Lecture 8.

Note Title 4/15/2011

Dynamic Programming

Consider a distribution defined over many variables $\pi(\underline{x})$ with $\underline{x} = (x_0, x_1, \ldots x_N)$

N large (e.g. N=100)

Suppose each x_i can take k possible values s_a

$$x_i \in S = \{s_1, \ldots, s_k\}$$

Total number of states $\underline{x}$ is k^N – very big – this makes it hard to do computations.

E.G. Suppose $\pi(\underline{x}) = \frac{1}{Z} e^{-E(\underline{x})}$ Gibbs distribution

Then computing $Z = \sum_{\underline{x}} e^{-E(\underline{x})}$ may require k^N computations – evaluate all possible states – too many!

computing $\hat{\underline{x}} = \text{ARG MAX}_{\underline{x}} \pi(\underline{x})$ – $\pi(\hat{\underline{x}}) \geq \pi(\underline{x})$ for all $\underline{x}$

may require k^N computations – too many!

(unless N & k small)

Also, how to sample from $\pi(\underline{x})$ to get i.i.d samples

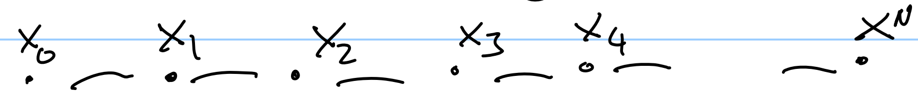
$$\underline{x}^1 = (x_0^1, \ldots x_N^1)$$
$$\underline{x}^2 = (x_0^2, \ldots x_N^2)$$
$$\underline{x}^3, \underline{x}^4, \ldots$$

?

How to estimate quantities like $\sum_{\underline{x}} \pi(\underline{x}) h(\underline{x})$ for some $h(\underline{x})$?

(2) Today we discuss a special class of probability distributions for which we can solve these problems efficiently by Dynamic Programming (DP).

<u>Background</u> probability distributions $\pi(\underline{x})$ can be represented graphically, by assigning a <u>node</u> to each variable x_i, and an <u>edge</u> linking nodes which are directly related (Markov condition).



$$\pi(\underline{x}) = \frac{1}{Z} e^{-\sum_{i=1}^{N} \varphi_i(x_{i-1}, x_i)}$$

$\varphi(\cdot,\cdot)$ is a <u>potential</u>.

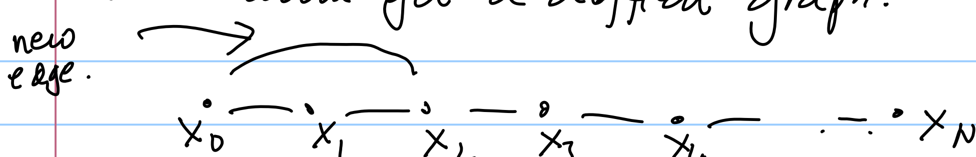
The <u>potentials</u> $\varphi_i(x_{i-1}, x_i)$ 'relate' variables x_{i-1} to x_i — so have 'edges' in the graph.

<u>If</u> we added a new potential $\varphi(x_0, x_2)$

— so $\pi(\underline{x}) = \frac{1}{\tilde{Z}} e^{-\varphi(x_0,x_2) - \sum_{i=1}^{N} \varphi_i(x_{i-1},x_i)}$ —

(note $\tilde{Z} \neq Z$, different normalization constant.)

then we would get a different graph.



<u>Today</u>'s lecture will stick to a distribution

$$\pi(\underline{x}) = \frac{1}{Z} e^{-\sum_{i=1}^{N} \varphi_i(x_{i-1}, x_i)}$$

(graph has no closed loops)

③ <u>Probabilities</u> like $\pi(\underline{x}) = \frac{1}{Z} e^{-\sum_{i=1}^{N} \varphi_i (x_{i-1}, x_i)}$

Note Title 4/2/2006

can be used to model events over time – eg.

x_i is state of system at time i

x_{i+1} is state of system at time $i+1$

state of system at time i is specified (probabilistically) by state of system at time $i-1$ (more on this next lecture.)

Or, more generally, any 1-D structure like the sequences of DNA

E.G GATTACA...

A - adenine
C - cytosine
G - guanine
T - thymine.

And many others....

<u>Dynamic Programming</u> can be used to find the global maximum of $\pi(\underline{x})$, $\hat{\underline{x}}$, and $\pi(\hat{\underline{x}})$ in $O(NK^2)$ operations. (not K^N)

<u>DP</u> can also compute $Z = \sum_{\underline{x}} e^{-\sum_{i=1}^{N} \varphi_i (x_{i-1}, x_i)}$ in $O(NK^2)$ operations

DP can also find the marginal distribution $\pi_i(x_i)$ and draw exact random samples from $\pi(\underline{x})$ efficiently.

(4)

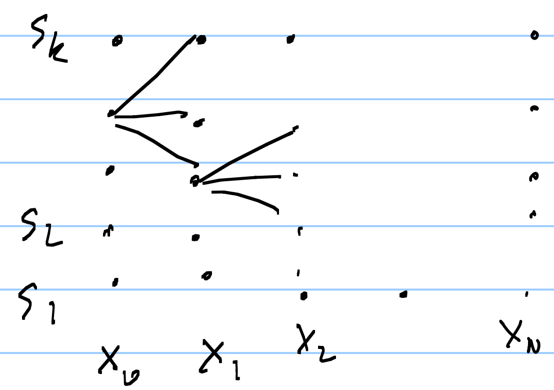
Maximizing $\pi(\underline{x})$ is equivalent to minimizing $E(\underline{x}) = \varphi_1(x_0, x_1) + \ldots + \varphi_N(x_{N-1}, x_N)$

Forward Pass of DP

DP acts recursively: (i.e. $x_0 = s_1$ or s_2 or s_3 or ... or s_k)

- Define $m_1(x_1) = \min_{s_i \in S} \varphi_1(s_i, x_1)$ for $x_1 = s_1, \ldots s_k$
- Recursively compute $m_t(x_t) = \min_{s_i \in S} \{m_{t-1}(s_i) + \varphi_t(s_i, x_t)\}$, for $x_t = s_1 \ldots s_k$.

Claim: Optimal value $E(\hat{x})$ is obtained by $\min_{s \in \{s_1 \ldots s_k\}} m_N(x_N)$



To compute $m_1(x_1)$ for $x_1 = s_1 \ldots s_k$ needs $O(k^2)$ operations.

Computing all $m_t(x_t)$... requires $O(kN^2)$

Justify Claim: minimum of $m_t(x_t)$ is the minimum of $\varphi_1(x_0, x_1)$

proof by induction: $\min_{x_t \in S} m_t(x_t) = \min_{x_0, \ldots x_1 \in S} \{\varphi_1(x_0, x_1) + \ldots + \varphi_t(x_{t-1}, x_t)\}$.

(5). To find the optimal path $\hat{\underline{x}}$ we trace backwards

<u>Backward Pass of DP</u>

- Let $\hat{x}_N$ be the minimizer of $m_N(x_N)$

$$\hat{x}_N = \arg\min_{s_i \in S} m_N(s_i)$$

(Break ties arbitrarily – ties mean that there are several $\underline{x}$'s s.t. $E(\underline{x}) = E(\hat{\underline{x}})$)

- For $t = N-1, N-2, \ldots, 0$

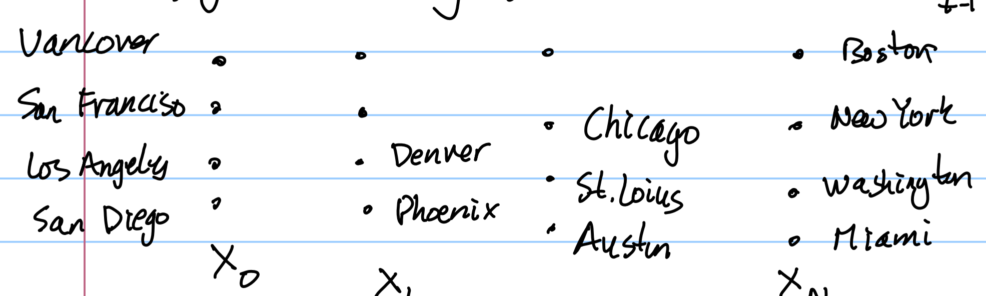
Let $\hat{x}_t = \arg\min_{s_i \in S} \{ m_t(s_i) + \varphi_{t+1}(s_i, \hat{x}_{t+1}) \}$,

(Break ties arbitrarily)

$\hat{\underline{x}} = (\hat{x}_0, \ldots, \hat{x}_N)$ obtained in this way is the minimizer. If ties, then several minimizers $\underline{x}$ (doesn't matter which you pick)

<u>Example:</u> Task – find shortest path from West Coast of US to East Coast.

On day t, you start in city x_{t-1} and drive distance $\varphi_{t-1}(x_{t-1}, x_t)$ to city x_t



(eg from LA to Phoenix) $m_t(x_t)$ is shortest distance to x_t from West Coast.

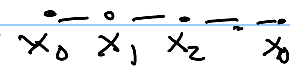
(6) Intuition: DP solves the minimization problem efficiently by breaking it down into seperate parts.

E.G. If you want to find the shortest path between the West Coast and the East Coast which goes through Chicago, then you can do this by solving two independent problems:

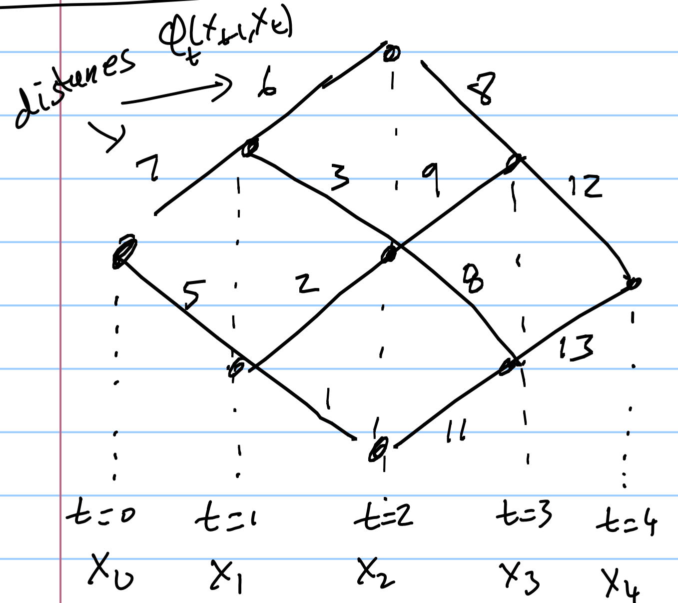
(i) Find shortest path from West Coast to Chicago

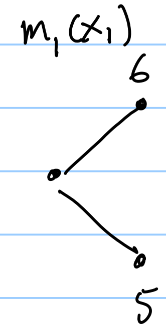
(ii) Find shortest path from Chicago to East Coast

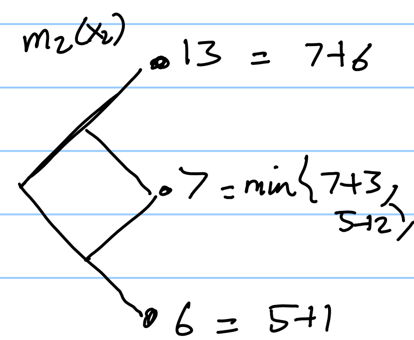
Note: DP only works if the probability distribution $\pi(\underline{x})$ is defined on a graph with no closed loops. Like.

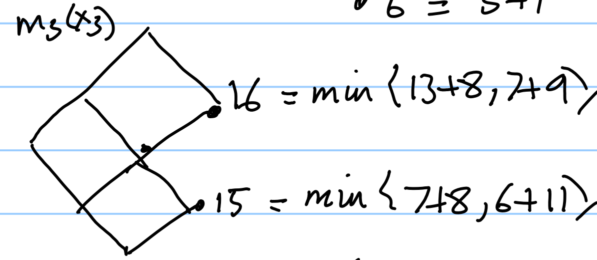


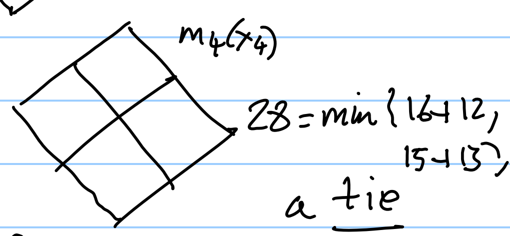
Simplest Example of DP: X_0 has 1 state only, X_1 2 states, X_3 3 states, X_4 2 states, X_5 1 state.



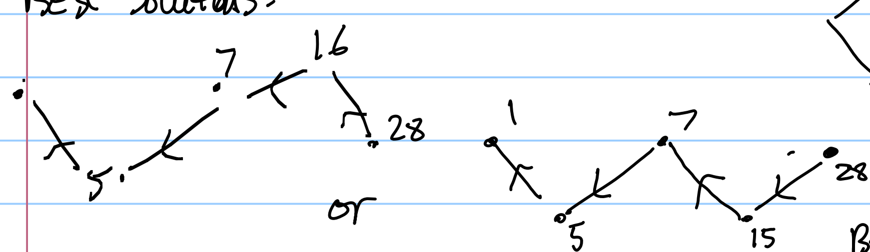








Two Best Solutions:



Both minimize $\sum_{t=1}^{t=4} \varphi_t(x_{t-1},x_t)$.