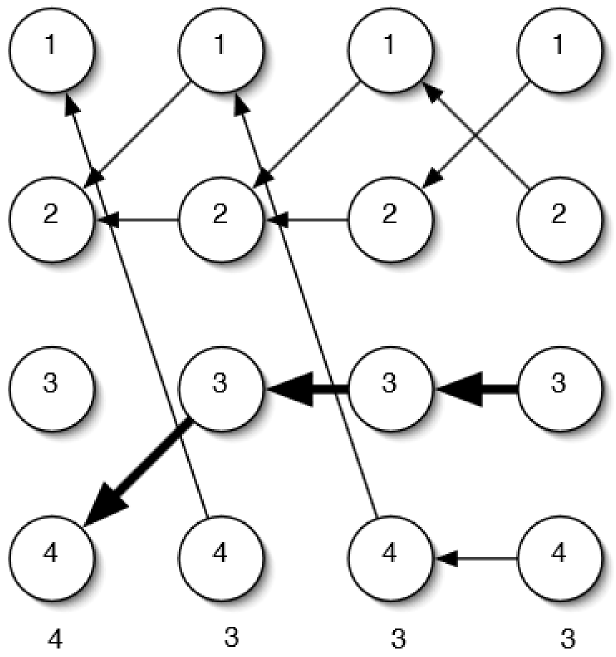
- Optimal state sequence backtracking:

$$q_t^* = \varphi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1.$$

- The main difference with the forward algorithm in the recursions step is that we are maximizing, rather than summing, and storing the state that was chosen as the maximum for use as a backpointer. The backtracking step is shown in the figure. The backtracking allows the best state sequence to be found from the back pointers stored in the recursion step, but it should be noted that there is no easy way to find the second best state sequence.



The recursion step of the viterbi algorithm



The backtracing step of the viterbi algorithm

Learning HMM

- Given a set of examples from a process, we would like to be able to estimate the model parameters $\lambda = (A, B, \pi)$ that best describe that process.
- The easiest solution for creating a model λ is to have a large corpus of training examples, each annotated with the correct classification. he classic example for this approach is PoS tagging.
- We define two sets:
  - $t_1, t_2,\ldots t_N$ is the set of tags, which we equate to the HMM state set $s_1, s_2, \ldots s_N$
  - $w_1, w_2,\ldots w_M$ is the set of words, which we equate to the HMM observation set $v_1, v_2,\ldots v_N$

so with this model we frame part-of-speech tagging as decoding the most probable hidden state sequence of PoS tags given an observation sequence of words

- To determine the model parameters λ, we can use maximum likelihood estimates(MLE) from a corpus containing sentences tagged with their correct PoS tags
- For the transition matrix we use:

$$a_{ij} = P\left(t_i \middle| t_j\right) = \frac{Count(t_i, t_i)}{Count(t_i)}$$

where $Count(t_i, t_j)$ is the number of times t_j followed t_i in the training data.

- For the observation matrix we use:

$$b_j(k) = P\left(w_k \middle| t_j\right) = \frac{Count\left(w_k, t_j\right)}{Count\left(t_j\right)}$$

where $Count(w_k, t_j)$ is the number of times w_k was tagged t_j in the training data.

- Lastly, the initial probability distribution:

$$\pi_i = P(q_1 = t_i) = \frac{Count(q_1 = t_i)}{Count(q_1)}$$

- In practice when estimating a HMM from counts it is normally necessary to apply smoothing in order to avoid zero counts and improve the performance of the model on data not appearing in the training set.

Learning Algorithm

- Training HMM to encode observation sequence such that HMM should identify a similar observation sequence in future
- Find $\lambda = (A, B, \pi)$, maximising $P(O|\lambda)$
- General algorithm:
  1. Initialise: λ_0
  2. Compute new model λ, using λ_0 and observed sequence O
  3. Then
  4. Repeat steps 2 and 3 until:

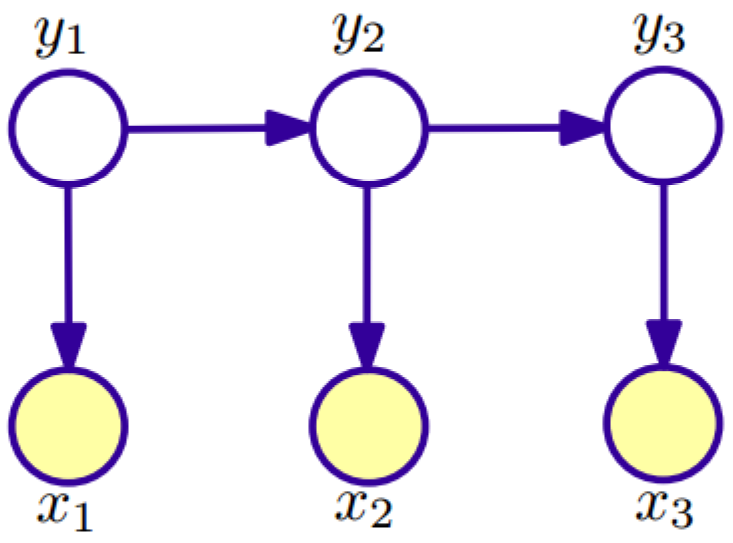
$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

i.e. maximizing log-likelihood.

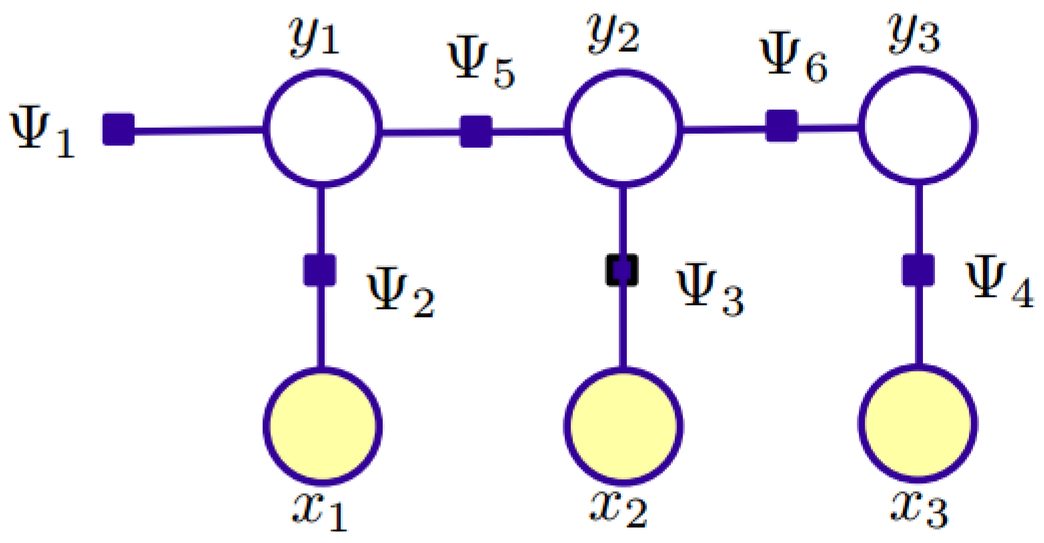
Baum-Welch Algorithm for training HMM

•

HMM Factor Graph



(a) Independency graph



(b) Factor graph

Independency and factor graph for the Hidden Markov Model

Graphical Models

- The underlying probability distributions of probabilistic models can be represented in a graphical form, this is why they are often called *probabilistic graphical models*.

- A *probabilistic graphical model* is a diagrammatic representation of a probability distribution. In such a graph **there is a node for each random variable**. The **absence of an edge between two variables represent conditional independence between those variables**.

- Conditional independence means that two random variables a and b are independent given a third random variable c if they are independent in their conditional probability distribution,

$$p(a, b|c) = p(a|b) \cdot p(b|c)$$

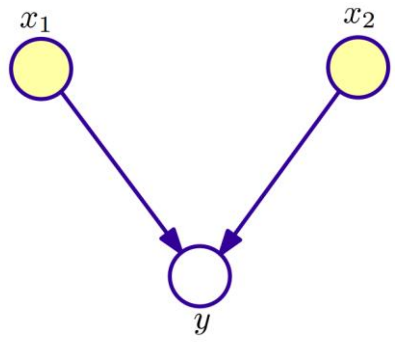
*Note that in contrast two random variables a and b are *statistically* independent if and only if $p(a, b) = p(a)p(b)$

Graphical models contd.

- From such graphs, also called independency graphs, one can read the conditional independence properties of the underlying distribution.
- Conditional independence is an important concept as it can be used to decompose complex probability distributions into a product of factors, each consisting of the subset of corresponding random variables.
- In general, the decomposition, in fact a factorization of a probability distribution, is written as the product of its factors ψ_s, with $\bar{v}_s$ the subset of the respective random variables constituting such a factor:

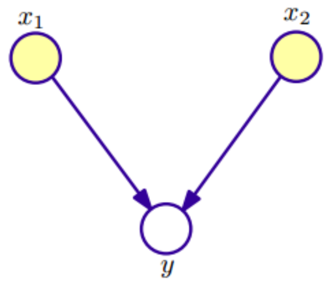
$$p(\bar{v}) = \prod_{s} \psi_s(\bar{v}_s) \quad ★$$

- Let $G = (V, E)$ be a graph with vertexes V and edges E. In an independency graph (*for example the one shown in figure*), the vertexes $V = X \cup Y$, with X and Y sets of random variables, are depicted by circles.
  - X will typically be considered as the set of input or observation variables (*shaded circles*), and
  - Y as the set of output variables (*empty nodes*).
  - An independency graph can have directed or undirected edges, depending on the kind of graphical model it represents

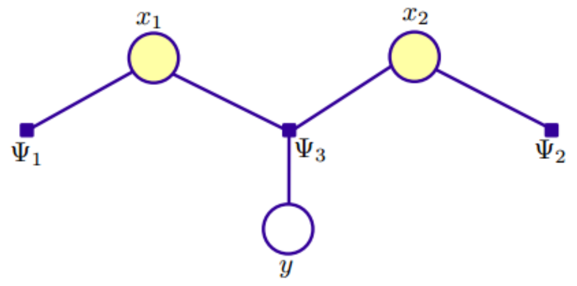


(a) Independency graph

- In a **factor graph**, such as the one shown in below figure (b), the circles represent, as in an independency graph, the random variables of the underlying distribution, depicted by circles.
- Further, a factor graph contains factor nodes, depicted by small, filled squares, which represent the factors ψ_s.
- The edges are always undirected, linking the random variables to the factor nodes.
- A factor ψ_s includes all random variables to which the respective factor node is directly connected by an edge.
- Thus, a factor graph represents more explicitly the factorization of the underlying probability distribution.
- As an example, assume a probability distribution $p(x_1, x_2, y)$ to factorize as $p(\bar{x}) = p(x_1)p(x_2)p(y|x_1, x_2)$. It has factors $\psi_1(x_1) = p(x_1)$, $\psi_2(x_2) = p(x_2)$ and $\psi_3(y) = p(y|x_1, x_2)$. Here x_1 and x_2 are conditionally independent given y. Below figure shows an independency graph and a factor graph representing this distribution:



(a) Independency graph



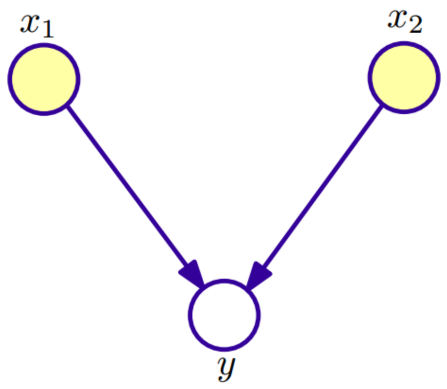
(b) Factor graph

Directed Graphical Models

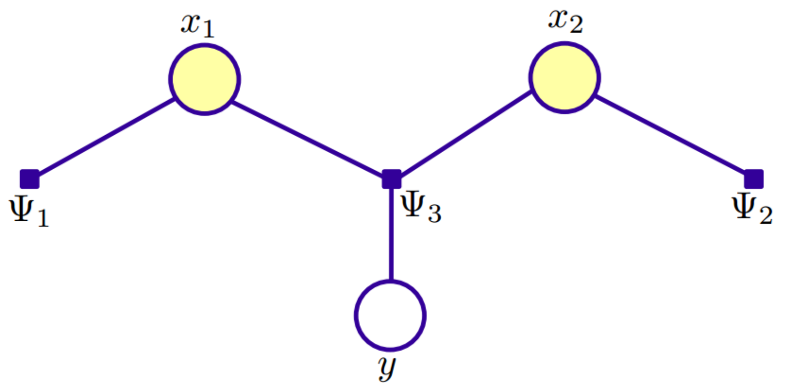
- A joint distribution $p(\bar{v})$ can be factorized into the product of conditional distributions for each node v_k, so that each such conditional distribution is conditioned on its set of *parent nodes* v_k^p.

$$p(\bar{v}) = \prod_{k=1}^{K} p(v_k | v_k^p)$$ ★

For example, the distribution $p(x_1, x_3, y)$can be given as:



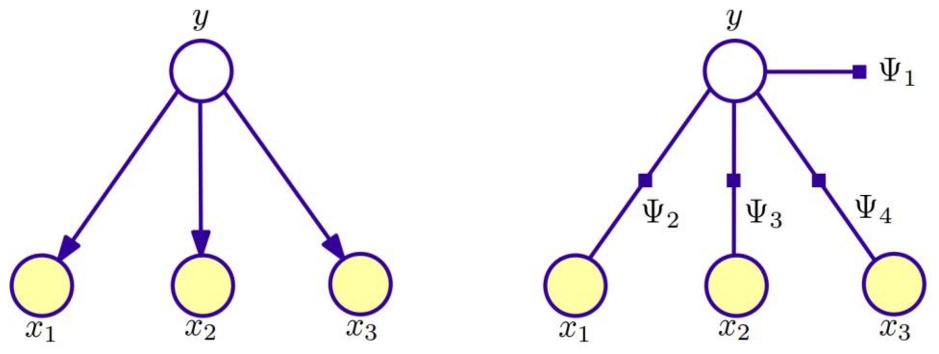
(a) Independency graph



(b) Factor graph

- Naïve Bayes

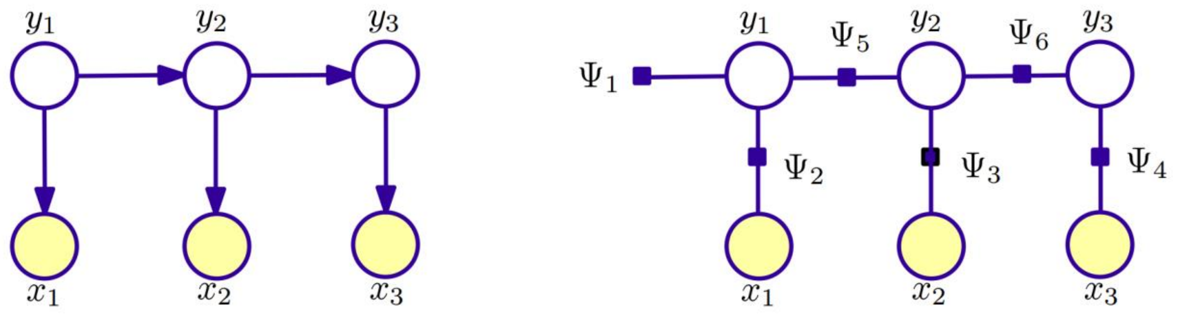
$$p(y|\vec{x}) \propto p(y,\vec{x}) = p(y)\prod_{i=1}^{m} p(x_i|y)\,.$$



(a) Independency graph (b) Factor graph

- Hidden Markov Model

$$p(y,x) = \prod_{t=1}^{T} p(y_t|y_{t-1}) \cdot p(x_t|y_t)$$



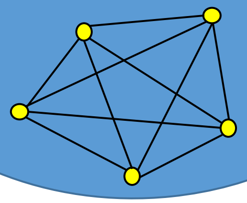
(a) Independency graph (b) Factor graph

Undirected Graphical Models

- A probability distribution can be represented by an undirected gra product of non-negative functions of the maximal cliques of G. Th performance in a way that conditionally independent nodes do same factor, that means that they belong to different cliques:

$$p(\bar{v}) = \frac{1}{Z}\prod_{c \epsilon C}\psi_c(\bar{v}_c)$$

A clique is any subgraph in which all vertices are neighbors



- The factors $\psi_c \geq 0$ are called potential functions of the random variable $\bar{v}_c$within a clique $c\epsilon C$.
- The potential functions may be any arbitrary function. Due to this generality the potential functions do not necessarily have to be probability functions.
- This is in contrast to directed graphs where the join distribution is factorized into a product of conditional distributions.
- This, normalization of the product of potential functions is necessary to achieve a proper probability measure. This yields the normalization factor Z.
- Calculating Z is one of the main challenges during parameter learning as summing over all possible variables is necessary:

$$Z = \sum_{\bar{v}}\prod_{c \epsilon C}\psi_c(\bar{v}_c)$$

- The Maximum Entropy Model is given by:

$$p^*_{\vec{\lambda}}(y|x) = \frac{1}{Z_{\vec{\lambda}}(x)} \exp(\sum_{i=1}^{m} \lambda_i \, f_i(x,y))$$

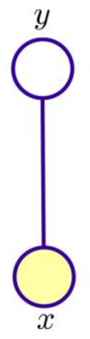
If we take the summation out of exponential, we get:

$$p^*_{\vec{\lambda}}(y|x) = \boxed{\frac{1}{Z_{\vec{\lambda}}(x)}} \boxed{\prod_{i=1}^{m} \exp(\lambda_i \, f_i(x,y))}$$

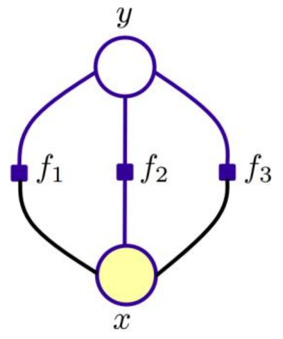
Product of non-negative potentials

Normalization factor

- In such log-linear models, potential functions are formulated as the exponential function of weighted features. Such a formulation is frequently used because it fulfills the requirements of strict positivity of the potential functions.



(a) Independency graph



(b) Factor graph

Conditional Random Field (CRF)

Conditional Random Field (CRF)

- Introduced by Lafferty et al. (2001), Conditional Random Fields (CRF) are probabilistic models for computing the probability $p(\bar{y}|\bar{x})$ of a possible output $\bar{y} = (y_1, y_2, \dots, y_n)$ given the input $\bar{x} = (x_1, x_2, \dots, x_n)$ which is also called the observation.
- A CRF in general can be derived from formula:

$$p(\bar{v}) = \frac{1}{Z} \prod_{c \epsilon C} \psi_c(\bar{v_c})$$ *.... (see previous slide for this equation)*

- The conditional probability $p(\bar{y}|\bar{x})$ can be written as:

$$p(\bar{y}|\bar{x}) = \frac{p(\bar{x}, \bar{y})}{p(\bar{x})}$$
$$= p(\bar{x}, \bar{y}) / \sum_{\bar{y}'} p(\bar{y}', \bar{x})$$
$$= \frac{\frac{1}{Z} \prod_{c \epsilon C} \psi_c(\bar{x}_c, \bar{y}_c)}{\frac{1}{Z} \sum_{\bar{y}'} \prod_{c \epsilon C} \psi_c(\bar{x}_c, \bar{y}_c)}$$

From this, the general model formulation of CRF's is derived as:

$$\boxed{p(\bar{y}|\bar{x}) = \frac{1}{Z(\bar{x})} \prod_{c \epsilon C} \boxed{\psi_c(\bar{x}_c, \bar{y}_c)}}$$

Potential function

Let’s start with HMM…

- We know, HMM is given by:

$$p(y,x) = \prod_{t=1}^{T} p(y_t|y_{t-1}) \cdot p(x_t|y_t)$$

- Now let's write the emission and transition probabilities in terms of the matrices A and B

$$p(y,x) = \prod_{t=1}^{T} \boldsymbol{A}(y_t|y_{t-1}) \cdot \boldsymbol{B}(x_t|y_t)$$

- Now we can apply an exponential-logarithm identity. Note that nothing has changed:

$$p(y,x) = e^{ln}\left(\prod_{t=1}^{T} \boldsymbol{A}(y_t|y_{t-1}) \cdot \boldsymbol{B}(x_t|y_t)\right)$$

- Now we take that the logarithm of the product:

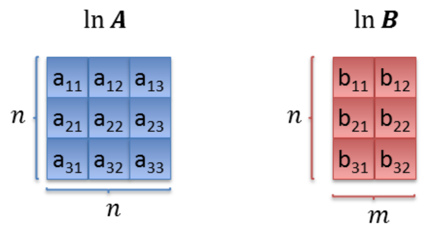
$$p(y,x) = exp\left(\sum_{t=1}^{T} log(\boldsymbol{A}(y_t|y_{t-1}) \cdot \boldsymbol{B}(x_t|y_t))\right)$$

- We proceed simplifying (or perhaps complicating even further) the expression:

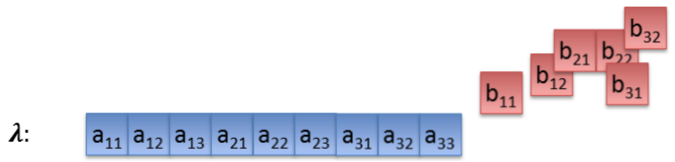
$$p(y,x) = exp\left(\sum_{t=1}^{T} \log \boldsymbol{A}(y_t|y_{t-1}) + \sum_{t=1}^{T} \log \boldsymbol{B}(x_t|y_t)\right)$$

$$p(y,x) = exp\left(\sum_{t=1}^{T} \log \boldsymbol{A}(y_t|y_{t-1}) + \sum_{t=1}^{T} \log \boldsymbol{B}(x_t|y_t)\right)$$

- Note that the two parts of the formula have an extremely similar form. How could we remove this duplication?
- We have two matrices with parameters to specify our model:



- If we "linearize" those matrices into a single parameter vector, we will only have one structure to maintain:



- But now, how could we access individual elements of this vector as if it were the original matrices? Well, we may do so by using *indicator functions*.
- Those are feature functions that activate only when their inputs are equal to some pre-defined values. Let's consider two classes of functions, one that activates for elements in the transition matrix and one that activate for elements of the emissions matrix.

$$f_{edge}(y_t, y_{t-1}, x; i, j) = \mathbf{1}_{\{y_t=\mathrm{i}\}}\mathbf{1}_{\{y_{t-1}=\mathrm{j}\}}$$
$$f_{node}(y_t, y_{t-1}, x; i, o) = \mathbf{1}_{\{y_t=\mathrm{i}\}}\,\mathbf{1}_{\{x_t=\mathrm{o}\}}$$

- Next, we instantiate a member from those functions for each corresponding parameter in our parameter vector:

$\boldsymbol{f}$:	f_e i=1 j=1	f_e i=1 j=2	f_e i=1 j=3	f_e i=2 j=1	f_e i=2 j=2	f_e i=2 j=3	f_e i=3 j=1	f_e i=3 j=2	f_e i=3 j=1	f_n i=1 o=1	f_n i=1 o=2	f_n i=2 o=1	f_n i=2 o=2	f_n i=3 o=1	f_n i=3 o=2
$\boldsymbol{\lambda}$:	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	b_{11}	b_{12}	b_{21}	b_{22}	b_{31}	b_{32}

$$k = n \times n + n \times m$$

- And now, when things seems to have gotten extremely hairier than before.... ☺

$$p(y,x) = exp\left(\sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}1_{\{y_t=i\}}1_{\{y_{t-1}=j\}} + \sum_{t=1}^{T}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}1_{\{y_t=i\}}1_{\{y_{t-1}=j\}}\right)$$

$$= exp\left(\sum_{t=1}^{T} a_{ij}f_{ij}(y_t, y_{t-1}, x_t) + \sum_{t=1}^{T} b_{ij}f_{ij}(y_t, y_{t-1}, x_t)\right)$$

.... we suddenly realize the structures are so similar, we can process then in the same way, lets create a common "interface" for accessing the matrix values:

$$p(y,x) = \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

- As we need to get proper probability measure, we need to add the normalization factor:

$$p(y,x) = \frac{1}{Z}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

- Up till now, nothing had changed in the formula. This is completely equivalent to HMM.

$$p(y,x) = \frac{1}{Z}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

In the previous equation we were computing the **joint probability** of the random variables y and x.

- Let’s see where we get if we try to get the **conditional probability**:

$$p(y|x) = \frac{p(x,y)}{p(x)}$$

$$p(y|x) = \frac{1}{Z(x)}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K}\underbrace{\lambda_k}_{\text{Weight vector}} \overbrace{f_k(y_t, y_{t-1}, x_t)}^{\text{Feature functions}}\right)$$

- Where $Z(x)$ is given by:

$$Z(x) = \sum_{y}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

This is nothing but the **Linear-Chain CRF**!

Linear Chain CRF

Remember the general for

$$p(\bar{y}|\bar{x}) = \frac{1}{Z(\bar{x})}\prod_{c\epsilon C}\psi_c(\bar{x}_c,\bar{y}_c)$$

- **Definition**: Let Y, X be random vectors, $\lambda = \lambda_k$ be a parameter vecto t of real-valued feature functions. Then a linear-chain conditional rand that takes the form:

$$p(y|x) = \frac{1}{Z(x)}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

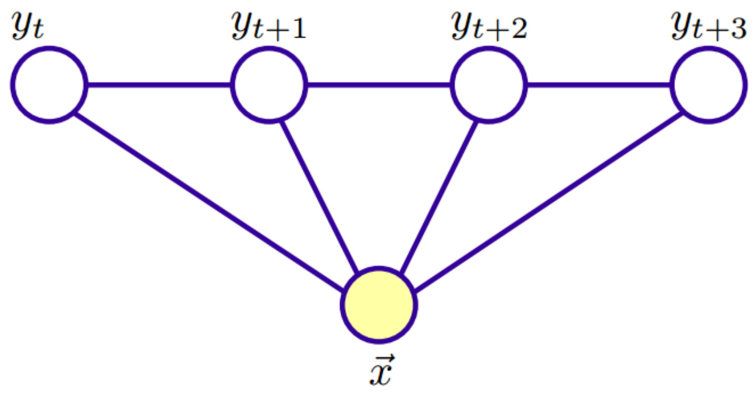
Potentials ψ_c

- Where $Z(x)$ is given by:

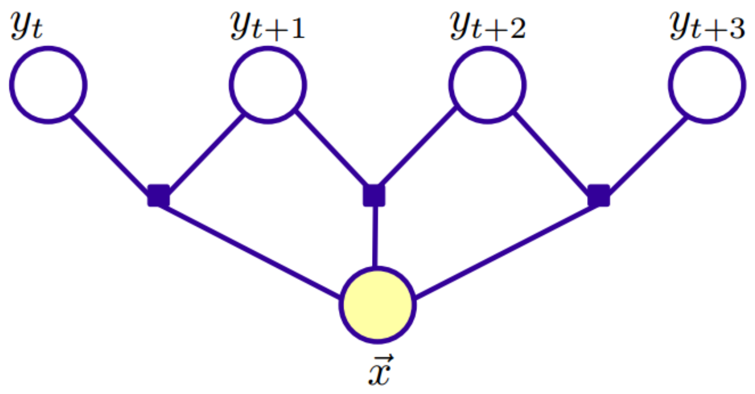
$$Z(x) = \sum_{y}\prod_{t=1}^{T} exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

- To indicate in the definition of linear-chain CRF that each feature function can depend on observations from any time step, we have written the observation argument to f_k as a vector x_t, which should be understood as containing all the components of the global observations x that are needed for computing features at time t.
- For example, if the CRF uses the next word x_{t+1} as a feature, then the feature vector x_t is assumed to include the identity of word x_{t+1}.
- Note that the normalization constant $Z(x)$ sums over all possible state sequences, an exponentially large number of terms. Nevertheless, it can be computed efficiently by forward-backward.

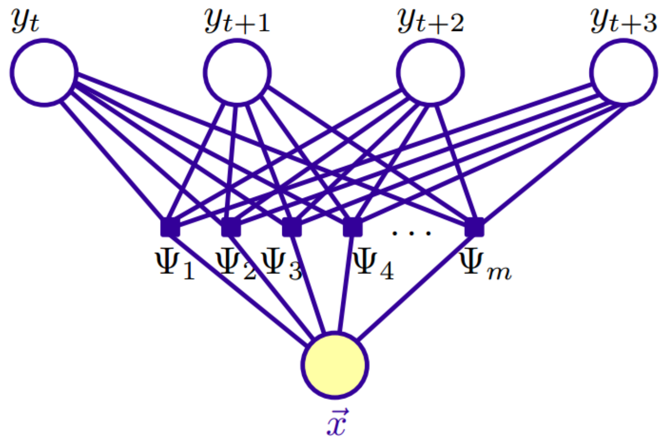
Graphical Representation of *Linear-Chain* CRF



(a) Independency graph



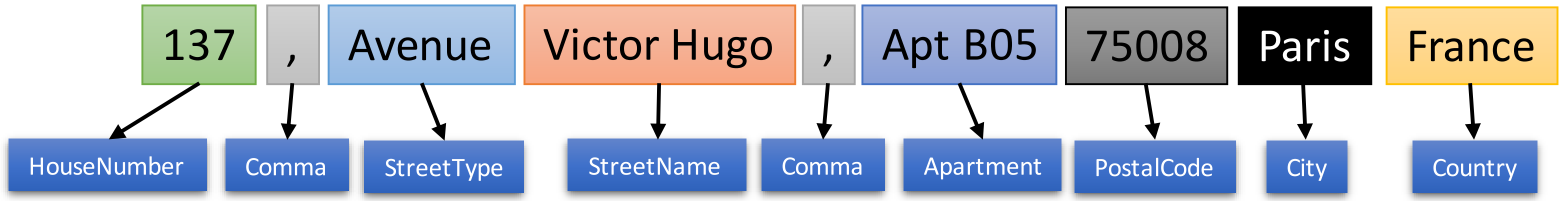
(b) Factor graph



Alternative interpretation of a linear-chain CRF

Full Address Parsing

- Given a raw address, we are interested in tagging (*labeling*) specific parts of the address
- For example, given "*137 , Avenue Victor Hugo , Apt B05 75008 Paris France*" we would like to tag each token in the address as:

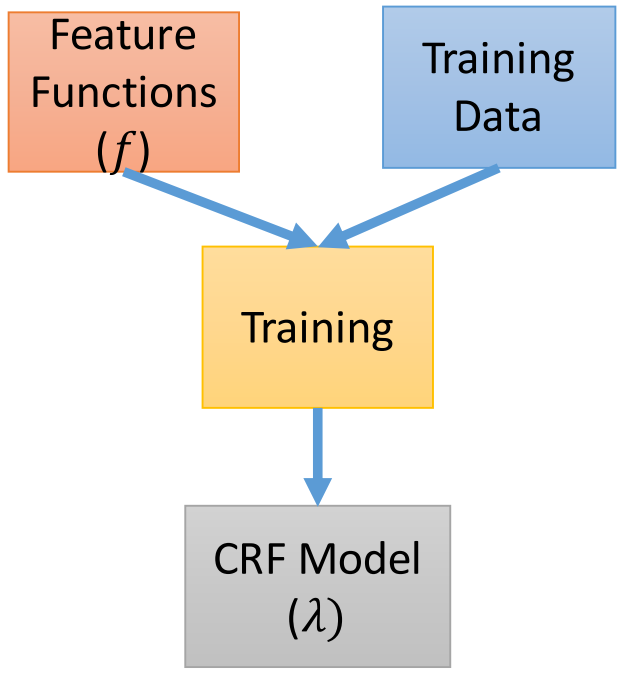


- Looking at the CRF equation: $p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)$
  - y – are the labels like HouseNumber, Comma, City etc
  - f – are the feature functions
  - λ – are the weights associated with the features
- λ - the weight vector is computed during the training phase
- f - the features functions must be selected by us

Feature Selection

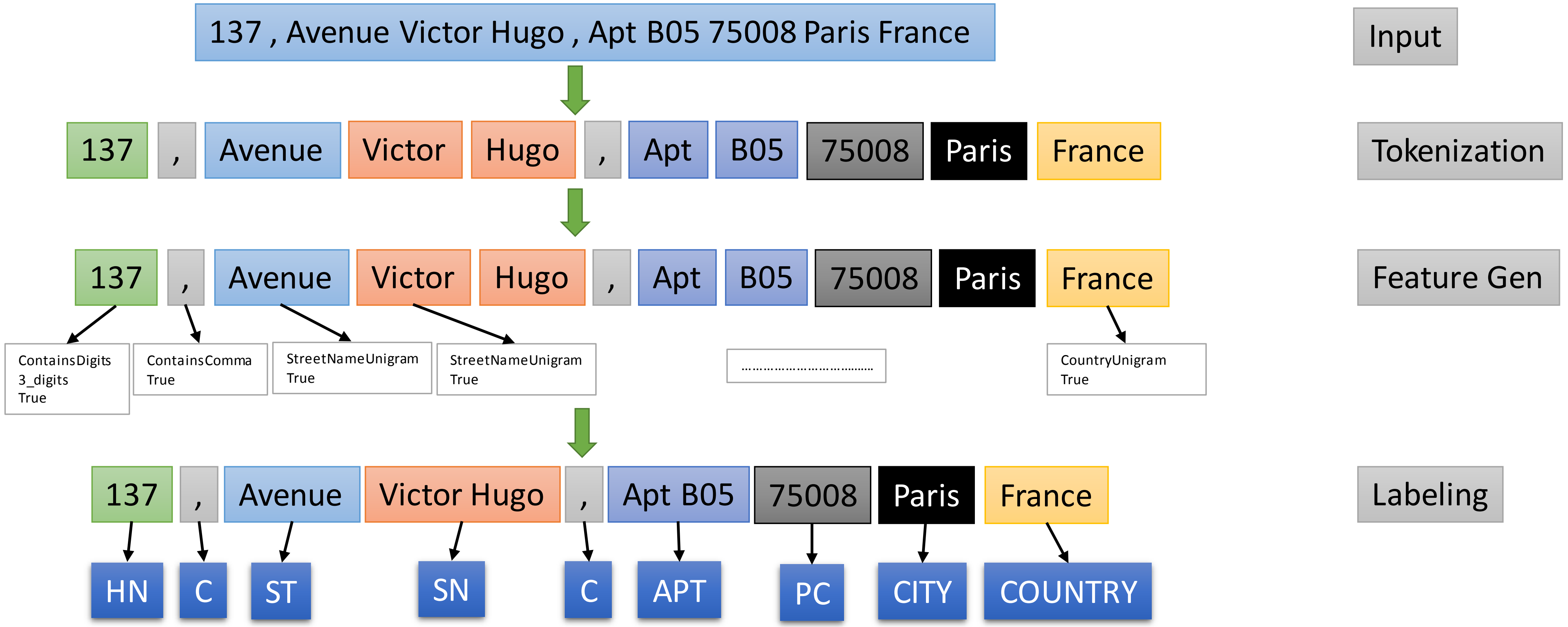
- Feature selection is an important steps in before training a CRF
- Typically features are of two forms:
  - Regular expression features
  - Lexicon features
- Regular expression features are simply bunch of regular expressions that model the pattern of your problem domain, for example in the case of full address parsing we could have:
  - Contains Comma
  - Contains Dot
  - Is USA Post Code
  - Contains Two Alphabets
  - Contains 5 digits
  - …etc.
- Lexicons features are simply list of lexicons. This can be considered as the prior knowledge provided to the CRF. For example, the list of lexicons could be:
  - List of city names
  - List of state names
  - List of street names
  - List of country names
  - List of spelled ordinal numbers
  - …etc.
- It's always better to have more feature functions to model the problem domain

Typical CRF Model Training Pipeline



- Various algorithms are available for training CRF (i.e. learning the weight vector or parameter estimation)
  - Perceptron learning
  - Stochastic Gradient Descent

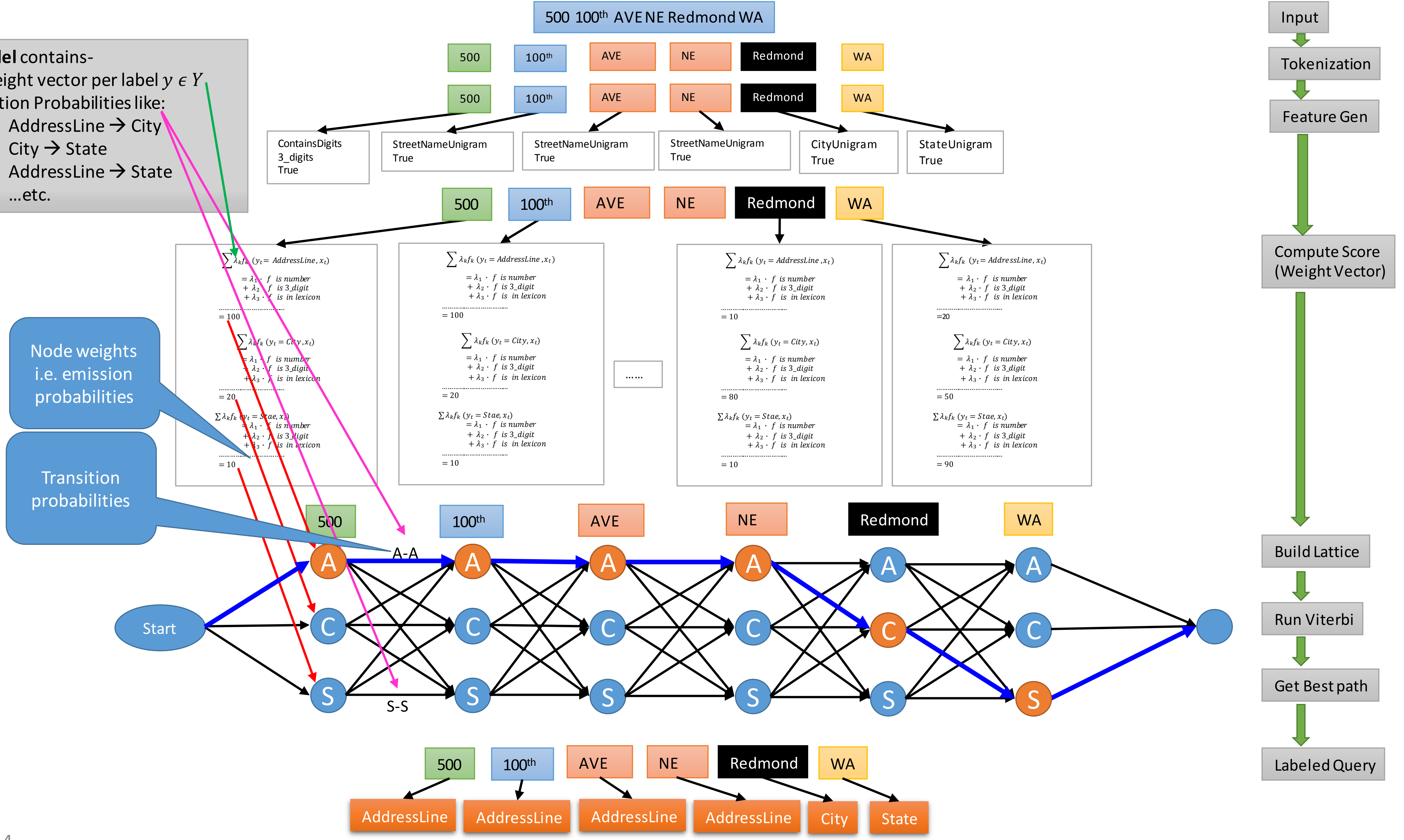
Example (Decoding)



Closer look

CRF Model contains-

- λ – weight vector per label $y \in Y$
- Transition Probabilities like:
  - AddressLine → City
  - City → State
  - AddressLine → State
  - ...etc.


500 100th AVE NE Redmond WA
500
100th
AVE
NE
Redmond
WA
ContainsDigits 3_digits True
StreetNameUnigram True
StreetNameUnigram True
StreetNameUnigram True
CityUnigram True
StateUnigram True
$\sum \lambda_k f_k (y_t = AddressLine, x_t) = \lambda_1 \cdot f$ is number $+ \lambda_2 \cdot f$ is 3_digit $+ \lambda_3 \cdot f$ is in lexicon $= 100$
$\sum \lambda_k f_k (y_t = City, x_t) = \lambda_1 \cdot f$ is number $+ \lambda_2 \cdot f$ is 3_digit $+ \lambda_3 \cdot f$ is in lexicon $= 20$
$\sum \lambda_k f_k (y_t = Stae, x_t) = \lambda_1 \cdot f$ is number $+ \lambda_2 \cdot f$ is 3_digit $+ \lambda_3 \cdot f$ is in lexicon $= 10$
$= 100$
$= 20$
$= 10$
……
$= 10$
$= 80$
$= 10$
$=20$
$= 50$
$= 90$
Node weights i.e. emission probabilities
Transition probabilities
A-A
S-S
Start
A
C
S
AddressLine
AddressLine
AddressLine
AddressLine
City
State
Input
Tokenization
Feature Gen
Compute Score (Weight Vector)
Build Lattice
Run Viterbi
Get Best path
Labeled Query

2/25/2014

Semi-Markov CRF (Paper)

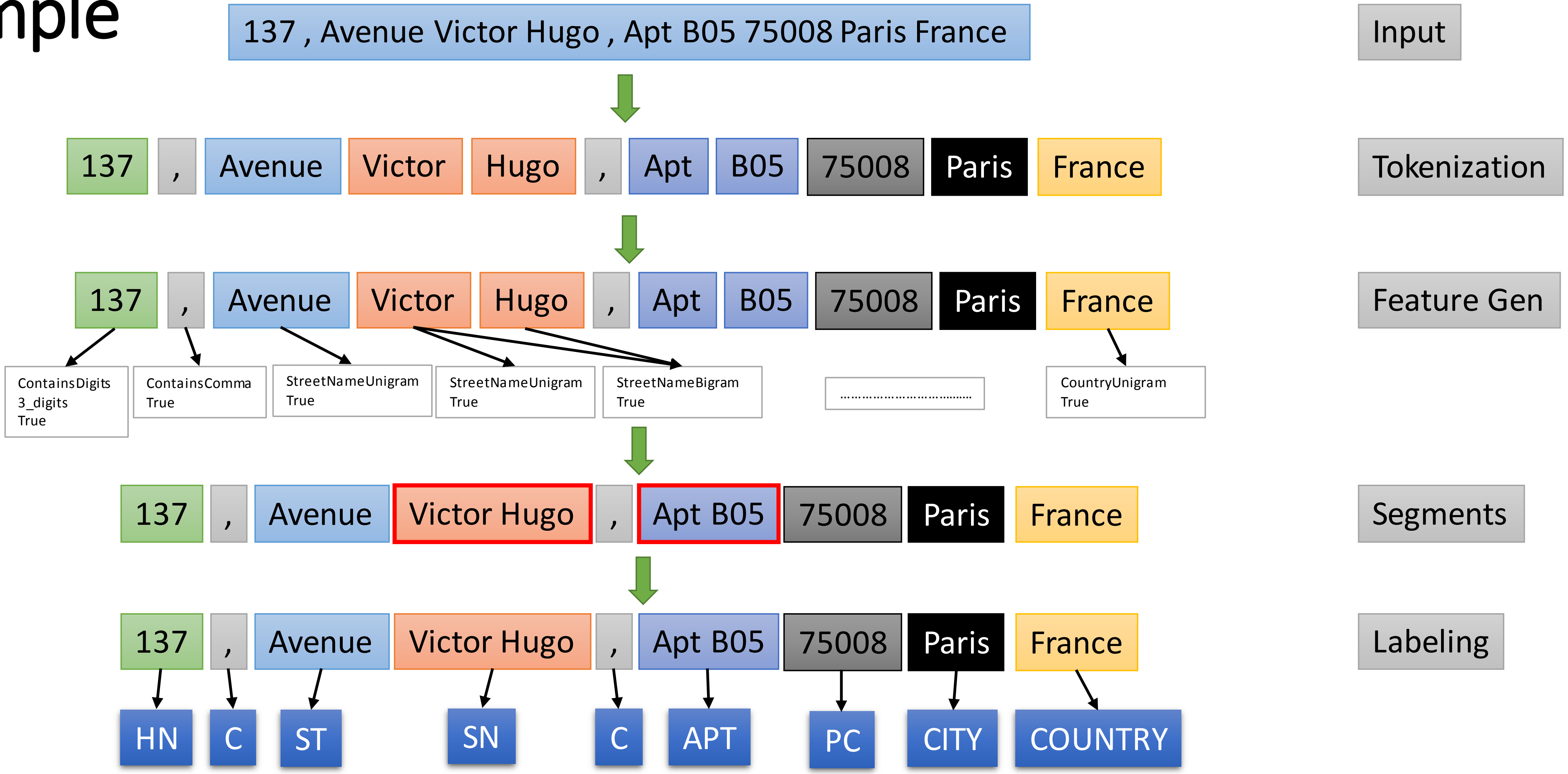
- We've seen the CRF of the form: $$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$
- Let $s = s_1, s_2, s_3, \dots s_n$ denote a segmentation of x, where segment $s_j = (t_j, u_j, y_j)$ consist of a $start\ position\ t_j$, and $end\ position\ u_j$ and a label $y_j \epsilon\ Y$
- Conceptually, a segment means that the tag y_j is given to all x_i's between $i = t_j$ and $i = u_j$ (*inclusive*)
- We assume segments have positive lengths and adjacent segments touch
- Assume a vector g of segment feature functions $g = g_1, g_2, g_3, \dots, g_k$, each of which maps a triple (j, x, s) and define $G(x, s) = \sum_j^{|s|} g(j, x, s)$
- We assume that every component g_k of g is a function only of x, s_j and the label y_{j-1} associated with the preceding segment s_{j-1}, this can be written as: $$g_k(j, x, s) = g'_k(y_j, y_{j-1}, x, t_j, u_j)$$

A semi-CRF is then an estimator of the form:

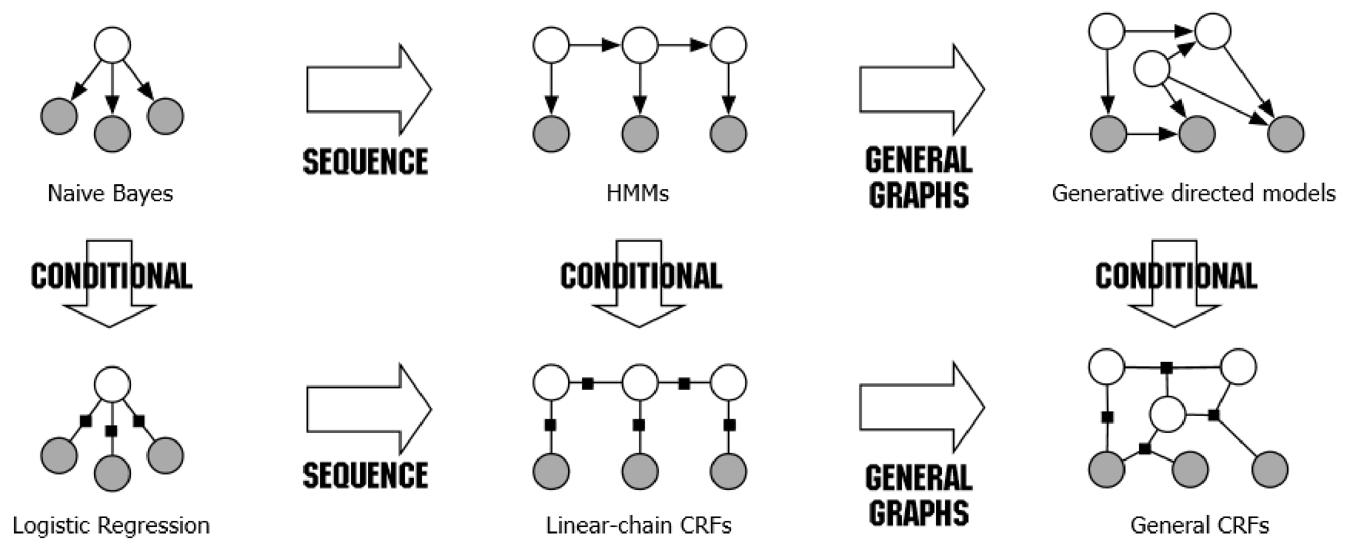
$$p(s|x) = \frac{1}{Z(x)} \prod_{j=1}^{J} \underbrace{exp\left(\sum_{k=1}^{K} \lambda_k g_k(y_j, y_{j-1}, x, t_j, u_j)\right)}_{\text{Potential } \psi}$$

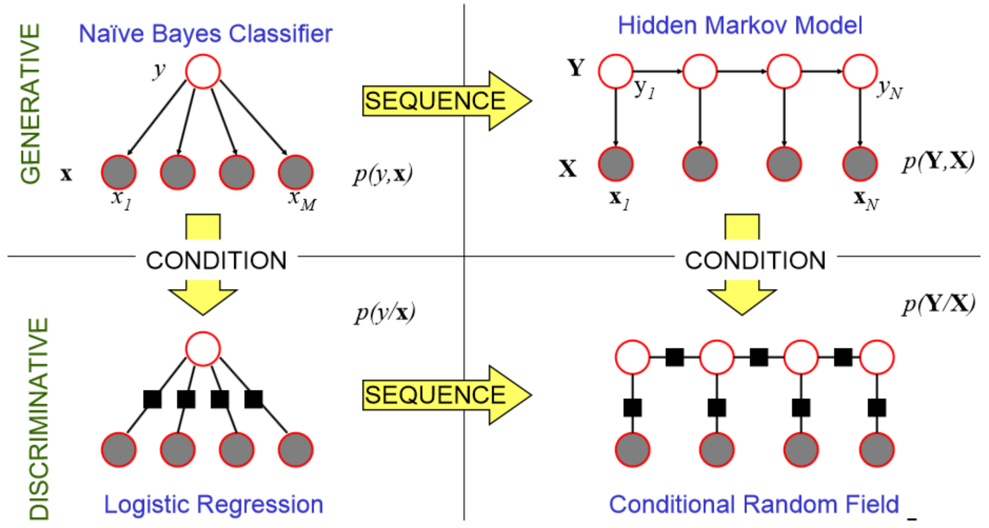
$$Z(x) = \sum_{y} \prod_{t=1}^{T} exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right)$$

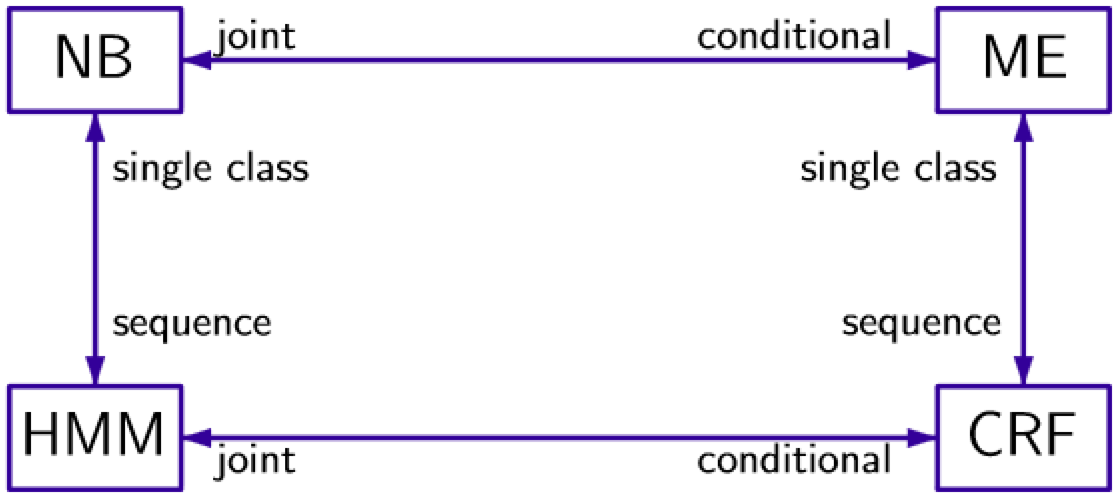
Example


137 , Avenue Victor Hugo , Apt B05 75008 Paris France
Input
137
,
Avenue
Victor
Hugo
,
Apt
B05
75008
Paris
France
Tokenization
Feature Gen
ContainsDigits
3_digits
True
ContainsComma
True
StreetNameUnigram
True
StreetNameUnigram
True
StreetNameBigram
True
……………………………
CountryUnigram
True
Victor Hugo
Apt B05
Segments
Labeling
HN
C
ST
SN
C
APT
PC
CITY
COUNTRY

Relationship between NB, LR, HMM & CRF







Note: Logistic Regression and Maximum Entropy are same.

Comparison of NB, HMM, ME and CRF

Naïve Bayes	ME/LR	HMM	CRF
Joint Probability $p(y, x)$	Conditional Probability $p(y\|x)$	Joint Probability $p(\bar{y}, \bar{x})$	Conditional Probability $p(\bar{y}\|\bar{x})$
Single class Output	Single class output	Multi-class output	Multi-class output
Generative model	Discriminative model	Generative model	Discriminative model

References

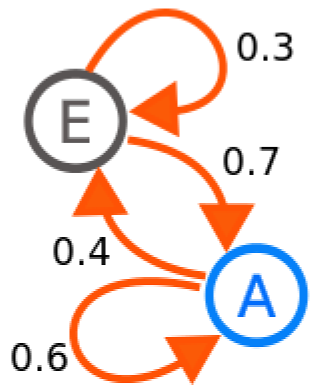
- C# implementation of naïve bayes on MSDN
- Naïve Bayes -Eamonn Keogh (UCR)
- http://www.codeproject.com/Articles/559535/Sequence-Classifiers-in-Csharp-Part-II-Hidden-Cond
- Wikipedia
- Semi-Markov Conditional Random Fields for Information Extraction
- An Introduction to Conditional Random Fields for Relational Learning
- A Maximum Entropy Approach to Natural Language Processing
- A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition
- Advance NLP: Conditional Random Fields
- Classical Probabilistic Models and Conditional Random Fields

Thank you!

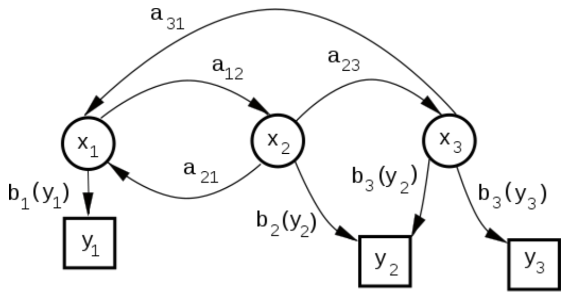
Appendix

Markov Chain

- A Markov chain (discrete-time Markov chain or DTMC) named after Andrey Markov, is a mathematical system that undergoes transitions from one state to another, between a finite or countable number of possible states.
- It is a random process usually characterized as memoryless: the next state depends only on the current state and not on the sequence of events that preceded it.
- This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes.



Markov Process



- In probability theory and statistics, a Markov process or Markoff process, named after the Russian mathematician Andrey Markov, is a stochastic process satisfying a certain property, called the Markov property. A Markov process can be thought of as 'memoryless': loosely speaking, a process satisfies the Markov property if one can make predictions for the future of the process based solely on its present state just as well as one could knowing the process's full history. I.e., conditional on the present state of the system, its future and past are independent.

- Markov processes arise in probability and statistics in one of two ways. A stochastic process, defined via a separate argument, may be shown mathematically to have the Markov property, and as a consequence to have the properties that can be deduced from this for all Markov processes. Alternately, in modelling a process, one may assume the process to be Markov, and take this as the basis for a construction. In modelling terms, assuming that the Markov property holds is one of a limited number of simple ways of introducing statistical dependence into a model for a stochastic process in such a way that allows the strength of dependence at different lags to decline as the lag increases.

- Often, the term Markov chain is used to mean a Markov process which has a discrete (finite or countable) state-space. Usually a Markov chain would be defined for a discrete set of times (i.e. a discrete-time Markov chain) although some authors use the same terminology where "time" can take continuous values.

Stochastic Process

- In probability theory, a stochastic process, or sometimes random process (widely used) is a collection of random variables; this is often used to represent the evolution of some random value, or system, over time. This is the probabilistic counterpart to a deterministic process (or deterministic system). Instead of describing a process which can only evolve in one way (as in the case, for example, of solutions of an ordinary differential equation), in a stochastic or random process there is some indeterminacy: even if the initial condition (or starting point) is known, there are several (often infinitely many) directions in which the process may evolve.

- In the simple case of discrete time, a stochastic process amounts to a sequence of random variables known as a time series (for example, see Markov chain). Another basic type of a stochastic process is a random field, whose domain is a region of space, in other words, a random function whose arguments are drawn from a range of continuously changing values.
- One approach to stochastic processes treats them as functions of one or several deterministic arguments (inputs, in most cases regarded as time) whose values (outputs) are random variables: non-deterministic (single) quantities which have certain probability distributions. Random variables corresponding to various times (or points, in the case of random fields) may be completely different. The main requirement is that these different random quantities all have the same type. Type refers to the codomain of the function. Although the random values of a stochastic process at different times may be independent random variables, in most commonly considered situations they exhibit complicated statistical correlations.

Likelihood Function

- In statistics, a likelihood function (often simply the likelihood) is a function of the parameters of a statistical model, defined as follows: the likelihood of a set of parameter values, θ, given some observed outcomes, x, is equal to the probability of those observed outcomes given those parameter values, i.e.

$$\mathcal{L}(\theta|x) = P(x|\theta)$$

- In non-technical parlance, "likelihood" is usually a synonym for "probability." But in statistical usage, a clear technical distinction is made depending on the roles of the outcome or parameter.

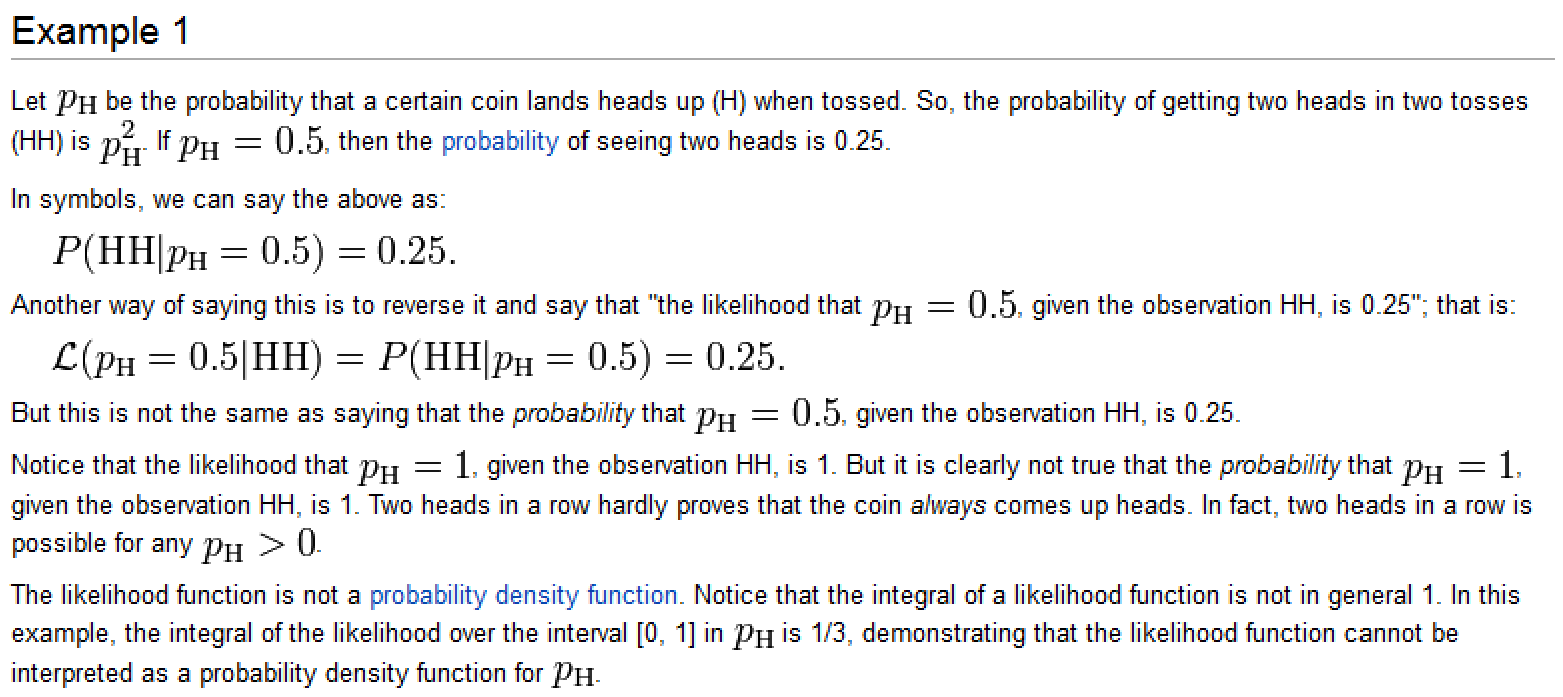
- Use probability when describing a function of the outcome given a fixed parameter value.
  - “Given that I have flipped a coin 100 times and it is a fair coin, what is the probability of it landing heads-up every time?"
- Use likelihood when describing a function of a parameter given a fixed outcome.
  - "Given that I have flipped a coin 100 times and it has landed heads-up 100 times, what is the likelihood that the coin is fair?"

Log-Likelihood Function

- For many applications involving likelihood functions, it is more convenient to work in terms of the natural logarithm of the likelihood function, called the log-likelihood, than in terms of the likelihood function itself. Because the logarithm is a monotonically increasing function, the logarithm of a function achieves its maximum value at the same points as the function itself, and hence the log-likelihood can be used in place of the likelihood in maximum likelihood estimation and related techniques. Finding the maximum of a function often involves taking the derivative of a function and solving for the parameter being maximized, and this is often easier when the function being maximized is a log-likelihood rather than the original likelihood function.

- For example, some likelihood functions are for the parameters that explain a collection of statistically independent observations. In such a situation, the likelihood function factors into a product of individual likelihood functions. The logarithm of this product is a sum of individual logarithms, and the derivative of a sum of terms is often easier to compute than the derivative of a product. In addition, several common distributions have likelihood functions that contain products of factors involving exponentiation. The logarithm of such a function is a sum of products, again easier to differentiate than the original function.

Likelihood Function Example



Example 1

Let p_H be the probability that a certain coin lands heads up (H) when tossed. So, the probability of getting two heads in two tosses (HH) is p_H^2. If $p_\text{H} = 0.5$, then the probability of seeing two heads is 0.25.

In symbols, we can say the above as:

$$P(\text{HH}|p_\text{H} = 0.5) = 0.25.$$

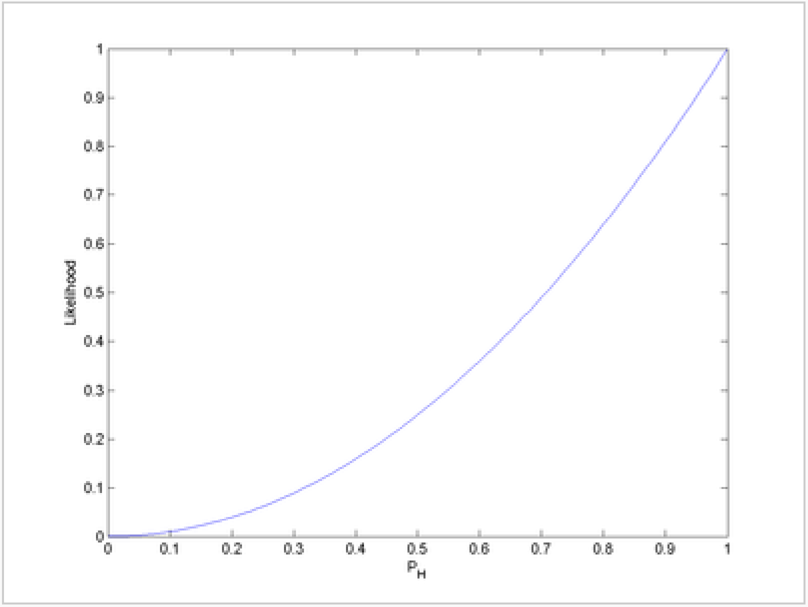
Another way of saying this is to reverse it and say that "the likelihood that $p_\text{H} = 0.5$, given the observation HH, is 0.25"; that is:

$$\mathcal{L}(p_\text{H} = 0.5|\text{HH}) = P(\text{HH}|p_\text{H} = 0.5) = 0.25.$$

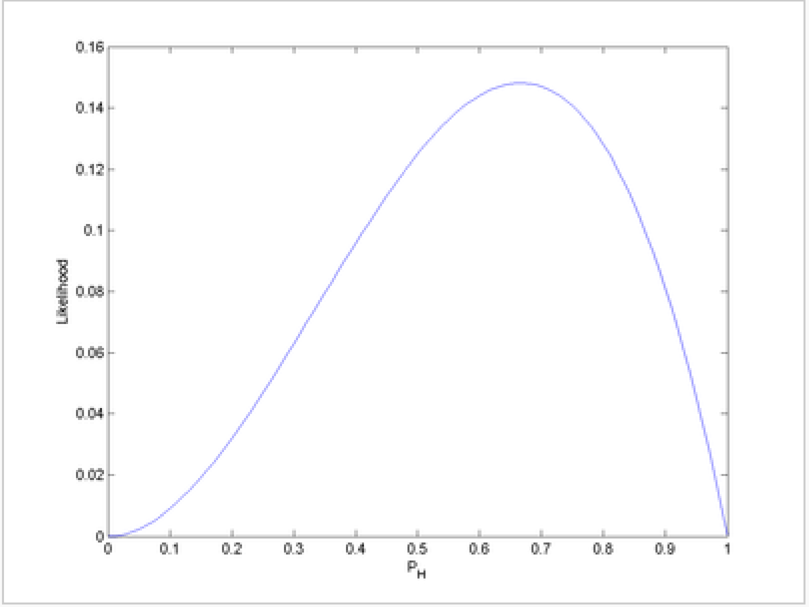
But this is not the same as saying that the *probability* that $p_\text{H} = 0.5$, given the observation HH, is 0.25.

Notice that the likelihood that $p_\text{H} = 1$, given the observation HH, is 1. But it is clearly not true that the *probability* that $p_\text{H} = 1$, given the observation HH, is 1. Two heads in a row hardly proves that the coin *always* comes up heads. In fact, two heads in a row is possible for any $p_\text{H} > 0$.

The likelihood function is not a probability density function. Notice that the integral of a likelihood function is not in general 1. In this example, the integral of the likelihood over the interval [0, 1] in p_H is 1/3, demonstrating that the likelihood function cannot be interpreted as a probability density function for p_H.



The likelihood function for estimating the probability of a coin landing heads-up without prior knowledge after observing HH



The likelihood function for estimating the probability of a coin landing heads-up without prior knowledge after observing HHT

Maximum Likelihood

- In statistics, maximum-likelihood estimation (MLE) is a method of estimating the parameters of a statistical model. When applied to a data set and given a statistical model, maximum-likelihood estimation provides estimates for the model's parameters.

- The method of maximum likelihood corresponds to many well-known estimation methods in statistics. For example, one may be interested in the heights of adult female penguins, but be unable to measure the height of every single penguin in a population due to cost or time constraints. Assuming that the heights are normally (Gaussian) distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish this by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable (given the model).

- In general, for a fixed set of data and underlying statistical model, the method of maximum likelihood selects the set of values of the model parameters that maximizes the likelihood function. Intuitively, this maximizes the "agreement" of the selected model with the observed data, and for discrete random variables it indeed maximizes the probability of the observed data under the resulting distribution. Maximum-likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems
- Maximum Likelihood for a fully observed dataset, maximum likelihood learning of the table entries correspond to counting the number of occurrences in the training data.

Entropy (Wikipedia)

- In information theory, entropy is a measure of the uncertainty in a random variable. In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the information contained in a message. Entropy is typically measured in bits, nats, or bans. Shannon entropy is the average unpredictability in a random variable, which is equivalent to its information content.
- The concept was introduced by Claude E. Shannon in his 1948 paper "A Mathematical Theory of Communication". Shannon entropy provides an absolute limit on the best possible lossless encoding or compression of any communication, assuming that[5] the communication may be represented as a sequence of independent and identically distributed random variables. Shannon's source coding theorem shows that, in the limit, the average length of the shortest possible representation to encode the messages in a given alphabet is their entropy divided by the logarithm of the number of symbols in the target alphabet.

Principle of Max. Entropy

- The **principle of maximum entropy** states that, subject to precisely stated prior data (such as a proposition that expresses testable information), the probability distribution which best represents the current state of knowledge is the one with largest entropy.