Markov Chains

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Sequence models

P(x) = probability of sequence x

$$P(x) = \underbrace{P(x_k, x_{k-1}, \ldots x_1)}_{\text{Joint probability of each base}}$$

Estimating P(x): # occurrences *inside* ÷ # occurrences total

For large k, might see few or no occurrences of x. Joint probabilities for very rare events are hard to estimate well!

Sequence models

$$P(x) = P(x_k, x_{k-1}, \dots x_1)$$

multiplication rule

$$= P(x_k \mid x_{k-1}, \dots x_1)\, P(x_{k-1}, \dots x_1)$$

multiplication rule

$$= P(x_k \mid x_{k-1}, \dots x_1)\, P(x_{k-1} \mid x_{k-2}, \dots x_1)\, P(x_{k-2}, \dots x_1)$$

(etc)

Assumption: probability of item at position *k* depends only on item at previous position: x_{k-1}

Technically: x_k is *conditionally independent* of $x_1 \dots x_{k-2}$ given x_{k-1}

Informally: "the future is independent of the past given the present"

Sequence models

Assumption: probability of item at position *k* depends only on item at previous position: x_{k-1}

$$
\begin{aligned}
P(x) &= P(x_k, x_{k-1}, \dots x_1) \\
&= P(x_k \mid x_{k-1}, \dots x_1)\, P(x_{k-1}, \dots x_1) \\
&= P(x_k \mid x_{k-1} \underbrace{, \dots x_1}_{\text{drop}})\, P(x_{k-1} \mid x_{k-2} \underbrace{, \dots x_1}_{\text{drop}})\, P(x_{k-2}, \dots x_1) \quad \text{(etc)}
\end{aligned}
$$

(etc)

$$
\approx P(x_k \mid x_{k-1})\, P(x_{k-1} \mid x_{k-2}) \dots P(x_2 \mid x_1)\, P(x_1)
$$

Markov property / assumption

Big assumption, but often reasonable and opens the door to tractable, powerful algorithms

Markov assumption

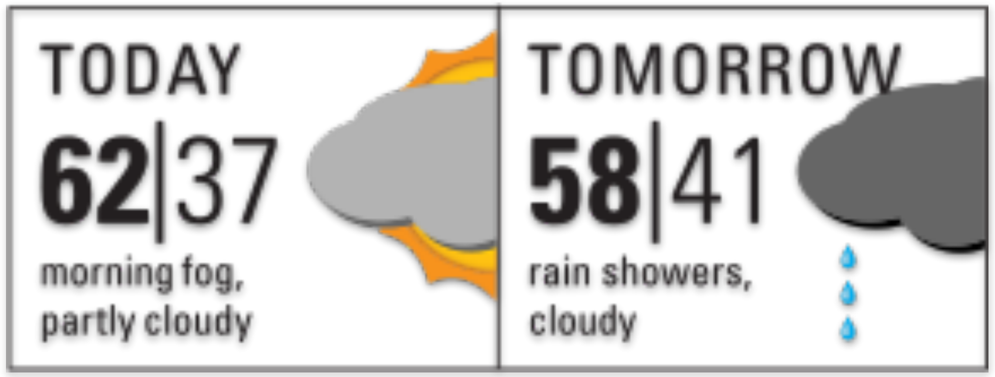
"To predict next state of the Parcheesi game, just tell me the current state. I don't care about any other previous states."

Reasonable assumption; basically true

commons.wikimedia.org/wiki/File:Parcheesi-board.jpg

"To predict today's weather, just tell me yesterday's weather. I don't care about any other previous days' weather."

It helps more to know more than just previous day's weather. Still, fairly reasonable assumption.



en.wikipedia.org/wiki/Weather_forecasting#/media/
File:Newspaper_weather_forecast_-_today_and_tomorrow.svg

Markov chain

Assigning a probability to a sequence using Markov property:

$$P(x) \underset{\text{Markov property}}{\approx} P(x_k \mid x_{k-1})\, P(x_{k-1} \mid x_{k-2}) \ldots P(x_2 \mid x_1)\, P(x_1)$$

Say x is a nucleotide k-mer

$P(x_i \mid x_{i-1})$ probability of nucleotide x_i in i^{th} position given previous nucleotide was x_{i-1}

Shorthand: $P(G \mid C)$ = probability of G given previous is C

Markov chain

Say someone gives us the sequences of several CpG islands. How do we estimate, say, $P(G \mid C)$?

$P(G \mid C) =$ # times CG occurs / # times CX occurs

where X is any base

Markov chain

Given CpG island sequences from human chromosome 1, count dinucleotide occurrences and estimate all 16 possible $P(x_i \mid x_{i-1})$:

P(A | A) = # times AA occurs / # times AX occurs

P(C | A) = # times AC occurs / # times AX occurs

P(G | A) = # times AG occurs / # times AX occurs

P(T | A) = # times AT occurs / # times AX occurs

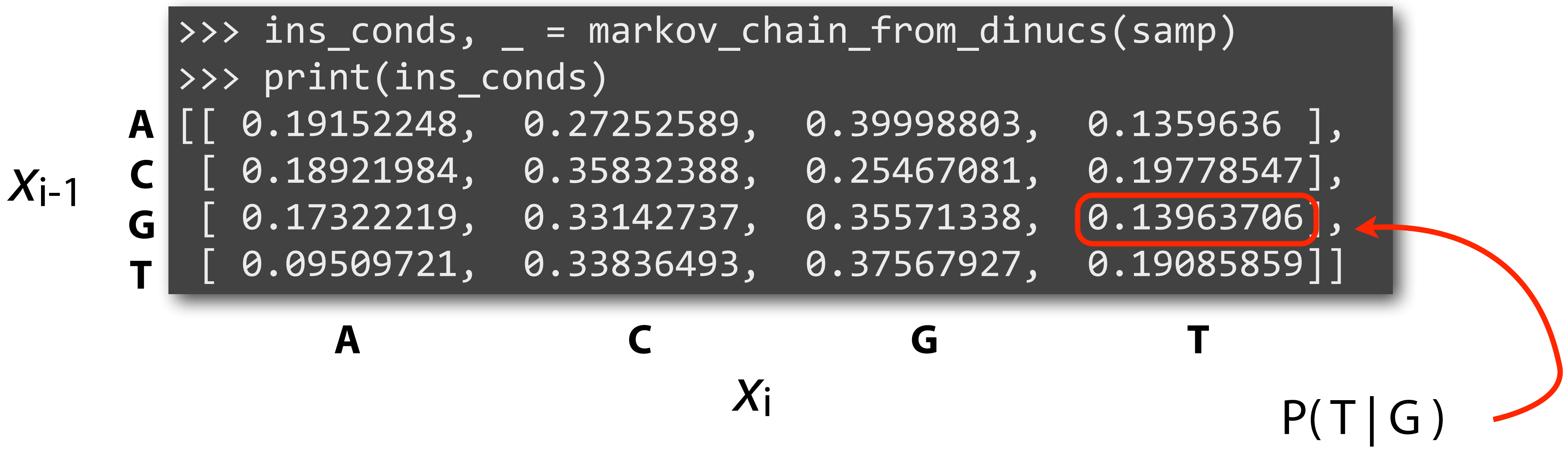
P(A | C) = # times CA occurs / # times CX occurs

(etc)

where X is any base

Markov chain

Given example CpG island substrings we can estimate all P(*base* | *previous base*):



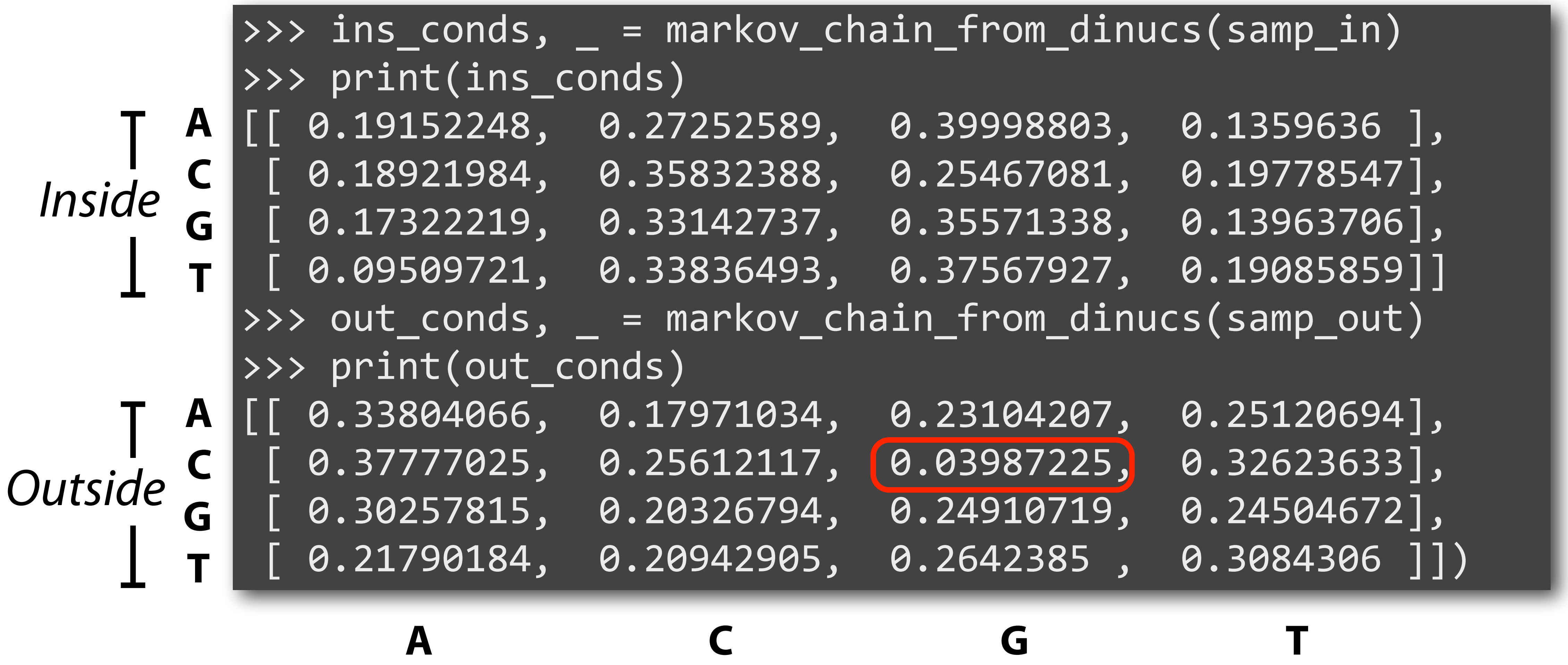
```
>>> ins_conds, _ = markov_chain_from_dinucs(samp)
>>> print(ins_conds)
[[ 0.19152248,  0.27252589,  0.39998803,  0.1359636 ],
 [ 0.18921984,  0.35832388,  0.25467081,  0.19778547],
 [ 0.17322219,  0.33142737,  0.35571338,  0.13963706],
 [ 0.09509721,  0.33836493,  0.37567927,  0.19085859]]
```

Rows sum to 1

http://bit.ly/CG_MarkovChain

Markov chain

We can do the same for dinucleotides *outside* of CpG islands

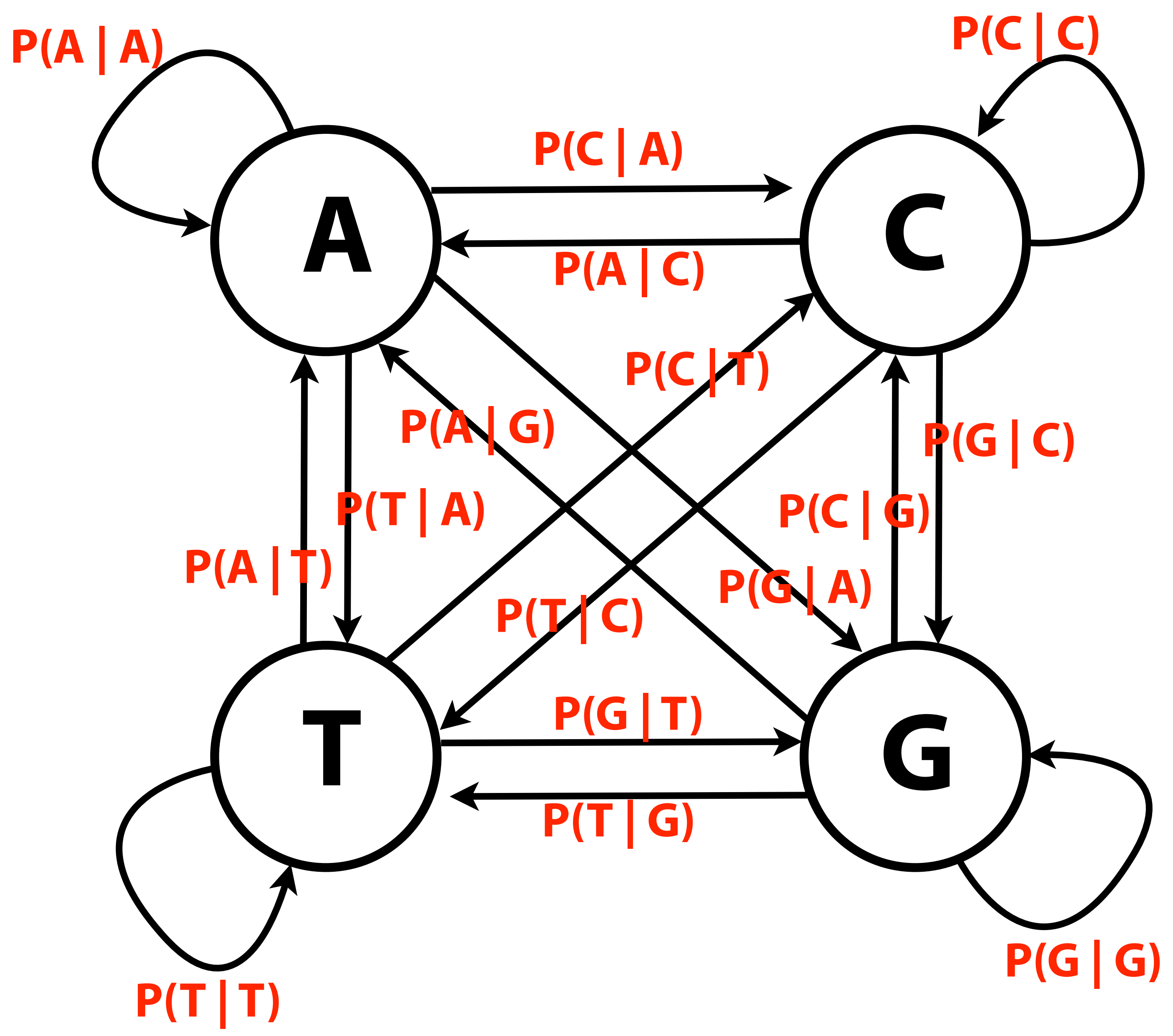


```
>>> ins_conds, _ = markov_chain_from_dinucs(samp_in)
>>> print(ins_conds)
[[ 0.19152248,  0.27252589,  0.39998803,  0.1359636 ],
 [ 0.18921984,  0.35832388,  0.25467081,  0.19778547],
 [ 0.17322219,  0.33142737,  0.35571338,  0.13963706],
 [ 0.09509721,  0.33836493,  0.37567927,  0.19085859]]
>>> out_conds, _ = markov_chain_from_dinucs(samp_out)
>>> print(out_conds)
[[ 0.33804066,  0.17971034,  0.23104207,  0.25120694],
 [ 0.37777025,  0.25612117,  0.03987225,  0.32623633],
 [ 0.30257815,  0.20326794,  0.24910719,  0.24504672],
 [ 0.21790184,  0.20942905,  0.2642385 ,  0.3084306 ]])
```

Notice anything interesting about the outside conditional probabilities?

P(G | C) is low: outside CpG islands, G is rarely preceded by C

Markov chain



Markov chain is a probabilistic automaton

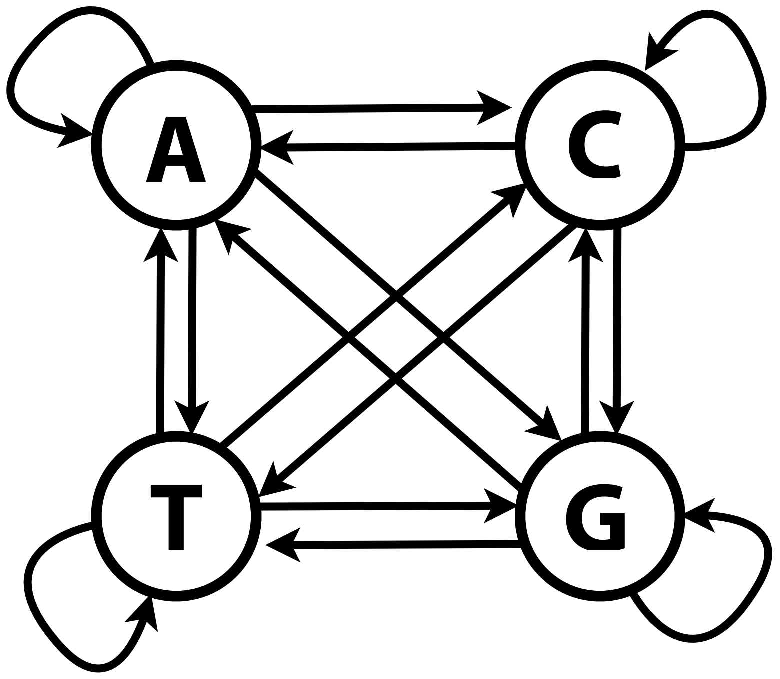
Edge has *transition probability*: probability that destination comes next after source

Markov chain

Recall how we assign a probability to a single string

$$P(x) \underset{\text{Markov property}}{\approx} P(x_k \mid x_{k-1})\, P(x_{k-1} \mid x_{k-2}) \ldots P(x_2 \mid x_1)\, P(x_1)$$

$P(x)$ equals product of Markov chain edge weights on our string-driven walk through the chain (…times $P(x_1)$)



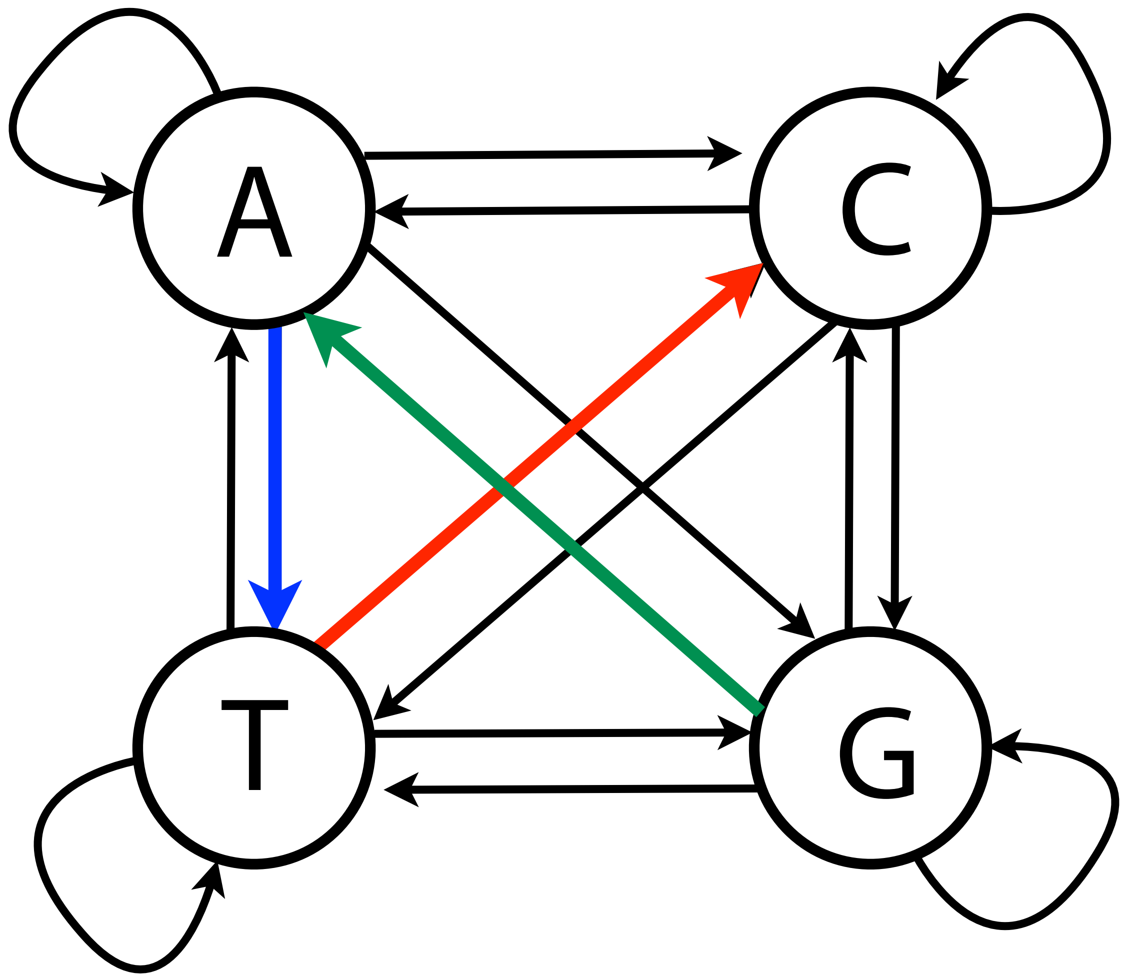
Markov chain

```
>>> ins_conds, _ = markov_chain_from_dinucs(samp_in)
>>> print(ins_conds)
[[ 0.19152248,  0.27252589,  0.39998803,  0.1359636 ],
 [ 0.18921984,  0.35832388,  0.25467081,  0.19778547],
 [ 0.17322219,  0.33142737,  0.35571338,  0.13963706],
 [ 0.09509721,  0.33836493,  0.37567927,  0.19085859]]
```

x_{i-1}: A, C, G, T

A C G T

x_i



$x = \text{GATC}$

$P(x) = P(x_4 \mid x_3)\, P(x_3 \mid x_2)\, P(x_2 \mid x_1)\, P(x_1)$

$P(x) = P(C \mid T)\; P(T \mid A)\; P(A \mid G)\; P(G)$

$= 0.33836493 \; *$
$\;\; 0.1359636 \; *$
$\;\; 0.17322219 \; *$
$\;\; 0.25$

$= 0.001992$

Markov chain

To avoid underflow, switch to log domain

$$\log P(x) \approx \log [\, P(x_k \mid x_{k-1})\, P(x_{k-1} \mid x_{k-2}) \ldots P(x_2 \mid x_1)\, P(x_1)\,]$$

$$= \log P(x_k \mid x_{k-1}) + \log P(x_{k-1} \mid x_{k-2}) + \ldots$$

product becomes **sum**

$$= \sum_{i=2}^{k} \log P(x_i \mid x_{i-1}) + \log P(x_1)$$

Assume base-2 logs

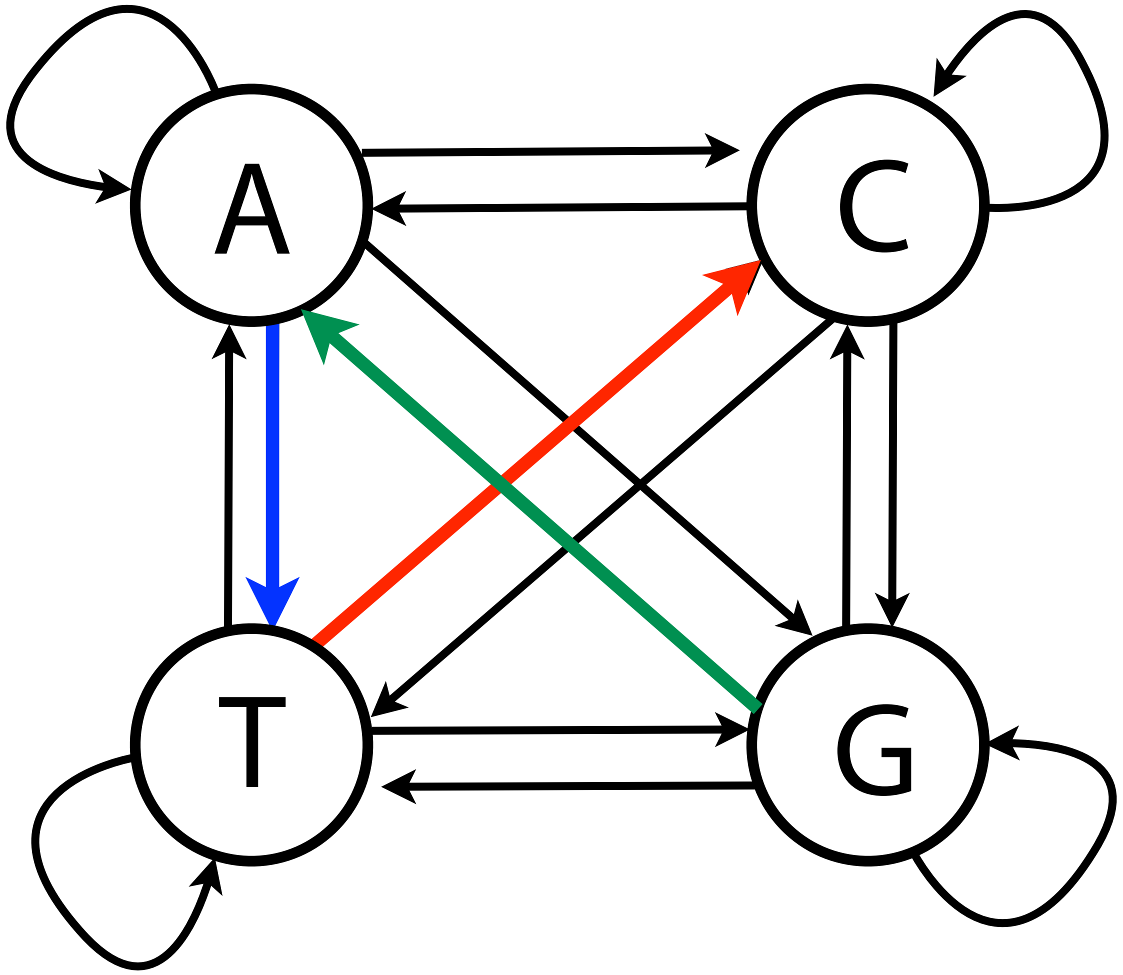
Markov chain

```
>>> ins_conds, _ = markov_chain_from_dinucs(samp_in)
>>> print(numpy.log2(ins_conds))
[[-2.44009488, -1.8820643 , -1.30195688, -2.84832282],
 [-2.38974049, -1.469396  , -2.00590131, -2.32864974],
 [-2.51948223, -1.60979755, -1.48694353, -2.82436637],
 [-3.41910668, -1.52509737, -1.43889385, -2.39435058]])
```

x_{i-1}: A, C, G, T

A C G T

x_i



$x = \text{GATC}$

$$\log P(x) = \sum_{i=2}^{4} \log P(x_i \mid x_{i-1}) + \log P(x_1)$$

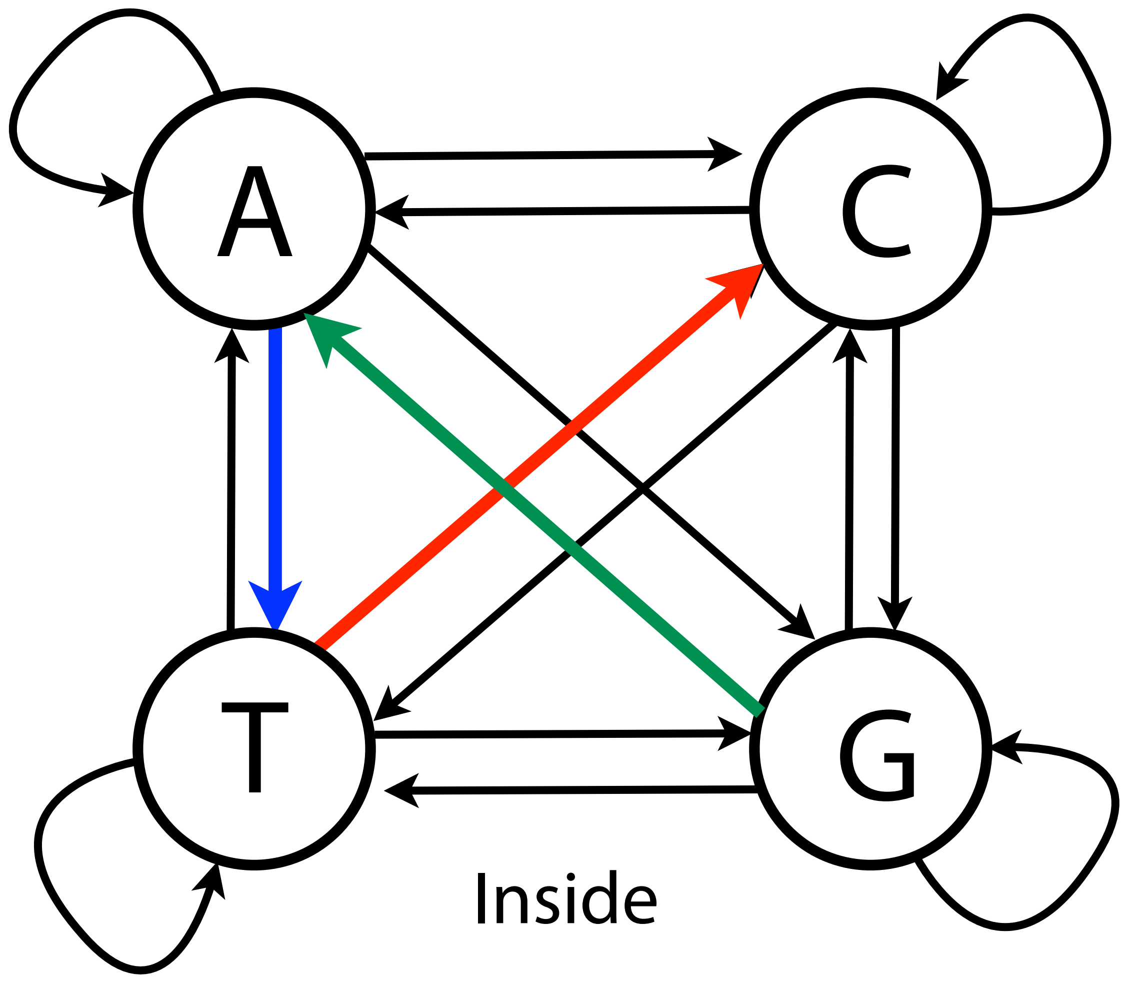
$$= -1.52509737 + -2.84832282 + -2.51948223 + -2.0$$

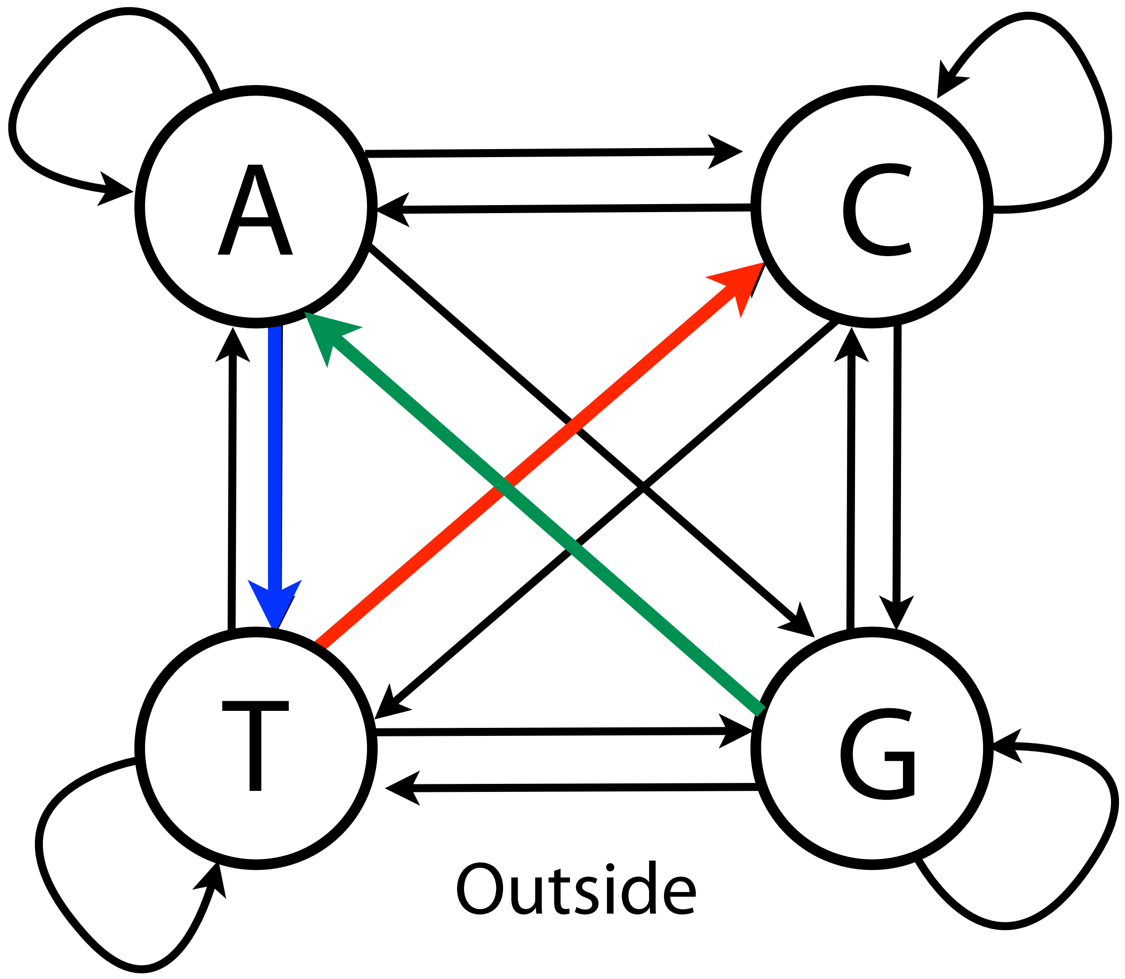
$$= -8.89290$$

Markov chain

P(x) given the inside-CpG model is helpful, but we really want to know which model is better, inside CpG or outside CpG?

Use *ratio:* $\dfrac{\text{P(}\,x\,\text{) from inside model}}{\text{P(}\,x\,\text{) from outside model}}$





Markov chain

Take log, get a *log ratio*: $S(x) = \log \dfrac{P(x) \text{ inside CpG}}{P(x) \text{ outside CpG}}$

If inside more probable than outside, fraction is > 1, log ratio is > 0.
Otherwise, fraction is ≤ 1 and log ratio is ≤ 0.

$$S(x) = \log \frac{P(x) \text{ inside CpG}}{P(x) \text{ outside CpG}}$$

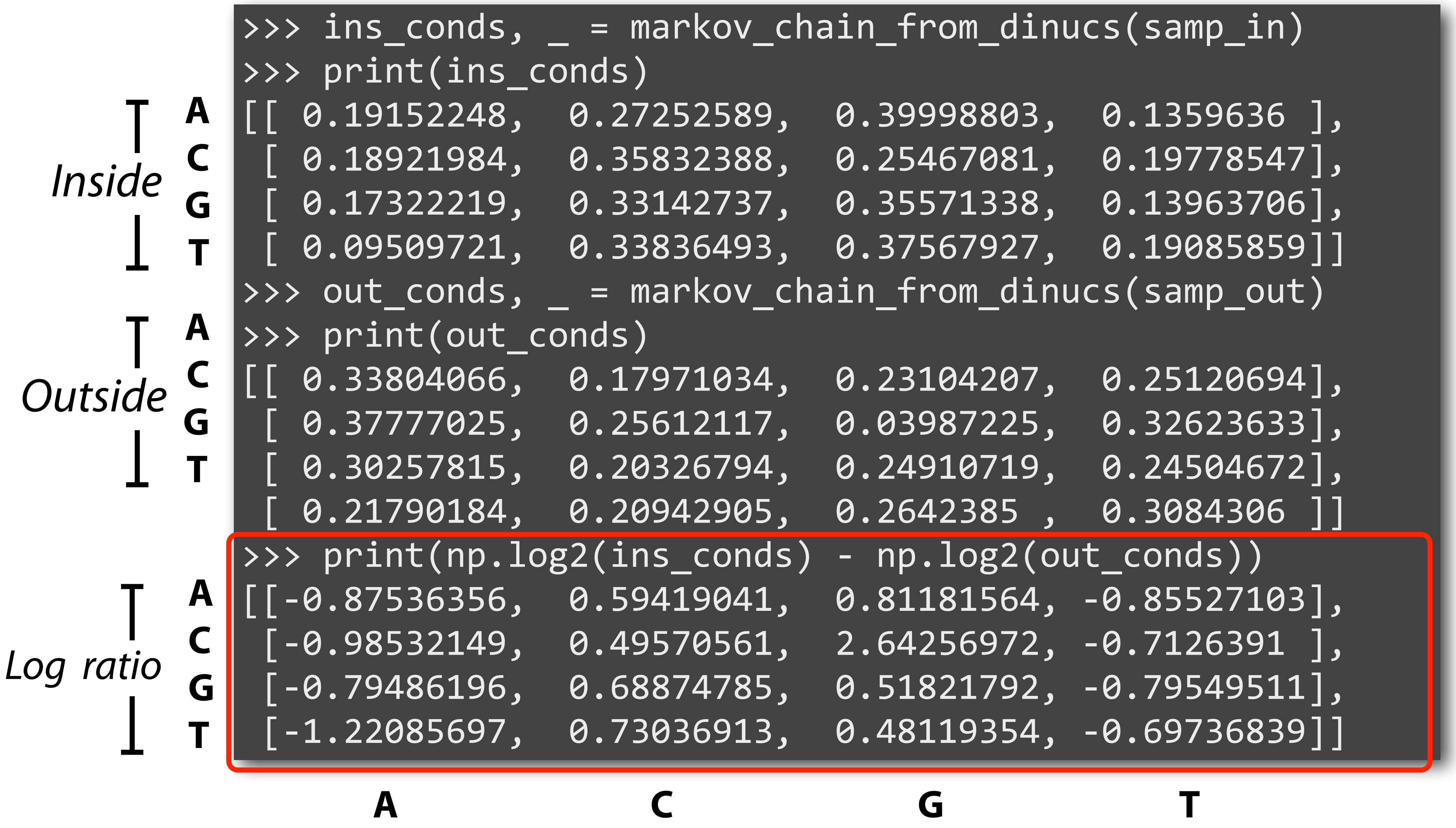
$$= \log [\, P(x) \text{ inside CpG} \,] - \log [\, P(x) \text{ outside CpG} \,]$$

(Marginal probabilities ignored here)

$$= \sum_{i=2}^{k} \left(\log \left[P(x_i \mid x_{i-1}) \text{ inside CpG} \right] \right) - \sum_{i=2}^{k} \log \left(\left[P(x_i \mid x_{i-1}) \text{ outside CpG} \right] \right)$$

$$= \sum_{i=2}^{k} \underbrace{\left(\log \left[P(x_i \mid x_{i-1}) \text{ inside CpG} \right] - \log \left[P(x_i \mid x_{i-1}) \text{ outside CpG} \right] \right)}_{\text{New table: elementwise log ratios between inside/outside}}$$

Markov chain



```
>>> ins_conds, _ = markov_chain_from_dinucs(samp_in)
>>> print(ins_conds)
[[ 0.19152248,  0.27252589,  0.39998803,  0.1359636 ],
 [ 0.18921984,  0.35832388,  0.25467081,  0.19778547],
 [ 0.17322219,  0.33142737,  0.35571338,  0.13963706],
 [ 0.09509721,  0.33836493,  0.37567927,  0.19085859]]
>>> out_conds, _ = markov_chain_from_dinucs(samp_out)
>>> print(out_conds)
[[ 0.33804066,  0.17971034,  0.23104207,  0.25120694],
 [ 0.37777025,  0.25612117,  0.03987225,  0.32623633],
 [ 0.30257815,  0.20326794,  0.24910719,  0.24504672],
 [ 0.21790184,  0.20942905,  0.2642385 ,  0.3084306 ]]
>>> print(np.log2(ins_conds) - np.log2(out_conds))
[[-0.87536356,  0.59419041,  0.81181564, -0.85527103],
 [-0.98532149,  0.49570561,  2.64256972, -0.7126391 ],
 [-0.79486196,  0.68874785,  0.51821792, -0.79549511],
 [-1.22085697,  0.73036913,  0.48119354, -0.69736839]]
```

Markov chain

Now, given a string *x*, we can easiy assign it a log ratio “score” *S*(*x*):

$$S(x) = \log \frac{P(\,x\,) \text{ inside CpG}}{P(\,x\,) \text{ outside CpG}}$$

$$\approx \sum_{i=2}^{k} \Big(\log \big[\, P(\,x_i \mid x_{i-1}\,) \text{ inside CpG} \,\big] - \log \big[\, P(\,x_i \mid x_{i-1}\,) \text{ outside CpG} \,\big] \Big)$$

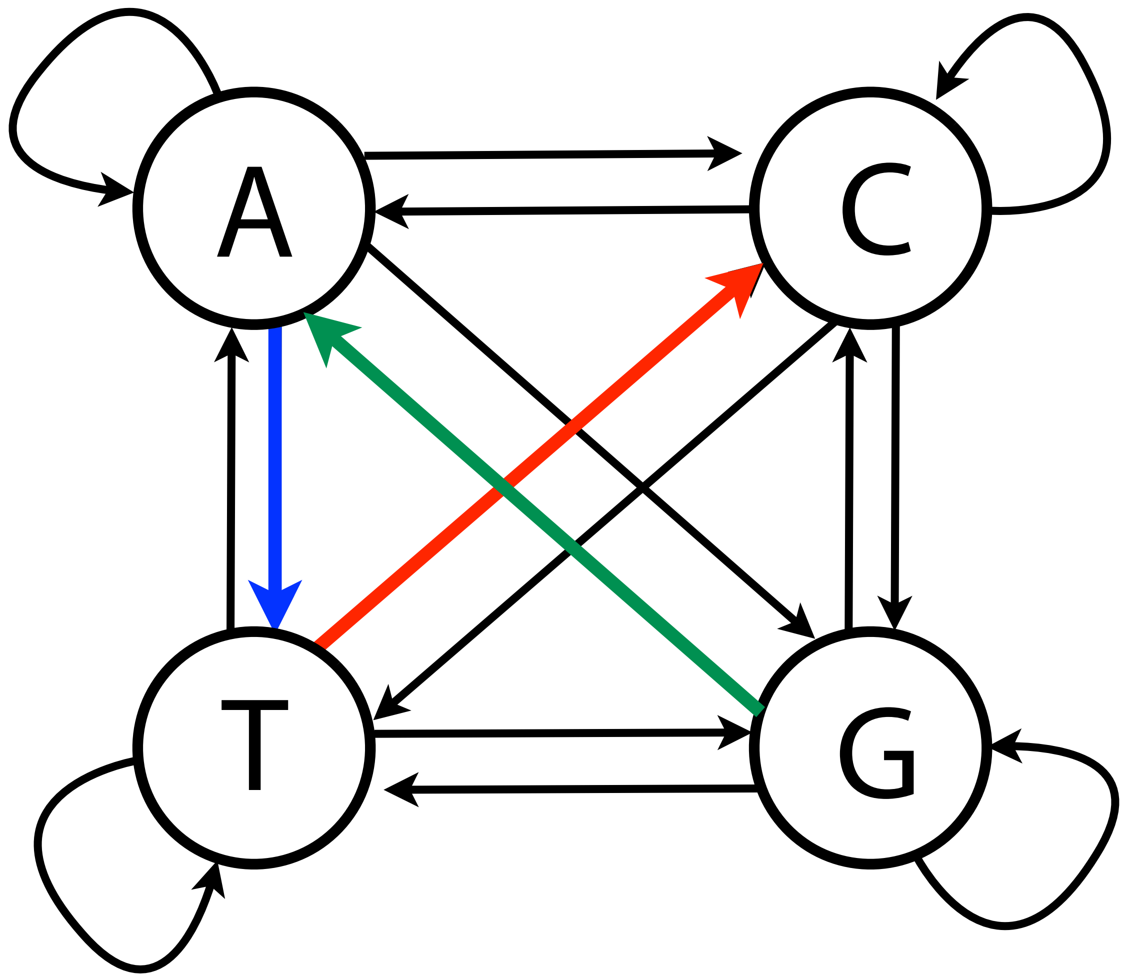
Markov chain

```
>>> ins_conds, _ = markov_chain_from_dinucs(samp_in)
>>> out_conds, _ = markov_chain_from_dinucs(samp_out)
>>> print(np.log2(ins_conds) - np.log2(out_conds))
[[-0.87536356,  0.59419041,  0.81181564, -0.85527103],
 [-0.98532149,  0.49570561,  2.64256972, -0.7126391 ],
 [-0.79486196,  0.68874785,  0.51821792, -0.79549511],
 [-1.22085697,  0.73036913,  0.48119354, -0.69736839]]
```

x_{i-1}: A, C, G, T

A C G T

x_i



$x = \text{GATC}$

$$\begin{aligned} S(x) = &\ 0.73036913\ + \\ &-0.85527103\ + \\ &-0.79486196 \\ = &\ -0.919763 \end{aligned}$$

Negative, so probability with *outside* model is greater

Markov chain

$$S(x) = \log \frac{P(x) \text{ inside CpG}}{P(x) \text{ outside CpG}}$$

S(CGCGCGCGCGCGCGCGCGCGCGCG) = 42.618

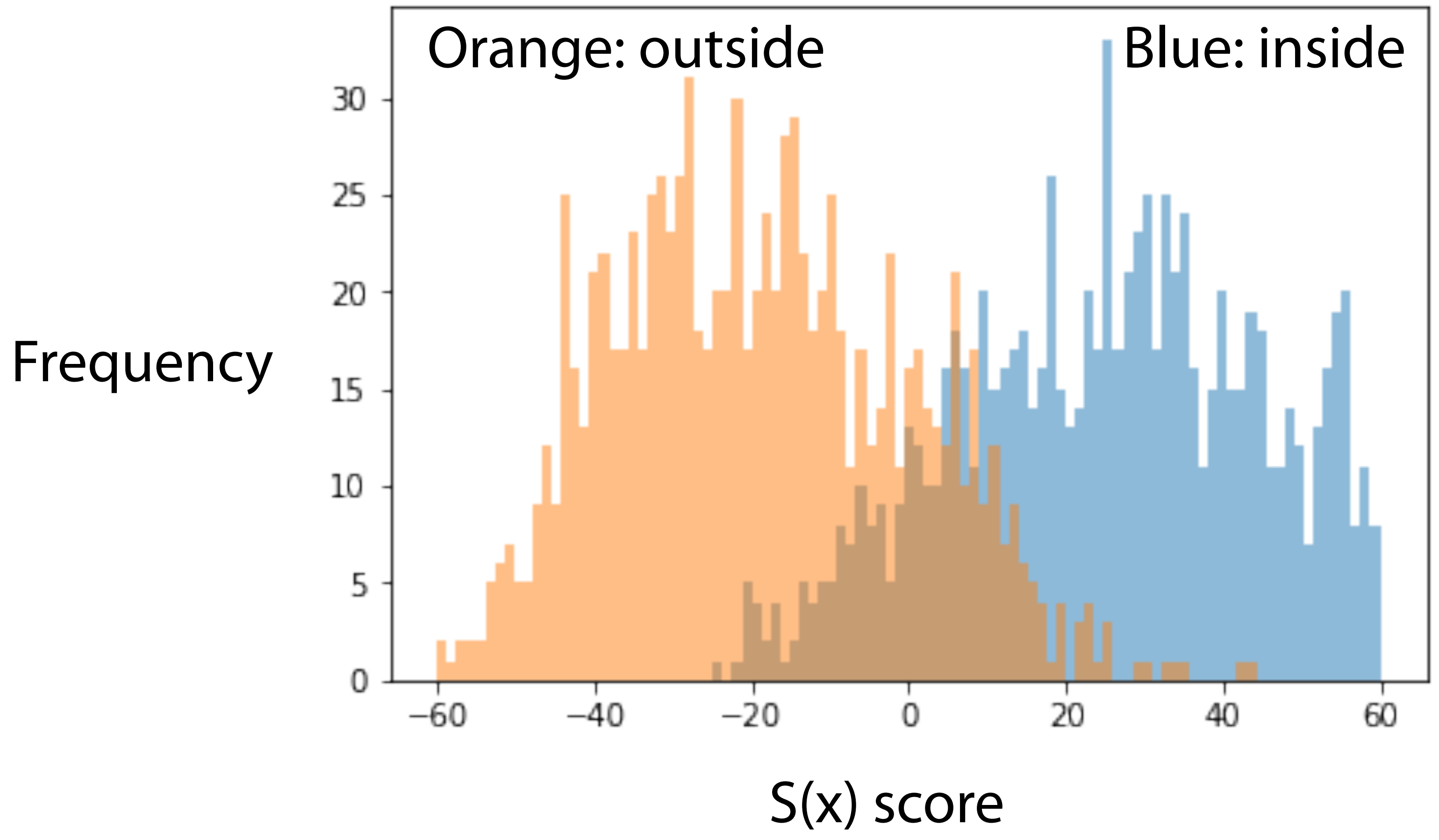
S(ATTCTACTATCATCTATCTATCTTCT) = -10.839

http://bit.ly/CG_MarkovChain

Markov chain

Drew 1,000 100-mers from inside CpG islands on chromosome 18, and another 1,000 from outside, and calculated S(x) for all

Trained markov chain on dinucleotides from chromosome 22



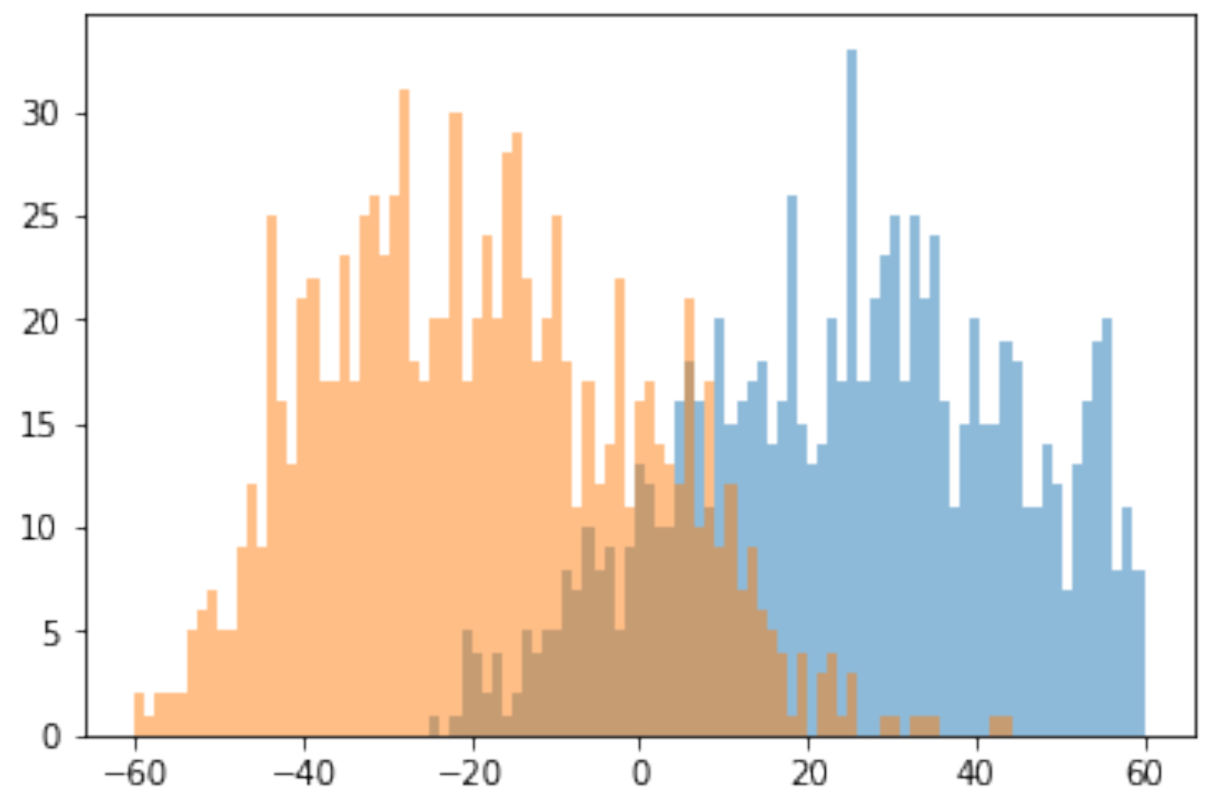
http://bit.ly/CG_MarkovChain

Markov chain

Markov property made our problem very tractable

- $P(x_i \mid x_{i-1})$s estimated in single, simple pass through training data
- Transition probability tables have $|\Sigma|^2$ cells; fine for DNA & protein
- Calculating $S(x)$ is $O(|x|)$; just lookups and additions

Discriminates well between inside & outside:



30
25
20
15
10
5
0
−60
−40
−20
0
20
40
60