Title: PSI 2018/2019 - Machine Learning - Lecture 5

Speakers: Lauren Hayward Sierens

Collection: PSI 2018/2019 - Machine Learning (Hayward Sierens)

Date: March 29, 2019 - 9:00 AM

URL: http://pirsa.org/19030057

## Outline for today:

- Monte Carlo sampling
- Summary of Week #1 and recap of hyperparams in NNs

## Monte Carlo (MC) Sampling

Why study MC sampling within this course?

↳ for generating data samples that can be used when training NNs (eg: Homework 2)

↳ for use within RL and generative modelling (see Lectures 9-15)

Consider a system where we want to calculate expectation values of quantities $Q$:

$$\langle Q \rangle = \sum_{\mu}^{+} P_\mu Q_\mu$$

In classical stat. mech.: $P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$, $Z = \sum_{\mu}^{-} e^{-\beta E_\mu}$

where
- $E_\mu$ = energy of state $\mu$
- $\beta = \frac{1}{k_B T}$

The sums $\sum_{\mu}'$ are over exponentially many states

$\Rightarrow$ it is analytically intractable

Idea of MC: estimate $\langle Q \rangle$ by summing over $M \ll 2^N$ states

$\mu_1, \mu_2, \cdots, \mu_M$

If each $\mu_i$ $(1 \leq i \leq M)$ is selected from the $2^N$ states with prob. $P_{\mu_i} = e^{-\beta E_{\mu_i}} / Z$,

then:

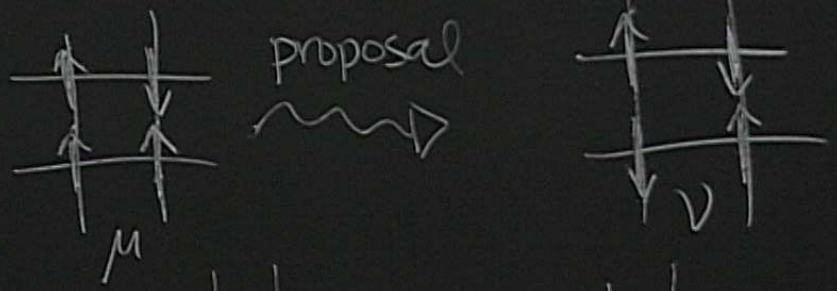
$$\langle Q \rangle \approx Q_M = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu_i}$$

This estimator uses "importance sampling" (the more probable states are more likely to be sampled).

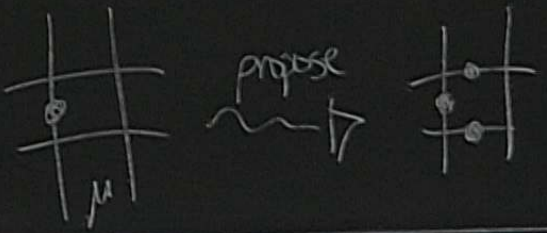
To implement this sampling, we can use Markov chain MC (MCMC)

↳ use current state $\mu$ to propose the next state $\nu$

eg) Ising model:




Ising gauge theory:



Given $\mu$, propose $\nu$, where:

- in principle, any proposal is allowed
  - ↳ single-spin flip (Homework 2)
  - ↳ non-local "cluster" of spin flips
  - ⋮
- We can choose to accept or reject the proposal
- the probability of moving from $\mu$ to $\nu$ is the transition prob. $T(\mu \to \nu)$, with $\sum_{\nu} T(\mu \to \nu) = 1$

We can choose the set $\{T(\mu \to \nu)\}$ in any way as long as it satisfies.

(1) Detailed balance (DB)

$$\underbrace{\sum_\nu P_\nu T(\nu \to \mu)}_{\text{rate of transition into } \mu} = \underbrace{\sum_\nu P_\mu T(\mu \to \nu)}_{\text{rate of transition out of } \mu}$$

Can satisfy this if: $\boxed{P_\nu (T\nu \to \mu) = P_\mu T(\mu \to \nu)}$ DB

So when $p_\mu = \frac{1}{Z} e^{-\beta E_\mu}$

$$\frac{T(\mu \to \nu)}{T(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)}$$

② Ergodicity:
There is a non-zero prob. of transitioning from $\mu$ to any $\nu$

$\mu \to \ldots \to \nu$

$\mu \to \nu$)

transition

$\mu$

$p_\mu T(\mu \to \nu)$ DB

Notation:

$$T(\mu \to \nu) = g(\mu \to \nu)\, A(\mu \to \nu)$$

$g(\mu \to \nu)$: selection prob.

$A(\mu \to \nu)$: acceptance prob.

For a given set of selection probs., we want the acceptance probs. $A(\mu \to \nu)$ to be as close to 1 as possible.

From DB:

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = \frac{g(\nu \to \mu)}{g(\mu \to \nu)} e^{-\beta(E_\nu - E_\mu)}$$

Metropolis algorithm:

$$A(\mu \to \nu) = \min\left(1, \frac{g(\nu \to \mu)}{g(\mu \to \nu)} e^{-\beta(E_\nu - E_\mu)}\right)$$

↳ this we ensure that $A(\mu \to \nu)$ or $A(\nu \to \mu)$ is equal to 1

# Pseudocode for Monte Carlo sampling

(generates M samples $\mu_1, \ldots, \mu_M$ from the dist. $P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$)

Choose $\mu_1$ at random

for $i = 2, \ldots, M$:

propose a move to state $\nu$ with prob. $g(\mu_i \to \nu)$

generate $r$ from a uniform random dist. $\mathcal{U}[0,1)$

ng

dist. $P_\mu = \frac{1}{Z} e^{-\beta E_\mu}$)

$\sim U[0,1)$

if $r < A(\mu \to \nu)$:

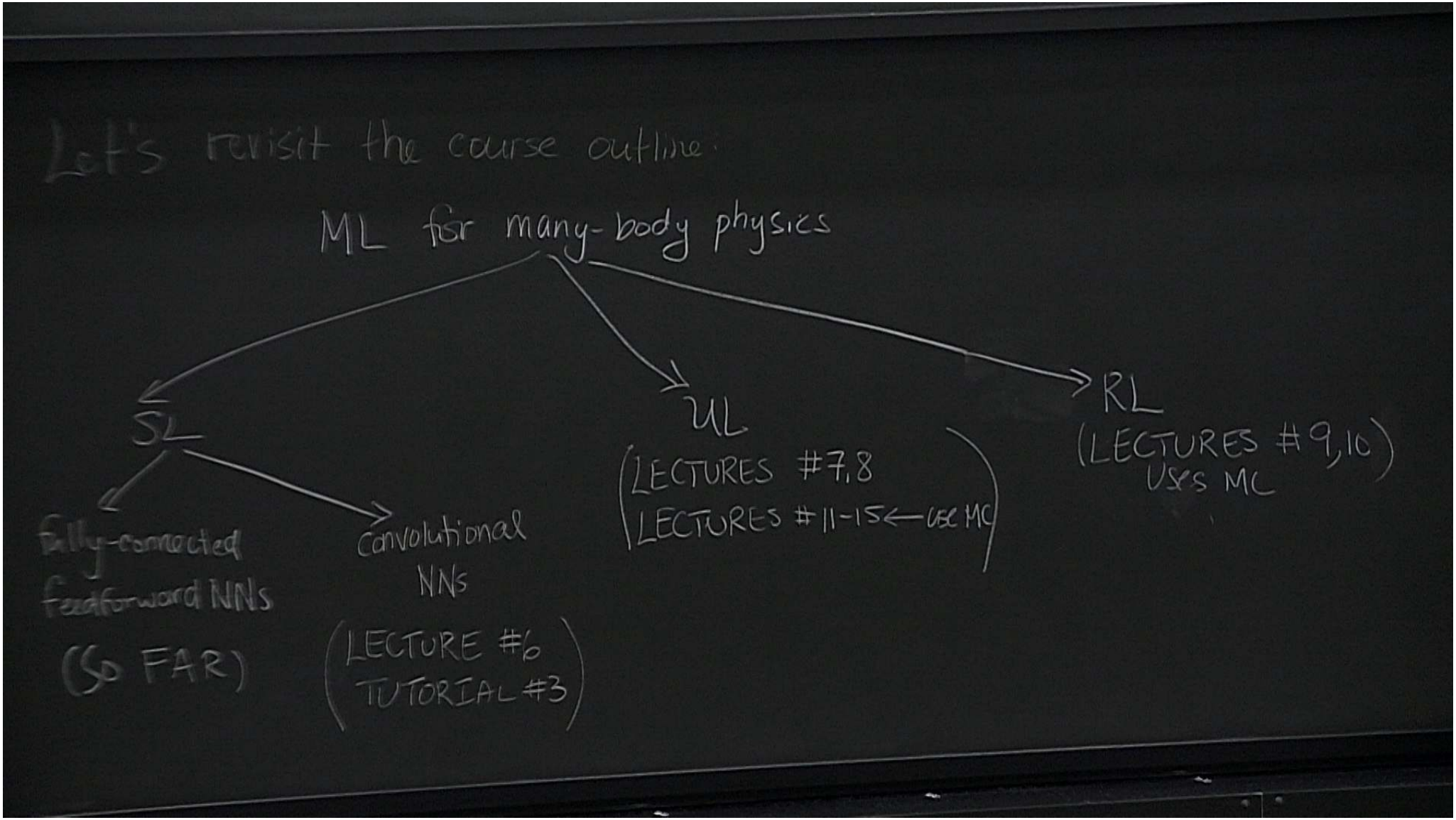
$\mu_i = \nu$

else:

$\mu_i = \mu_{i-1}$

}

(Note: in practice we ignore the first several samples since they are not truly sampled from $P_\mu$)


Let's revisit the course outline.
ML for many-body physics
SL
UL
RL
(LECTURES #9,10)
Uses MC
(LECTURES #7,8
LECTURES #11-15 ← use MC)
fully-connected feedforward NNs
(So FAR)
convolutional NNs
(LECTURE #6
TUTORIAL #3)

# Hyperparams in feedforward NNs

- learning rate $\eta$, and how it changes with epoch
- # of layers (how "deep" is the network)
- # of neurons in the hidden layers
- activation functions
- cost function (MSE, CE, ...)

- how much regularization $\lambda$ we add to $C$
- masking prob. in dropout
- training algorithm chosen (GD, SGD, Adam, ...)
  - → add momentum with hyperparam. $\gamma$
  - ↓ Size of mini batch $B$

- learning rate $\eta$, and how it changes with epoch
- # of layers (how "deep" is the network)
- # of neurons in the hidden layers
- activation functions on each layer
- cost function (MSE, CE, ...)

- how much regularization $\lambda$ we add to $C$
- masking prob. in dropout
- training algorithm chosen (GD, SGD, Adam, ...)
  - ↓ Size of mini batch $B$
  - → add momentum with hyperparam. $\gamma$
- how we partition the data into training, validation, testing
- adding noise to data
- how the weights & biases are initialized