Approximate Inference: Randomized Methods

October 15, 2015

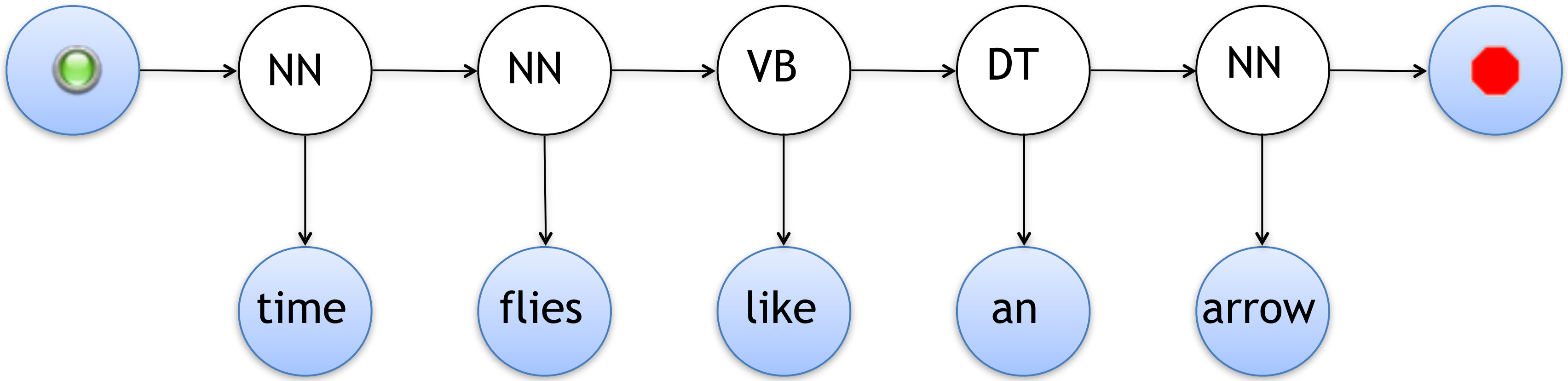
Topics

- Hard Inference
  - Local search & hill climbing
  - Stochastic hill climbing / Simulated Annealing
- Soft Inference
  - Monte-Carlo approximations
  - Markov-Chain Monte Carlo methods
    - Gibbs sampling
    - Metropolis Hastings sampling
  - Importance Sampling

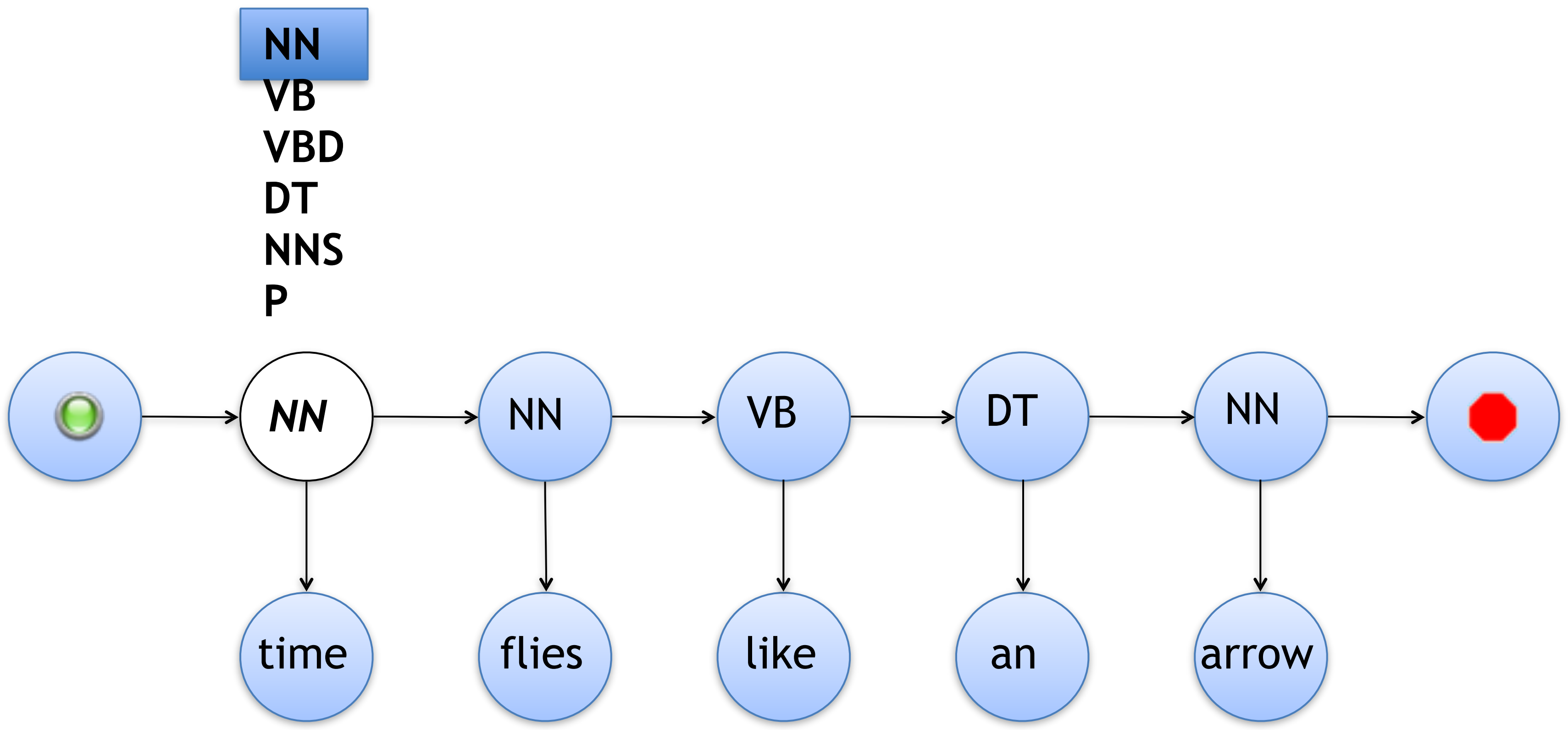
Local Search

- Start with a candidate solution
- Until (time > limit) or no changes possible:
  - Apply a local change to generate a new candidate solutions
  - Pick the one with the highest score (“steepest ascent”)
- A **neighborhood function** maps a search state (+ optionally, algorithm state) to a set of neighboring states
  - Assumption: computing the score (*cf*. unnormalized probability) of the new state is inexpensive

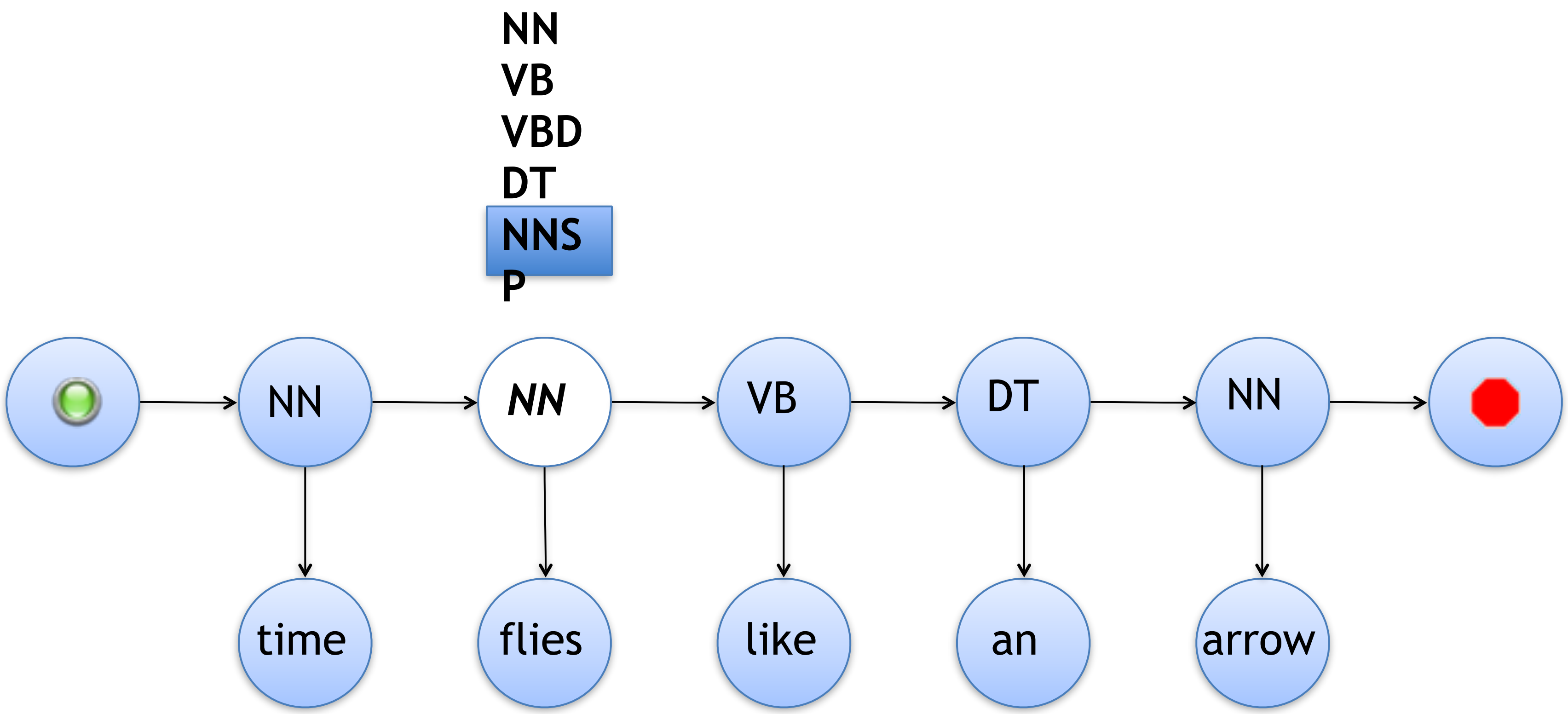
Hill Climbing



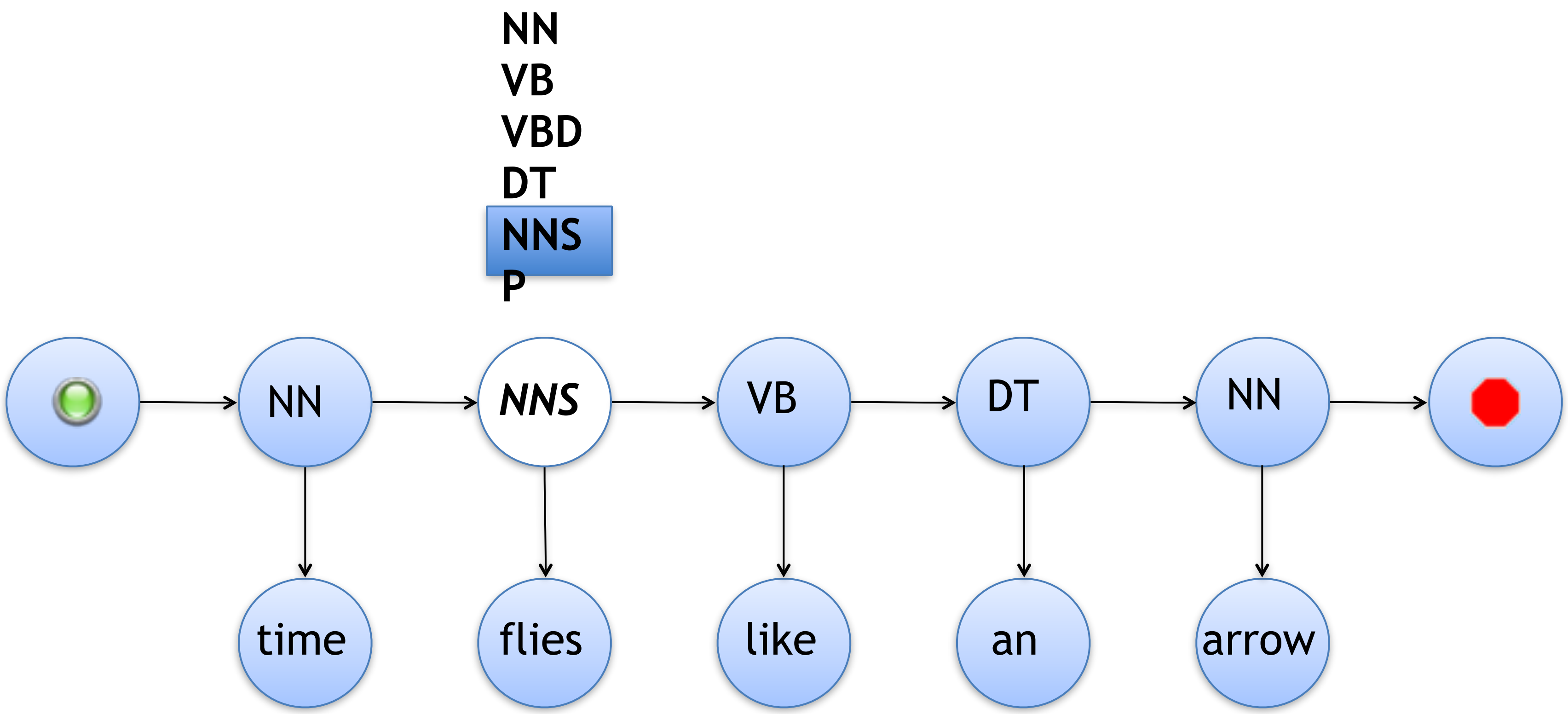
Hill Climbing



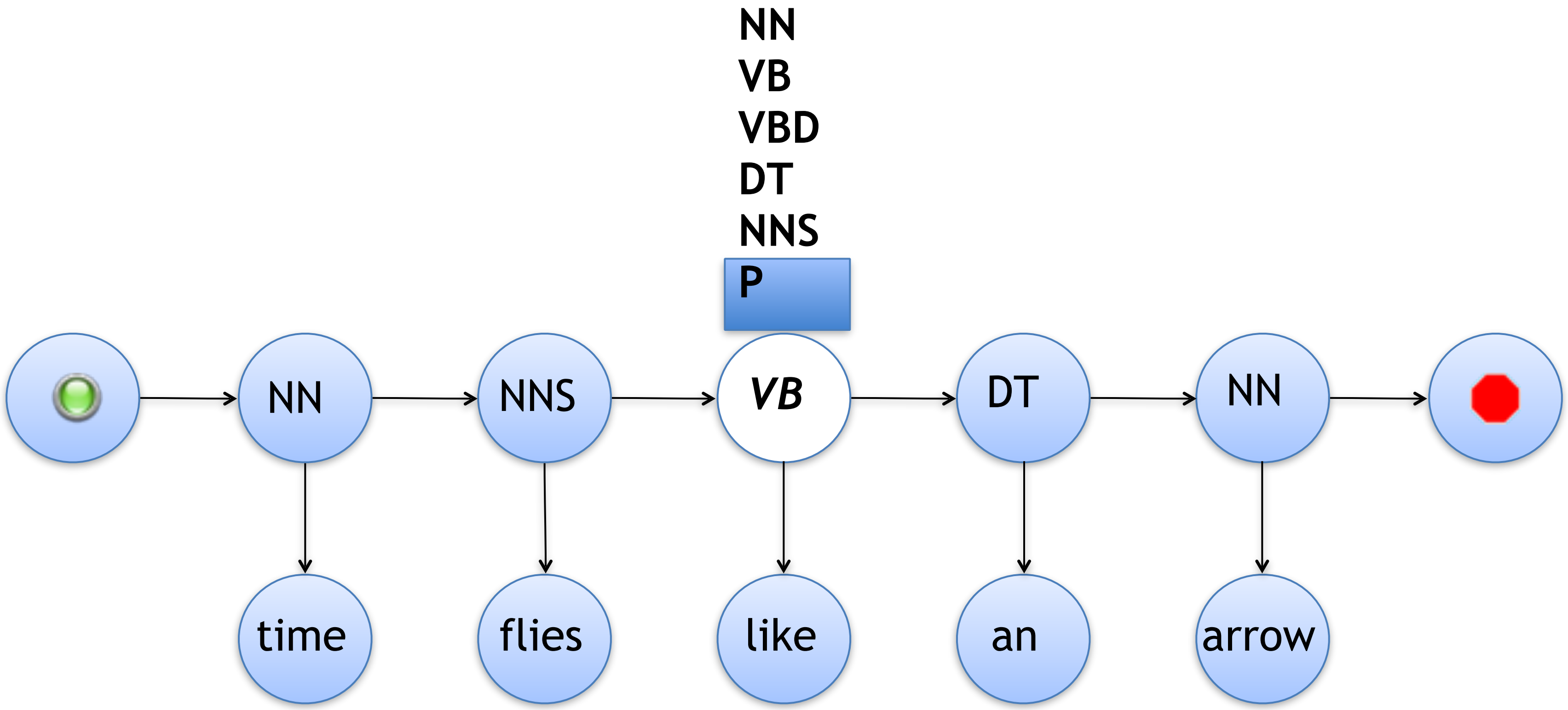
Hill Climbing



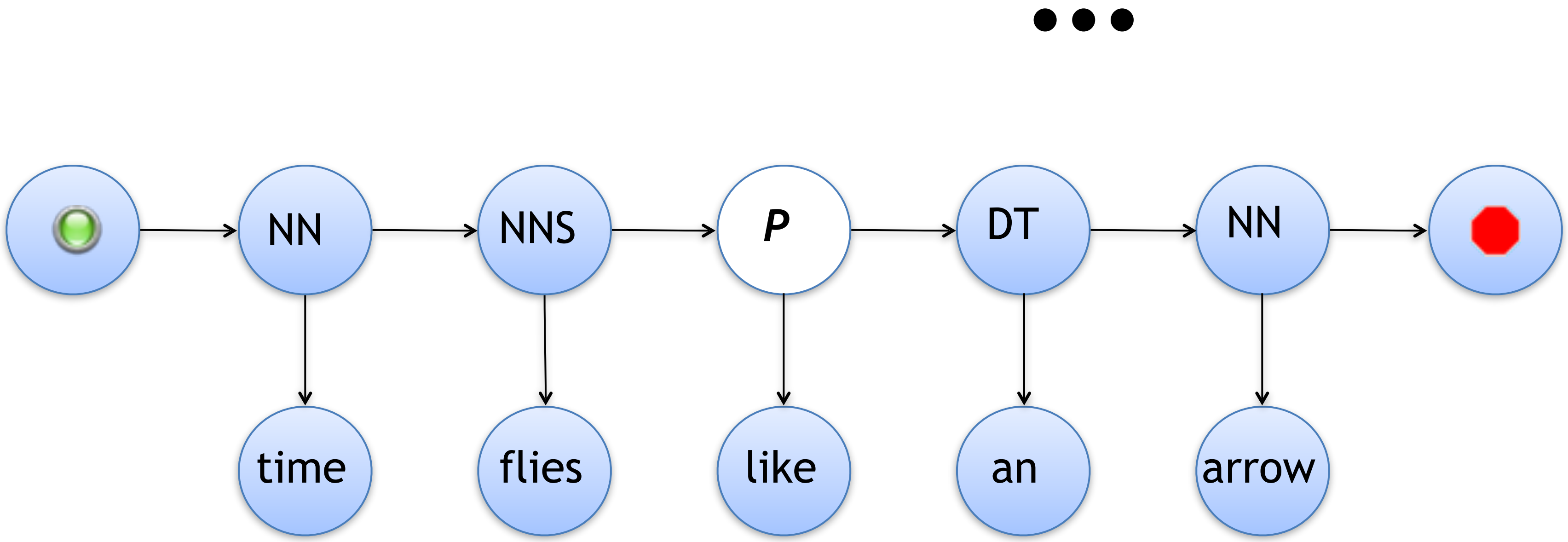
Hill Climbing



Hill Climbing



Hill Climbing



Hill Climbing: Sequence Labeling

- **Start with greedy assignment – O(*n*|*L*|)**
- While stop criterion not met
  - For each label position (*n* of them)
    - Consider changing to any label, including no change

- When should we stop?

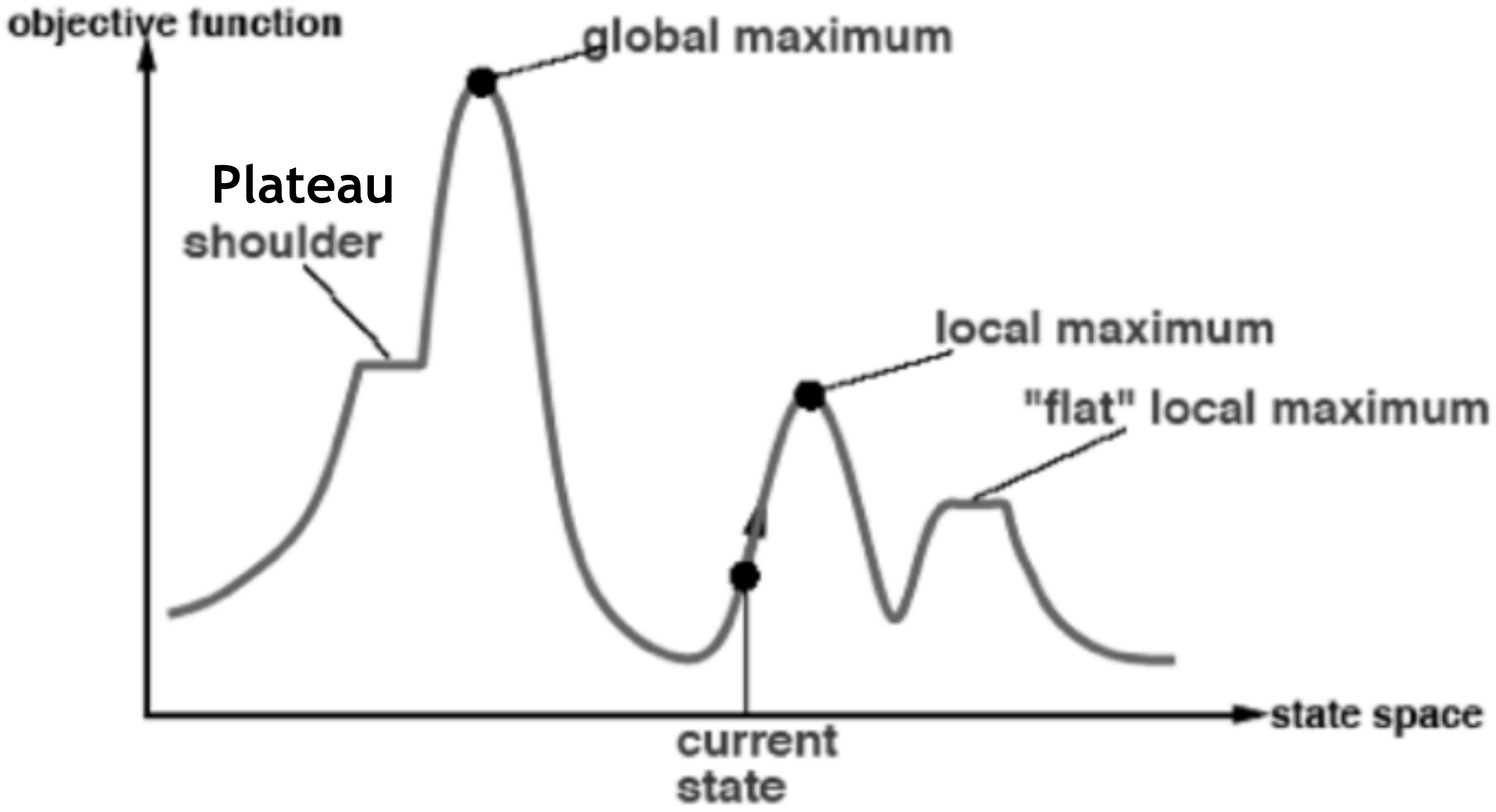
Fixed number of iterations

- Let’s say we run the previous algorithm for $|L|$ iterations
  - The runtime is $O(n|L|^2)$
  - The Viterbi runtime for a bigram model is $O(n|L|^2)$
- Here’s where it gets interesting:
  - Now imagine we were using a k-gram model Viterbi runtime: $O(n|L|^k)$
  - We could get arbitrarily better speedup!

Local Search

- Pros
  - This is an “any time” algorithm: stop any time and you will have a solution
- Cons
  - There is no guarantee that we found a good solution
  - **Local optima: to get to a good solution, you have to go through a bad scoring solution**
  - **Plateau: you get caught on a plateau and you can either go down or “stay the same”**

In Pictures


objective function
global maximum
Plateau
shoulder
local maximum
"flat" local maximum
state space
current state

Local Optima: Random Restarts

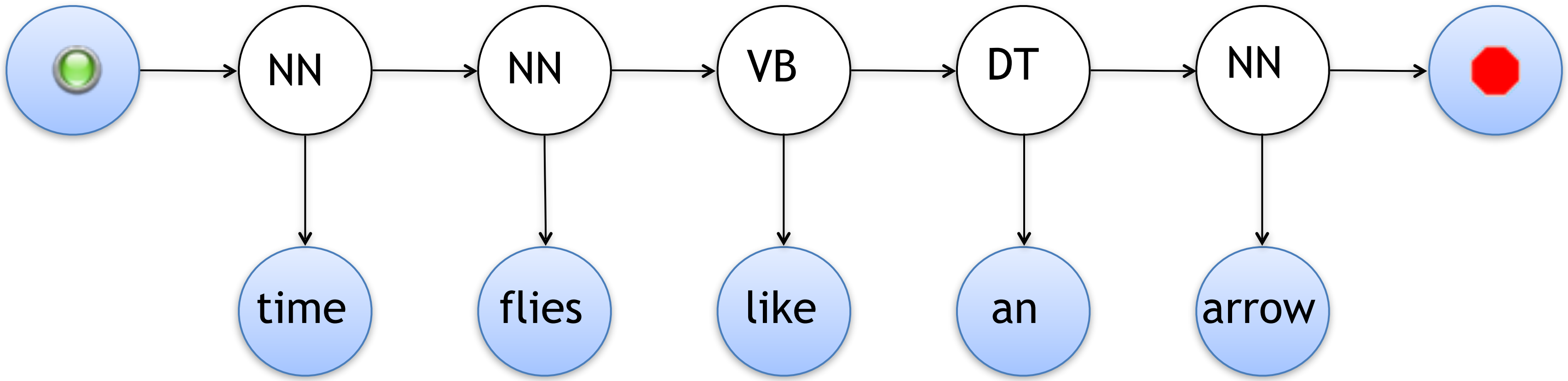
- Start from lots of different places
- Look at the score of the best solution
- **Pros**
  - Easy to parallelize
  - Easy to implement
- **Cons**
  - Lots of computational work
- Interesting paper:

Zhang et al. (2014) Greed is Good if Randomized: New Inference for Dependency Parsing. *Proc. EMNLP*.

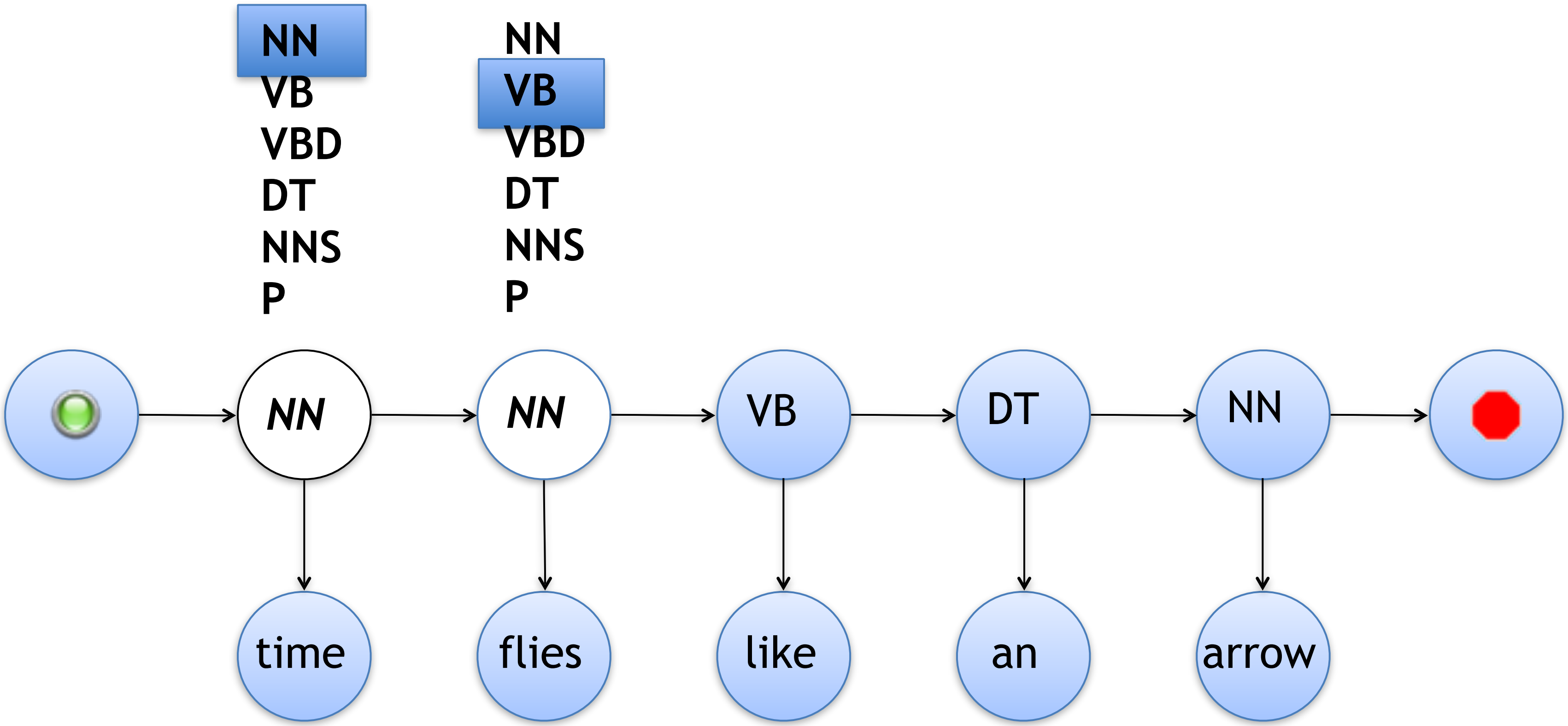
Local Optima: Take Bigger Steps

- We can use any neighborhood function!
- Why not use a bigger neighborhood function?
  - E.g., consider two words at once

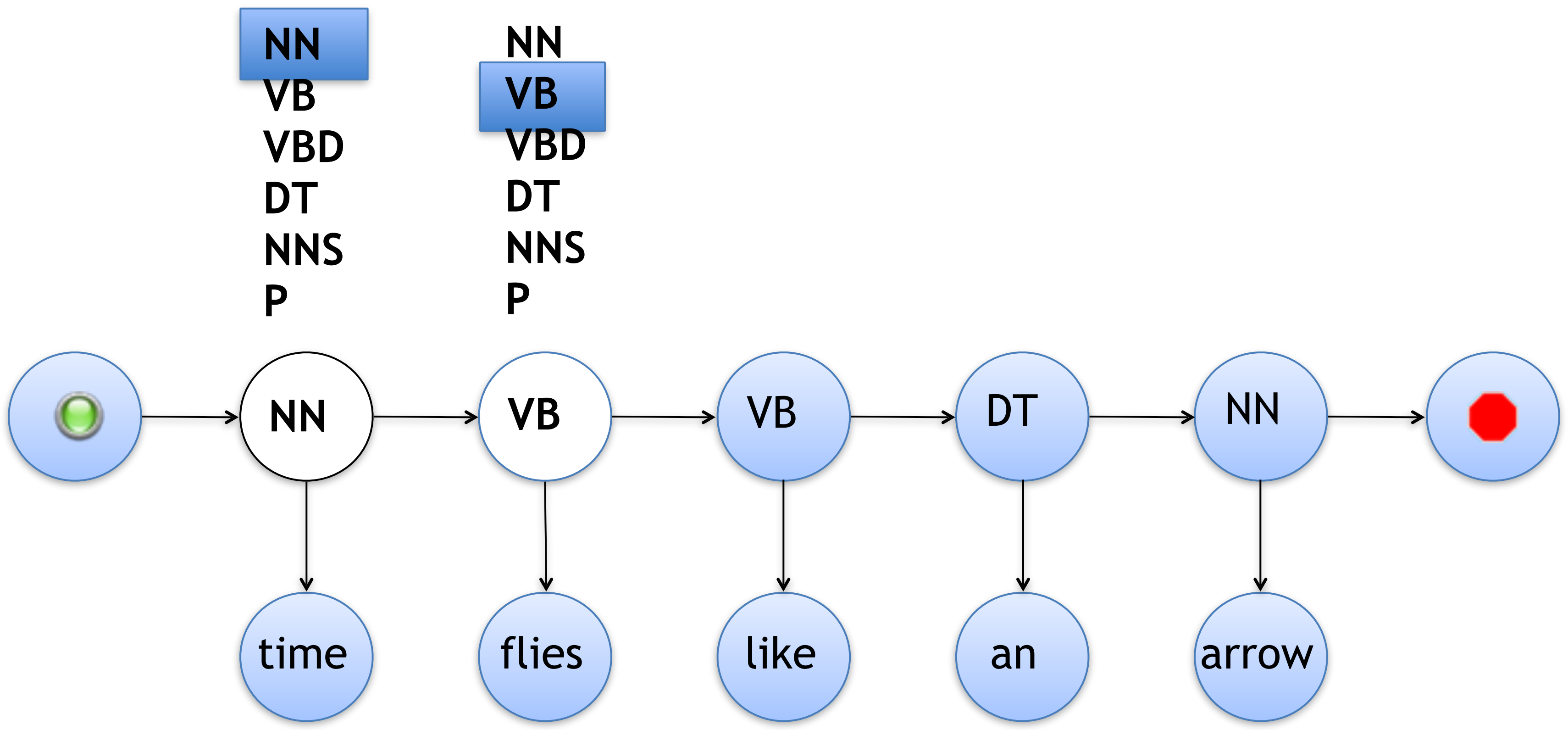
Local Search



Local Search



Local Search



Neighborhood Sizes

- **In general**: neighborhood size is **exponential in the number of variables** you are considering changing
- **But**, sometimes you can use dynamic programming (or other combinatorial algorithms) to search exponential spaces in polytime
  - Consider a sequence labeling problem where you have a bigram Markov model + some global features
  - **Example**: NER with constraints that say that all phrases should have the same label across a document

Stochastic Hill Climbing

- In general, there is no neighborhood function that will give you correct and efficient local search
  - Hill climbing may still be good enough!
  - "Some of my best friends are hill climbing algorithms!" (EM)
- Another variation
  - Replace the **arg max** with a **stochastic decision**: pick low-scoring decisions with some probability

Simulated Annealing

- View configurations as having an “energy”

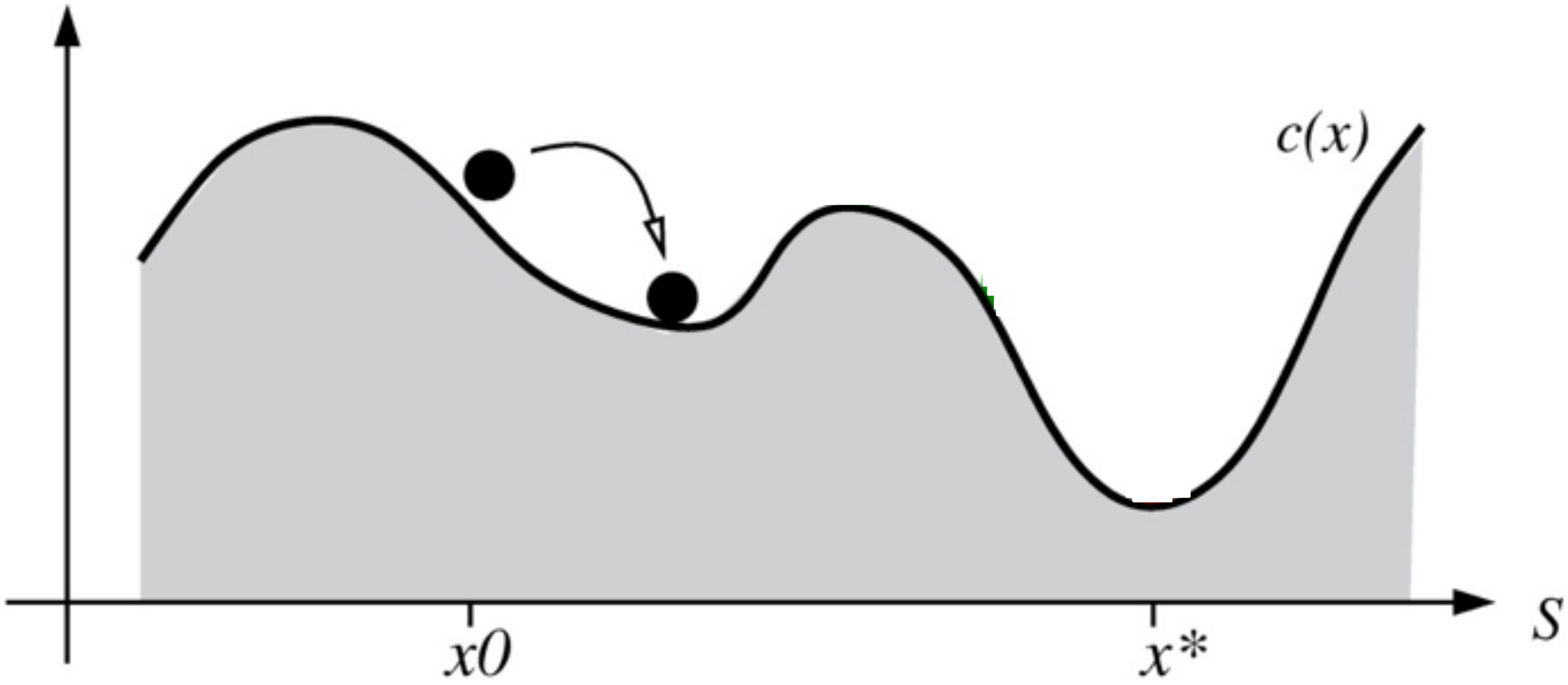
$$E(\boldsymbol{x}) = -\text{score}(\boldsymbol{x})$$

- Pick change in state by sampling

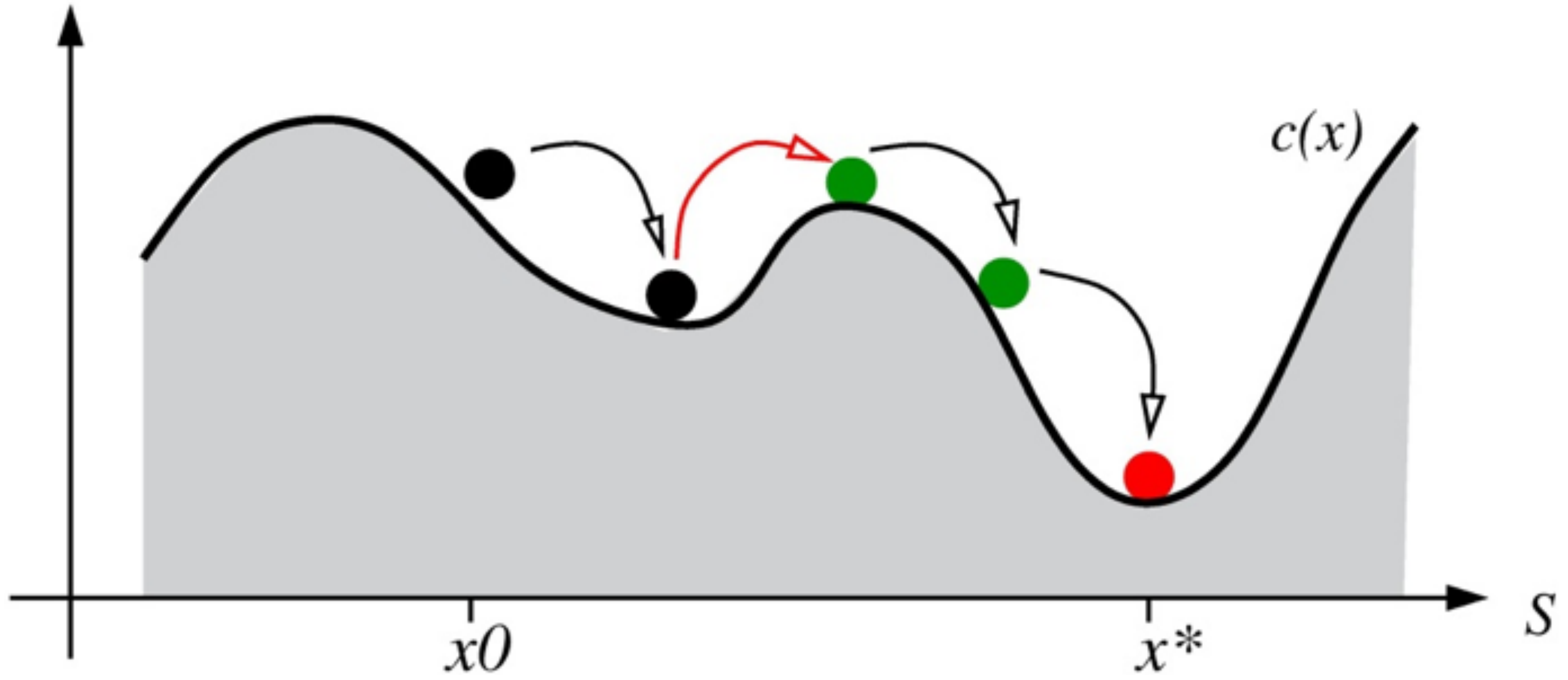
$$\propto e^{\frac{\Delta E}{T}}$$

- Start with a high “temperature” (model specific)
- Gradually cool down to T=0
- **Important: keep track of best scoring x so far!**

In Pictures



In Pictures



Simulated Annealing

- **We don’t have to compute the partition function, just differences in energy**
- In general:
  - Better solutions for slower annealing schedules
  - For probabilistic models, T=1 corresponds to Gibbs sampling (more in a few slides), provided certain conditions are met on the neighborhood function

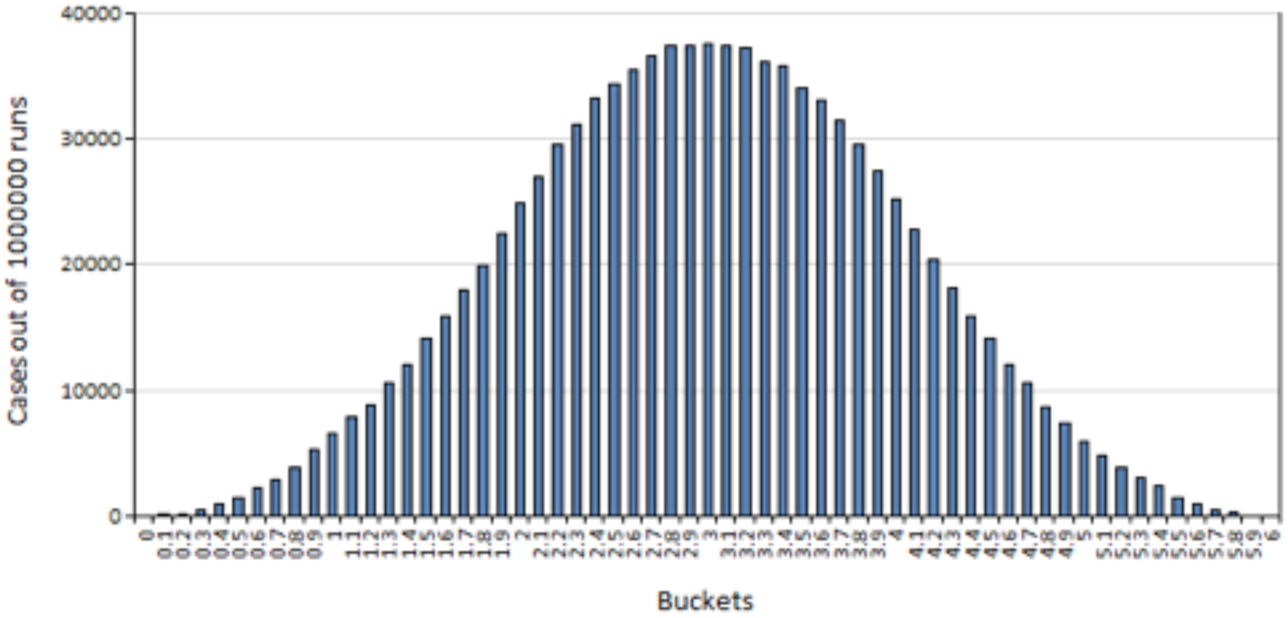
Whither Soft Inference?

- As we discussed, hard inference isn’t the only game in town
- We can use local search to approximate soft inference as well
  - Posterior distributions
  - Expected values of functions under distributions
- This brings us to the family of **Monte Carlo techniques**

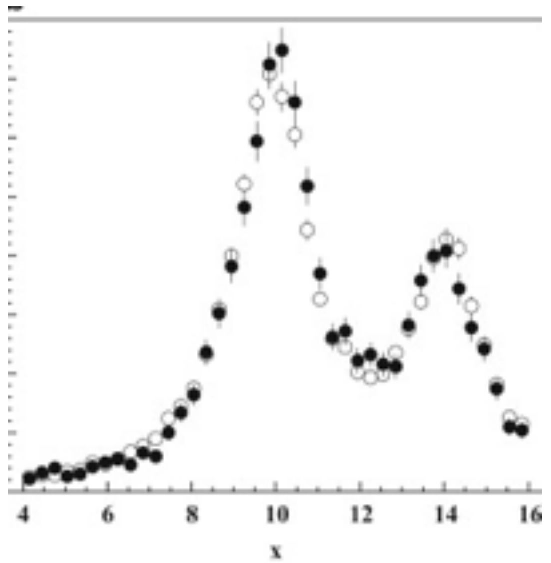
Monte Carlo Approximations

- Monte Carlo techniques let you
  - Approximately represent a distribution p(x) [x can be discrete, continuous, or mixed] using a collection of *N* **samples** from p(x)
  - Approximate marginal probabilities of x using samples from a joint distribution p(x,y)
  - Approximate expected values of f(x) using samples from p(x)

Monte Carlo approximation of a Gaussian distribution:



Monte Carlo approximation of a ??? distribution:



Monte Carlo Questions

- How do we generate samples from the target distribution?
  - Direct (or “perfect”) sampling
  - Markov-Chain MC methods (Gibbs, Metropolis-Hastings)
- How good are the approximations?

Monte Carlo Approximations

“Samples”

$$X^{(i)} \sim p(x), \quad \text{for } i = 1, \ldots, N$$

$$\hat{p}^{\mathrm{MC}}(x) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^{(i)}}(x)$$

Point mass at X$^{(i)}$

Monte Carlo Expectations

$$\hat{\mathbb{E}}^{\mathrm{MC}}[f(x)] = \int f(x)\hat{p}^{\mathrm{MC}}(x)dx$$

$$= \frac{1}{N}\sum_{i=1}^{N} f(X^{(i)})$$

Monte Carlo estimator of $\mathbb{E}[f(x)]$

Monte Carlo Expectations

- Nice properties
  - Estimator is **unbiased**
  - Estimator is **consistent**
  - Approximation error decreases at a rate of O(1/N), independent of the dimension of *X*
- Problems
  - We don’t generally know how to sample from *p*
  - When we do, the sampling scheme would be linear in dim(X)

Direct Sampling from *p*

- Sampling from *p* is generally hard
  - We may need to compute some very hard marginal quantities
- **Claim**. For every Viterbi/Inside-Outside algorithm there is a sampling algorithm that you get with the same "start up" cost
  - There is a question about this in the HW…
- But we want to use MC approximations when we can't run Inside-Outside!

Gibbs Sampling

- Markov chain Monte Carlo (MCMC) method
  - Build a Markov model
    - The states represent samples from *p*
    - Transitions = Neighborhoods from local search!
    - Transition probabilities constructed such that the MM's **stationary distribution** is *p*
  - MCMC samples are correlated
    - Taking every *m* samples can make samples more independent (How big should *m* be?)

Gibbs Sampling

- Gibbs sampling relies on the fact that sampling from p(a|b,c,d,e,f) is easier than sampling from p(a,b,c,d,e,f)
- Algorithm
  - We want *N* samples from $\mathbf{X} = \{X_1, \ldots, X_m\}$
  - The *i*th sample is $\mathbf{x}^{(i)} = \{x_1^{(i)}, \ldots, x_m^{(i)}\}$
  - Start with some **x**(0)
  - For each sample *i*=1,…,*N*
    - For each variable *j*=1,…,*m*
      - Sample $x_j^{(i)} \sim p(x_j \mid \mathbf{x}^{(i)} \backslash x_j^{(i)})$

The Beauty Part: No More Partitions

$$p(\mathbf{x}) \doteq \frac{u(\mathbf{x})}{Z}$$

$$\begin{aligned}
p(x_j \mid \mathbf{x}\backslash x_j) &= \frac{p(\mathbf{x})}{\sum_{x'_j \in \mathcal{X}_j} p(\mathbf{x}\backslash x_j, x'_j)} \\
&= \frac{u(\mathbf{x})/Z}{\sum_{x'_j \in \mathcal{X}_j} u(\mathbf{x}\backslash x_j, x'_j)/Z} \\
&= \frac{u(\mathbf{x})}{\sum_{x'_j \in \mathcal{X}_j} u(\mathbf{x}\backslash x_j, x'_j)}
\end{aligned}$$

Requirements

- There must be a positive probability path between any two states
- Process must satisfy **detailed balance**

$$\pi_i P_{ij} = \pi_j P_{ji}$$

  - Ie, this is a reversible Markov process
  - **Important:** This does ***not*** mean that you have to be able to reverse what happened at time (t) at time (t+1). **Why?**

Ensuring Detailed Balance

- **Option 1**: Visit all variables in a deterministic order that is independent of their current settings
- **Option 2**: Visit variables uniformly at random, independently of their current settings
- **Option 3**: Unfortunately, both of the above may not be feasible
  - Other orders are possible, but you have to prove that detailed balance obtains. This can be a pain.

Glossary

- **Mixing time**
  - How long until a Markov chain approaches the stationary distribution?
- **Collapsed sampling**
  - Marginalize some variables during sampling
  - Obviously: marginalize variables you don’t care about!
- **Block sampling**
  - Resample a block of random variables
  - This is exactly equivalent to the “large neighborhoods” idea – goal: reduce mixing time

Gibbs Sampling

- How do we sample trees?
- How do we sample segmentations?
- Key idea: sampling representation
  - Encode your random structure as a set of random variables
  - **Important: these will not (necessarily) be the same as your model**

Sampling Representations

独家:图解如何开高质量民主生活会

Sampling Representations

独家:图解如何开高质量民主生活会

独家:图解如何开高质量民主生活会

Ⓑ Ⓒ Ⓑ Ⓑ Ⓒ Ⓒ Ⓑ Ⓑ Ⓑ Ⓑ Ⓒ Ⓑ Ⓒ Ⓑ Ⓑ Ⓑ

$x_1 \; x_2 \, x_3 \ldots$

Sampling Representations

独家:图解如何开高质量民主生活会

独家:图解如何开高质量民主生活会

Ⓑ Ⓒ Ⓑ Ⓑ Ⓑ Ⓑ Ⓒ Ⓒ Ⓒ Ⓑ Ⓑ Ⓒ Ⓑ Ⓒ Ⓒ Ⓑ

$x_1 \, x_2 \, x_3 \ldots$

Sampling Representations

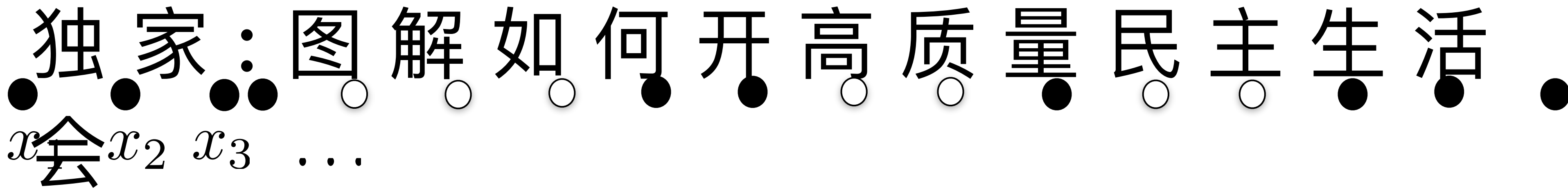
独家:图解如何开高质量民主生活会

独 家 : 图 解 如 何 开 高 质 量 民 主 生 活 会

x_1 x_2 x_3 $\ldots$

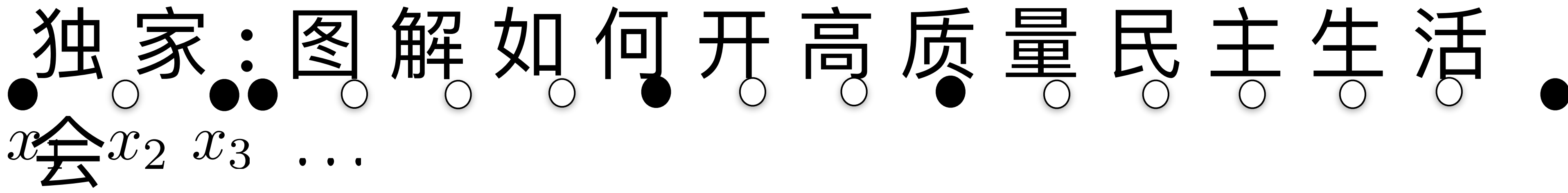
Sampling Representations

独家:图解如何开高质量民主生活会



Sampling Representations

独家:图解如何开高质量民主生活会


独 家 : 图 解 如 何 开 高 质 量 民 主 生 活 会
● ○ ●● ○ ○ ○ ● ○ ○ ● ○ ○ ○ ○ ○ ●
x_1 x_2 x_3 $\ldots$

Sampling Representations

- Requirements
  - Define reasonably sized neighborhoods
  - Model score changes should be easy to compute
- Standard tricks
  - Binary variables that indicate breaks
  - Random variables that indicate span lengths
  - Categorical random variables that indicate break,type
- Many papers just written on sampling representations for structured problems!

How Things Go Wrong

- Three common failure modes
  - Mixing time is awful
  - Sampling density is intractable/incomputable
  - Variance of estimates (e.g., of expectations) is too high
- This is why MCMC methods are still an active area of research
- We consider two (potential) solutions that rely on **proposal distributions**

Using Proposal Distributions

- **Idea**: sample from a distribution that "looks like" the distribution you want to sample from, i.e. $p(x_j \mid \mathbf{x} \backslash x_j)$ or $p(x)$
  - Common trade off: good approximation of p vs. easy to sample from
- Then perform some kind of correction using *p* (or, usually, *p*C*)
  - Metropolis-Hastings: possibly reject sample
  - Importance sampling: reweight sample

What Proposal Distribution?

$$p(\mathbf{x}) > 0 \implies q(\mathbf{x}) > 0$$

- Specifics depend on your problem
  - Sample from a bigram HMM's posterior distribution as a proposal for a *k*-gram HMM
  - Sample from a Gaussian as a proposal for some other continuous density
  - Sample from an unconditional distribution as a proposal for a conditional distribution
- In general: good proposal distributions have **heavier tails**

Metropolis Hastings Sampling

- Very simple strategy for incorporating a proposal distribution
- Can be used to propose full ensemble of variables, a single variable, or anything in between
- Standard uses
  - Sampling continuous variables (e.g., sample from Gaussian and accept into non-Gaussian distribution)
  - Sample sequence or tree from PCFG/HMM and accept into model with non-local factors

Metropolis Hastings Sampling

- The MH algorithm works as follows
- For each block of variables you are resampling
  - Sample $\mathbf{x}' \sim q(\mathbf{x}' \mid \mathbf{x})$
  - Accept this sample with probability

$$A(\mathbf{x} \to \mathbf{x}') = \min \left\{ 1, \frac{p(\mathbf{x}')}{p(\mathbf{x})} \frac{q(\mathbf{x} \mid \mathbf{x}')}{q(\mathbf{x}' \mid \mathbf{x})} \right\}$$

  - If accepted, update **x**
  - Otherwise **x** remains the same

Metropolis Hastings Sampling

- Note: with an unconditional proposal

$$A(\mathbf{x} \to \mathbf{x}') = \min\left\{1, \frac{p(\mathbf{x}')}{p(\mathbf{x})}\frac{q(\mathbf{x})}{q(\mathbf{x}')}\right\}$$

- Also note: you only need to be able to sample from *p* and *q* and evaluate them up to a fixed factor (e.g., partition)

Metropolis-Hastings

- Pros
  - A paper cited 18,000 times can’t be wrong!
  - Hand-crafted proposal distributions give you the ability to improve performance
- Cons
  - Keep track of your rejections
  - Variance of computed quantities can be exceedingly high

Importance Sampling

- MH samples can be highly correlated -> high variance of MC estimates of expectations
- Importance sampling is a technique for **reducing variance** (albeit by increasing bias)
- Intuition
  - Rather than rejecting bad samples, down-weight them appropriately
- Benefits
  - Lower variance
  - Biased, but still consistent
  - Estimate of Z

Importance Sampling

- Given $p(\mathbf{x}) = \frac{u(\mathbf{x})}{Z}$

  $$\text{where } Z = \sum_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x})$$

- We define the **unnormalized weight**

Importance Sampling

- Given $p(\mathbf{x}) = \frac{u(\mathbf{x})}{Z}$ and importance dist. $q(\mathbf{x})$

  $\text{where } Z = \sum_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x})$ $p(\mathbf{x}) > 0 \implies q(\mathbf{x}) > 0$

- We define the **unnormalized weight**

Importance Sampling

- Given $p(\mathbf{x}) = \frac{u(\mathbf{x})}{Z}$ and importance dist. $q(\mathbf{x})$

  $$\text{where } Z = \sum_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x})$$

  $$p(\mathbf{x}) > 0 \implies q(\mathbf{x}) > 0$$

- We define the **unnormalized weight** function

  $$w(\mathbf{x}) = \frac{u(\mathbf{x})}{q(\mathbf{x})}$$

Importance Sampling

- Given $p(\mathbf{x}) = \frac{u(\mathbf{x})}{Z}$ and importance dist. $q(\mathbf{x})$

  $$\text{where } Z = \sum_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x})$$

  $$p(\mathbf{x}) > 0 \implies q(\mathbf{x}) > 0$$

- We define the **unnormalized weight function**

  $$w(\mathbf{x}) = \frac{u(\mathbf{x})}{q(\mathbf{x})}$$

- We can now write

  $$Z = \sum_{\mathbf{x} \in \mathcal{X}} w(\mathbf{x}) q(\mathbf{x})$$

Importance Sampling

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} w(\mathbf{x}) q(\mathbf{x})$$

Notice that this has the form of an expected value of w(x) under q:

$$Z = \mathbb{E}_{q(\cdot)} w(\mathbf{x})$$

Importance Sampling

$$Z = \sum_{\mathbf{x} \in \mathcal{X}} w(\mathbf{x}) q(\mathbf{x})$$

Notice that this has the form of an expected value of w(x) under q:

$$Z = \mathbb{E}_{q(\cdot)} w(\mathbf{x})$$

We can replace this with a Monte Carlo estimate

$$\hat{Z} = \hat{\mathbb{E}}^{\mathrm{MC}}_{q(\cdot)} w(\mathbf{x})$$

$$\hat{Z} = \frac{1}{N} \sum_{i=1}^{N} w(\mathbf{x}^{(i)})$$

Importance Sampling

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x}^{(i)})$$

This lets us derive the following approximation:

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{w(\mathbf{x})\hat{q}(\mathbf{x})}{\hat{Z}}$$

Importance Sampling

$$\hat{Z} = \frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x}^{(i)})$$

This lets us derive the following approximation:

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{w(\mathbf{x})\hat{q}(\mathbf{x})}{\hat{Z}}$$

Intuitively, we have reweighted each sample $\mathbf{x}^{(i)}$ from q(**x**) with an **importance weight**

$$\frac{w(\mathbf{x}^{(i)})}{\sum_{j=1}^{N} w(\mathbf{x}^{(j)})}$$

Importance Sampling

IS Expectations are defined straightforwardly as

$$\begin{aligned}
\hat{\mathbb{E}}^{\text{IS}}_{p(\cdot)}\left[f(\mathbf{x})\right] &= \sum_{i=1}^{N}\left[\frac{w(\mathbf{x}^{(i)})}{\sum_{j=1}^{N} w(\mathbf{x}^{(j)})} f(\mathbf{x}^{(i)})\right] \\
&= \frac{1}{\hat{Z}}\sum_{i=1}^{N} w(\mathbf{x}^{(i)}) f(\mathbf{x}^{(i)}) \\
&= \frac{1}{\hat{Z}}\sum_{i=1}^{N} \frac{u(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})} f(\mathbf{x}^{(i)})
\end{aligned}$$

Importance Sampling

- You can show
  - That the IS estimator is **biased**
  - That the IS estimator is **consistent**
  - That the IS estimator obeys a central limit theorem with asymptotic variance

$$\frac{1}{N}\sum_{\mathbf{x}\in\mathcal{X}}\frac{p^2(\mathbf{x})}{q(\mathbf{x})}\left[f(\mathbf{x})-\mathbb{E}_{p(\mathbf{x}')}f(\mathbf{x}')\right]^2$$

  - That the IS estimator is more efficient than rejection sampling

Particle Filtering

- **Particle filtering** is a special kind of importance sampling
  - It creates proposal distributions by conditioning only on the **past** and **current observations**
  - Each “particle” is a single sample that is built up progressively across time
    - This looks a lot like **beam search** except you sample a single decision at each time step and then discard anything else
  - As time progresses, you figure out that some particles have a bad **importance weight** and others are good
    - Key idea: throw out low-weight particles and duplicate high weight particles

Summary

- Monte Carlo techniques are a huge field of research
  - This is a survey of the important ones that are used in structured prediction
- We will return to these methods when we talk about Bayesian unsupervised learning