Title: LECTURE: Generative Modelling

Speakers: Roger Melko

Collection: Quantum and AI Career Trajectories Mini-Course: Computational Methods and their Applications

Date: May 10, 2023 - 11:00 AM

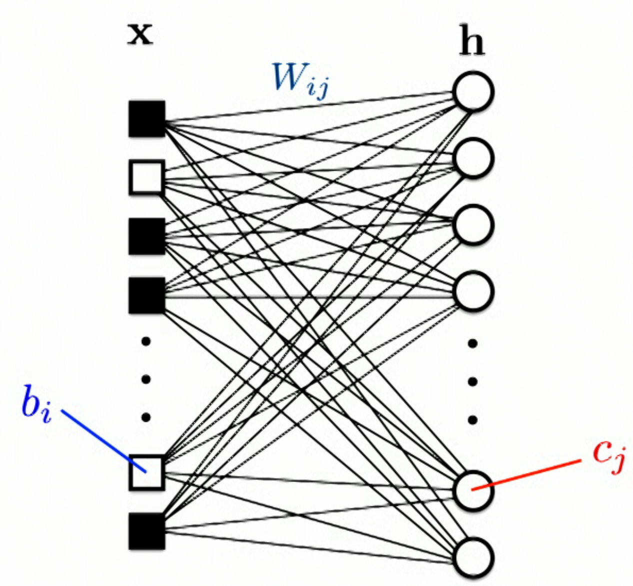
URL: https://pirsa.org/23050095

Abstract: TBC

Restricted Boltzmann Machine

Smolensky, Hinton, Salakhutdinov, Bengio

N visible units $\quad$ n_h hidden units



$x_i = 0, 1$ $\quad$ $h_j = 0, 1$

Like a Hopfield network, RBMs are "energy-based" models:

$$p_\lambda = \frac{1}{Z_\lambda} e^{-E_\lambda(\mathbf{x},\mathbf{h})}$$

joint probability distribution

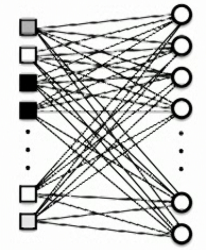
$$E_\lambda(\mathbf{x}, \mathbf{h}) = -\sum_{ij} W_{ij} x_i h_j - \sum_i b_i x_i - \sum_j c_j h_j$$

"Training" means tuning the machine parameters to get the marginal distribution $p_\lambda(\mathbf{x})$ to approximate the (unknown) target distribution

$$\lambda = \{W, b, c\} \qquad p_\lambda(\mathbf{x}) = \sum_{\mathbf{h}} p_\lambda(\mathbf{x}, \mathbf{h})$$

model parameters

Block Gibbs Sampling



RBM: being “restricted” is a special property that allows sampling one layer at a time.

$$x_0 \to h_0 \to x_1 \to h_1 \to \cdots \to x_k \to h_k$$

Each layer is updated with ***conditional*** probabilities

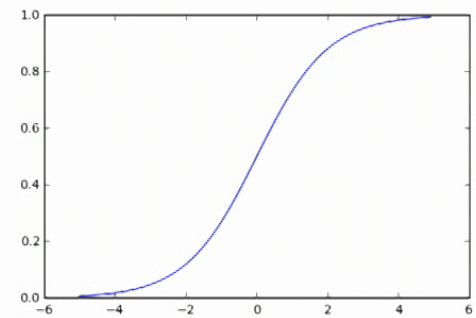
$$p_\lambda(\mathbf{x}|\mathbf{h}) = \frac{p_\lambda(\mathbf{x},\mathbf{h})}{p_\lambda(\mathbf{h})} = \prod_i p(x_i|\mathbf{h}) \qquad p_\lambda(\mathbf{h}|\mathbf{x}) = \frac{p_\lambda(\mathbf{x},\mathbf{h})}{p_\lambda(\mathbf{x})} = \prod_j p(h_j|\mathbf{x})$$

$$p(x_i = 1|\mathbf{h}) = \sigma\Big(\sum_j W_{ij}h_j + b_i\Big)$$
$$p(h_j = 1|\mathbf{x}) = \sigma\Big(\sum_i W_{ij}x_i + c_j\Big)$$

} The firing rate of a (stochastic) neuron

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Sigmoid function



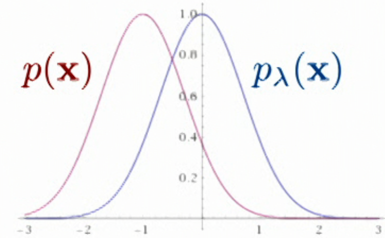
Training the RBM

G. Hinton, Neural computation 14, 1771 (2002)

Training means tuning the machine parameters to minimize the difference between the marginal distribution $p_\lambda(\mathbf{x}) = \sum_{\mathbf{h}} p_\lambda(\mathbf{x}, \mathbf{h})$ and the (unknown) physical "target" distribution

Define an optimization problem: minimize the ***Kullback-Leibler divergence***

$$\mathrm{KL}(p||p_\lambda) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{p_\lambda(\mathbf{x})} \geq 0$$



- A non-symmetric measure of the distance between two distributions
- Always positive, and zero iff $p = p_\lambda$

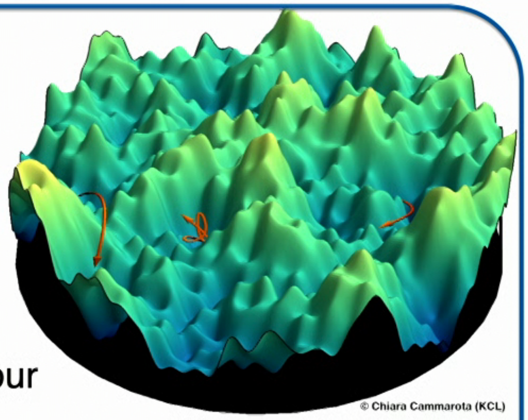
$$\mathrm{KL}(p||p_\lambda) = \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) - \sum_{\mathbf{x}} p(\mathbf{x}) \log p_\lambda(\mathbf{x})$$

the entropy of p — depends on the parameters over which to optimize

$$= -\langle \log p_\lambda(\mathbf{x}) \rangle_p \approx -\sum_i \log p_\lambda(\mathbf{x}_i)$$

Equivalent to maximizing the "log-likelihood" $\mathcal{L} = \langle \log p_\lambda(\mathbf{x}) \rangle_p$

Stochastic Gradient Descent



The optimization landscape is thus obtained - minimize using gradient descent

$$\lambda' = \lambda - \eta \nabla \mathcal{L} \qquad \lambda = \{W, b, c\}$$

The full gradient is too costly to calculate. Instead sample some number m of your dataset, and perform *stochastic* gradient descent

$$\frac{1}{m} \sum_{j=1}^{m} \nabla \mathcal{L}(\mathbf{x}_j) \approx \nabla \mathcal{L}$$

"mini-batch" size = m

$$\frac{\partial \mathcal{L}}{\partial \lambda} = -\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}) \frac{\partial E}{\partial \lambda} + \sum_{\mathbf{x},\mathbf{h}} p(\mathbf{x}, \mathbf{h}) \frac{\partial E}{\partial \lambda}$$

- The first term is computationally easy to calculate.
- The second term is hard. Requires a MCMC to generate samples from the station distribution of the machine. In practice a short chain of k steps is run

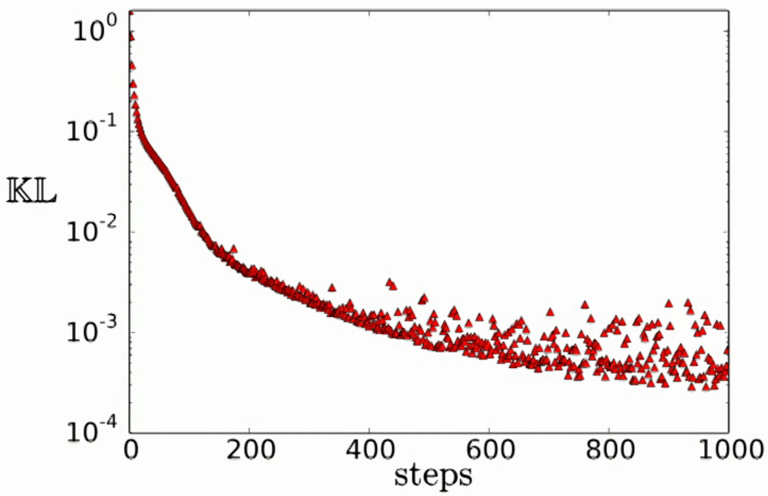
"Contrastive Divergence"

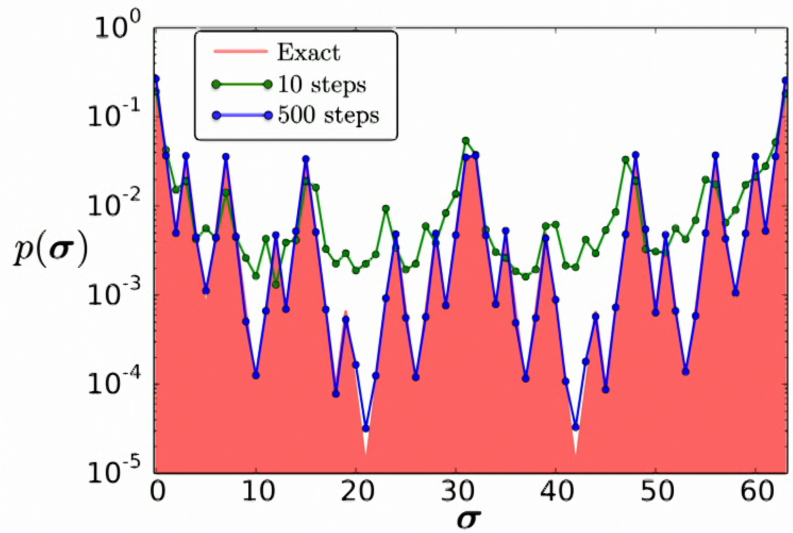
CD_k

Stochastic Gradient Descent

Torlai and RGM, Phys. Rev. B 94, 165134 (2016)

Example: 1D Ising model with 6 spins, trained using CD_5





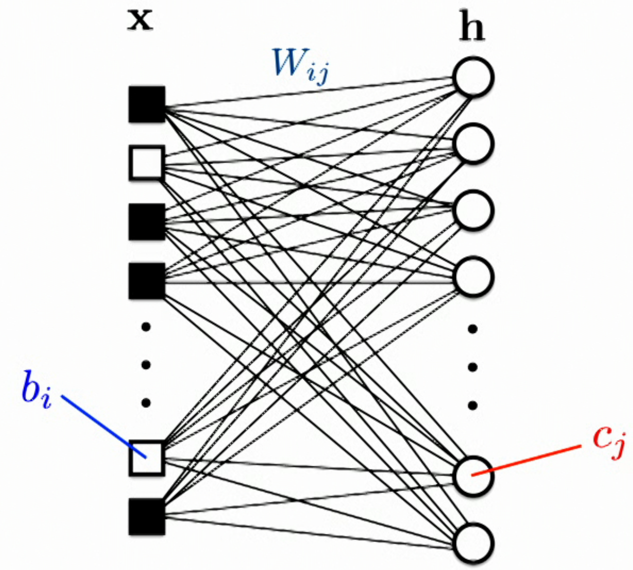
KL not possible to calculate in the general case for larger N.

Other metrics, like physical observables, could be used to validate the training…

Restricted Boltzmann Machine

Smolensky, Hinton, Salakhutdinov, Bengio

N visible units $\quad$ n_h hidden units



$x_i = 0, 1 \qquad h_j = 0, 1$

Like a Hopfield network, RBMs are "energy-based" models:

$$p_\lambda = \frac{1}{Z_\lambda} e^{-E_\lambda(\mathbf{x},\mathbf{h})}$$

joint probability distribution

$$E_\lambda(\mathbf{x},\mathbf{h}) = -\sum_{ij} W_{ij} x_i h_j - \sum_i b_i x_i - \sum_j c_j h_j$$

"Training" means tuning the machine parameters to get the marginal distribution $p_\lambda(\mathbf{x})$ to approximate the (unknown) target distribution

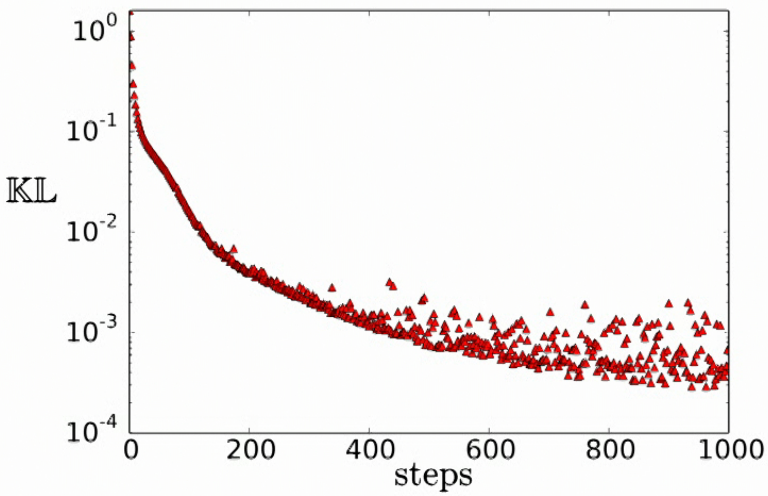
$$\lambda = \{W, b, c\} \qquad p_\lambda(\mathbf{x}) = \sum_{\mathbf{h}} p_\lambda(\mathbf{x},\mathbf{h})$$

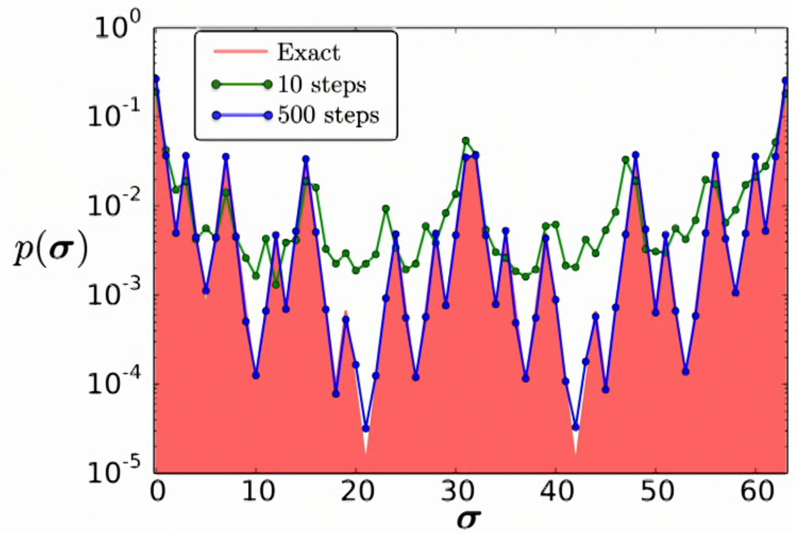
model parameters

Stochastic Gradient Descent

Torlai and RGM, Phys. Rev. B 94, 165134 (2016)

Example: 1D Ising model with 6 spins, trained using CD_5





KL not possible to calculate in the general case for larger N.

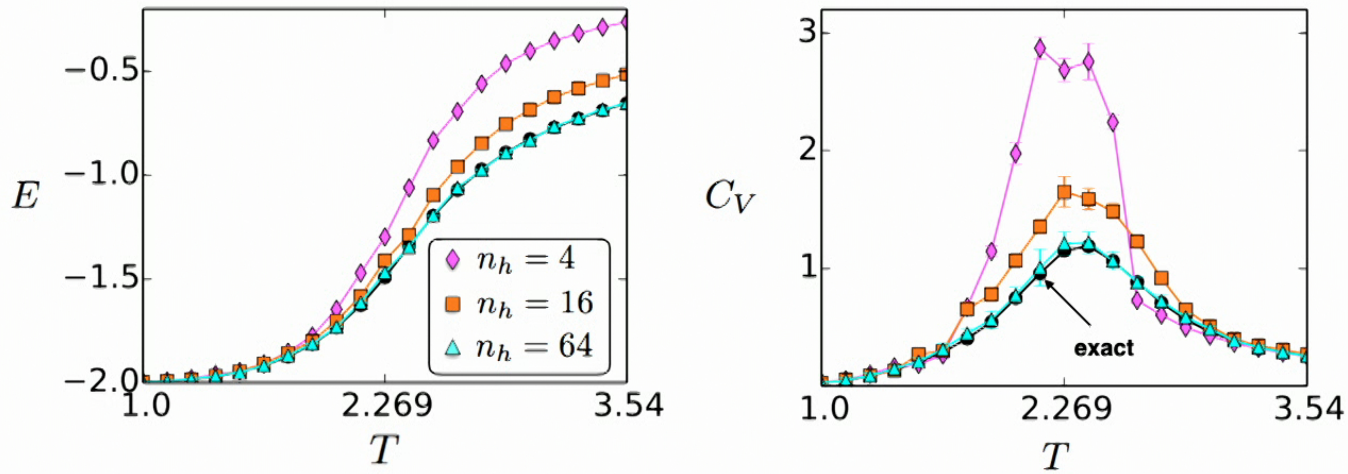
Other metrics, like physical observables, could be used to validate the training…

Learning Thermodynamics of the Ising model

Torlai and RGM, Phys. Rev. B 94, 165134 (2016)

Results from the generative model, after training: $\langle \mathcal{O} \rangle = \frac{1}{N_{\mathrm{MCS}}} \sum_{\mathbf{x}} \mathcal{O}_{\mathbf{x}}$ $\mathbf{x}$ from standard MCMC = "exact"

$N = 64$



This shows us in this example that the number of hidden units required for accurate generative modelling is approximately the same as the number of hidden units.

See: Chen, Cheng, Xie, Wang, Xiang, Phys. Rev. B 97, 085104 (2018)

Note: The number of measurements required for training & sampling also affects efficiency

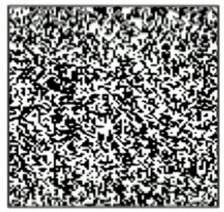
Learning wavefunctions 1

Torlai, Mazzola, Carrasquilla, Troyer, RGM, Carleo, Nature Physics (2018).
doi:10.1038/s41567-018-0048-5, arXiv:1703.05334

In the case where the wavefunction is real and positive in a certain basis

$$\psi_\lambda(\mathbf{x}) \propto \sqrt{p_\lambda(\mathbf{x})}$$

Train with samples in the S^z basis



and afterwords calculate estimators from samples produced on the trained machine

$$\langle \mathcal{O}^{\mathrm{D}} \rangle = \sum_{\mathbf{x}} p_\lambda(\mathbf{x}) \mathcal{O}_{\mathbf{x}}$$

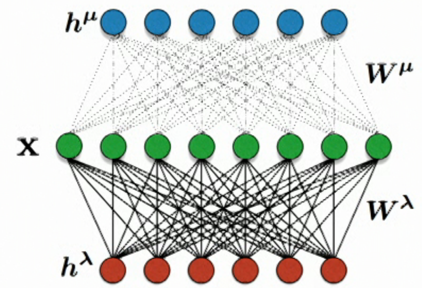
$$\langle \mathcal{O}^{\mathrm{OD}} \rangle = \sum_{\mathbf{x}\mathbf{x}'} \sqrt{p_\lambda(\mathbf{x})} \sqrt{p_\lambda(\mathbf{x}')} \mathcal{O}_{\mathbf{x}\mathbf{x}'} = \sum_{\mathbf{x}} p_\lambda(\mathbf{x}) \sum_{\mathbf{x}'} \frac{\sqrt{p_\lambda(\mathbf{x}')}}{\sqrt{p_\lambda(\mathbf{x})}} \mathcal{O}_{\mathbf{x}\mathbf{x}'}$$

"local" estimator

Learning wavefunctions 2

Torlai, Mazzola, Carrasquilla, Troyer, RGM, Carleo, Nature Physics (2018).
doi:10.1038/s41567-018-0048-5, arXiv:1703.05334

For a more generic wavefunction with amplitude and phase, represent both with hidden units



$$\psi_{\lambda,\mu}(\mathbf{x}) \propto \sqrt{p_\lambda(\mathbf{x})}e^{i\phi_\mu(\mathbf{x})}$$

Now, different bases are needed to estimate both the amplitude and phases of the target state.

$$\mathcal{L} = \sum_b^{N_b} \sum_{\mathbf{x}_b} \log |\psi_{\lambda,\mu}(\mathbf{x}_b)|^2$$

$N_b =$ number of bases

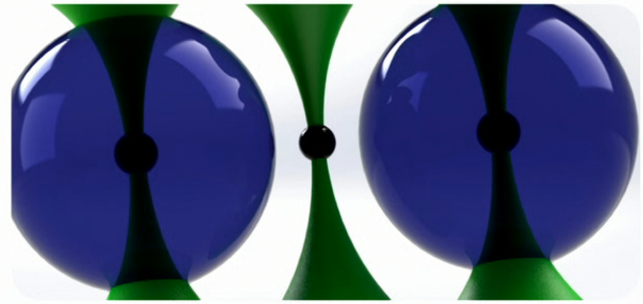
$\{X, X, Z, Z, \ldots\}, \{Z, X, X, Z, \ldots\}, \{Z, Z, X, X, \ldots\},$

state rotated into basis b with the appropriate unitary

From this, calculate $\nabla_\lambda \mathcal{L}$ and $\nabla_\mu \mathcal{L}$, use stochastic gradient descent, etc.

In practice, training is done in two stages: learning of amplitude first, then optimization of the phase parameters.

Rydberg atom arrays



- Neutral atoms (Rb, Sr) are loaded into a lattice formed by an array of optical tweezers
- Atoms can be in their ground state, or an excited state with a large principle quantum number (a Rydberg state). They form a strongly-interacting system.
- Single-atom resolved fluorescent imaging provides projective measurements
- Arrays of atoms are currently used for simulation (groundstates, critical phenomena), solving combinatorial optimization problems

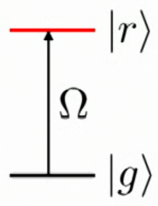
|QuEra>

PASQAL

planqc

Rydberg Blockade Hamiltonian

Jaksch, Cirac, Zoller, Rolston, Cote, Lukin, Phys. Rev. Lett. 85, 2208 (2000)
Lukin, Fleischhauer, Cote, Duan, Jaksch, Cirac, Zoller, Phys. Rev. Lett. 87, 037901 (2001)
Fendley, Sengupta, Sachdev, Phys. Rev. B 69, 075106 (2004)

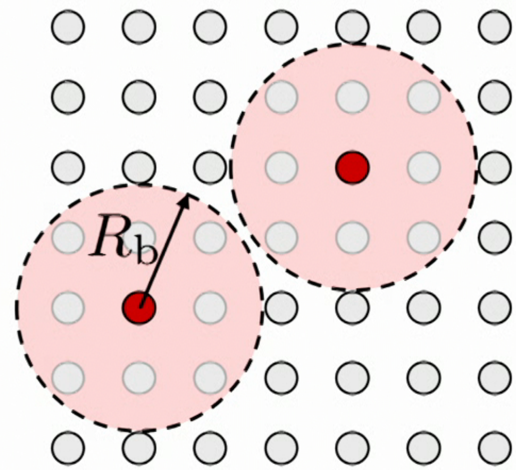


$$H = \Omega \sum_i \sigma_i^x - \Delta \sum_i n_i + \sum_{i<j} V_{ij} n_i n_j$$

$$V(R) = \frac{\Omega}{(R/R_b)^6}$$

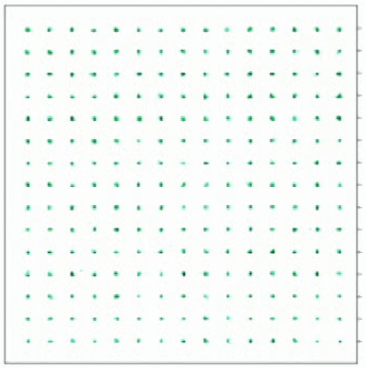
$$\sigma^x = |g\rangle\langle r| + |r\rangle\langle g| \qquad n = |r\rangle\langle r|$$

- Two atoms within the blockade radius cannot both be excited into a Rydberg state simultaneously
- Lattice geometry crucially affects physics

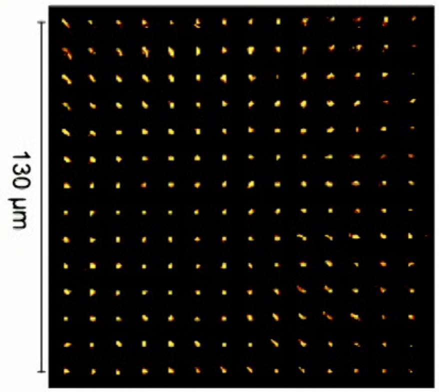


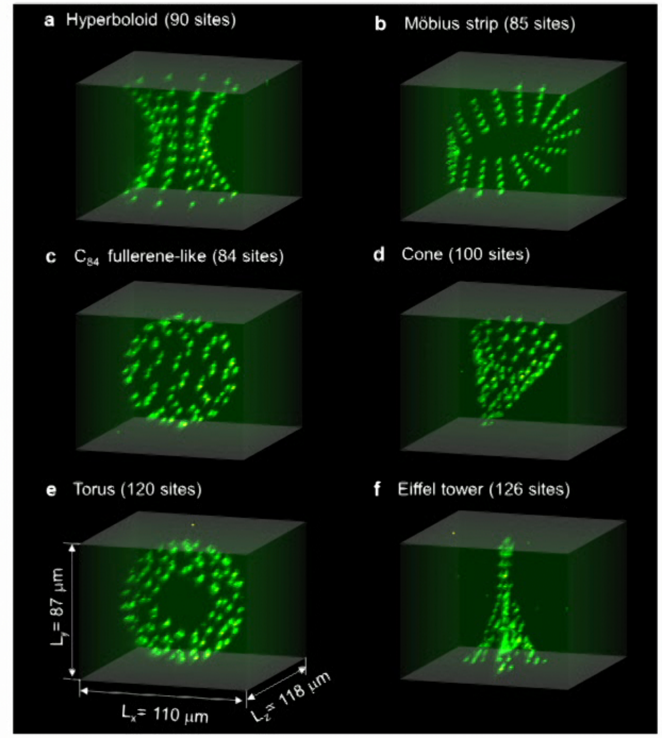
Browaeys, Lahaye, Nature Physics 16, 132 (2020)

Experimental lattices

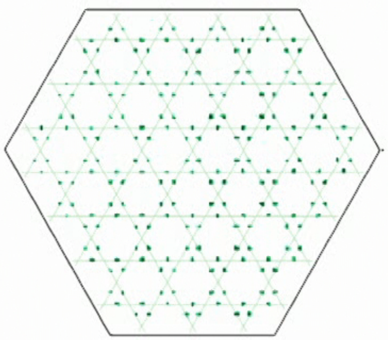


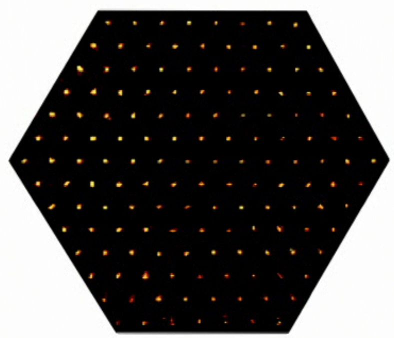
Ebadi et. al. arXiv:2012.12281
Nature 595, 227 (2021)





Barredo, Lienhard, de Léséleuc, Lahaye, Browaeys
Nature 561, (2018)



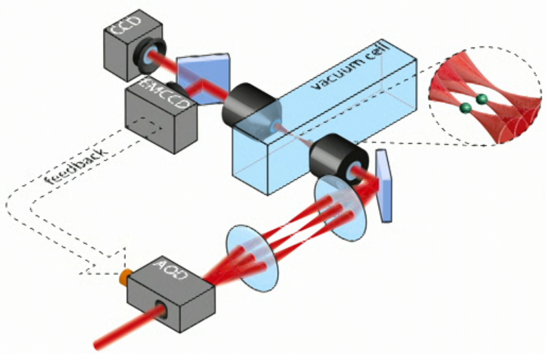


Semeghini et. al. arXiv:2104.04119
Science, 374, 1242 (2021)

Scholl et al. arXiv:2012.12268
Nature 595, 233 (2021)

Data driven state reconstruction

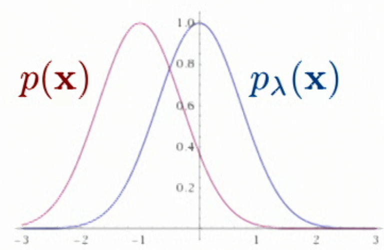
The availability of high quality projective measurement data allows for state reconstruction, e.g. through the KL divergence or maximum likelihood methods



qubit projective measurement data distributed according to Born rule, $p(\mathbf{x})$

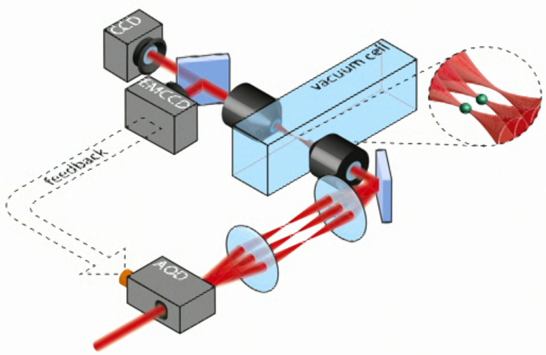
$$\left.\begin{aligned}\mathbf{x}_1 &= (1,0,0,1,1,1,0,0,0,0,\cdots,1)\\ \mathbf{x}_2 &= (1,1,1,0,1,1,0,1,1,1,\cdots,1)\\ \mathbf{x}_3 &= (0,1,1,0,0,1,0,1,0,1,\cdots,0)\\ &\vdots\end{aligned}\right\}\mathcal{D}$$

Goal: use available data to reconstruct the quantum state using a generative model



Data driven state reconstruction

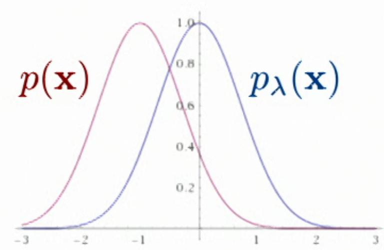
The availability of high quality projective measurement data allows for state reconstruction, e.g. through the KL divergence or maximum likelihood methods

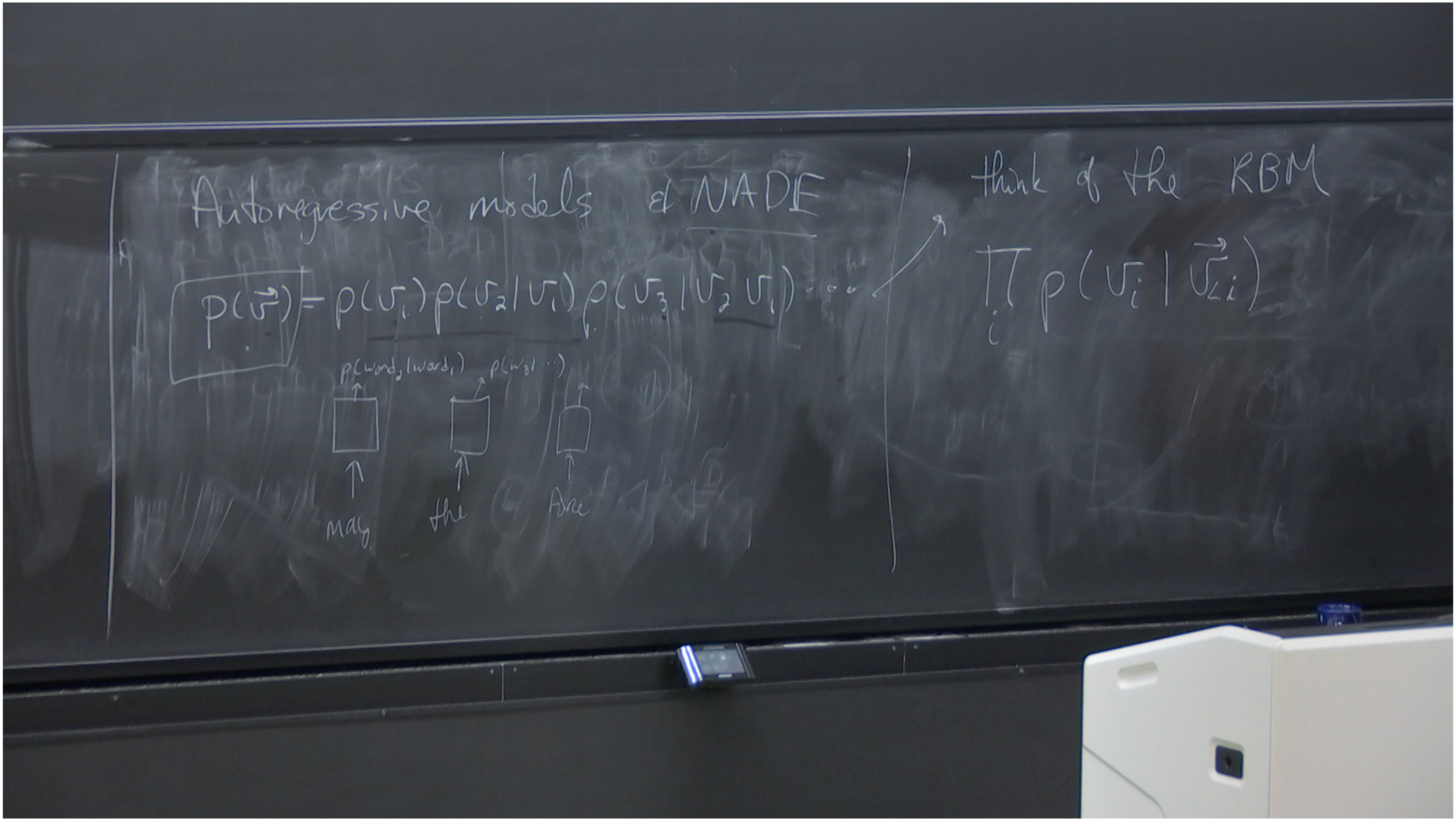


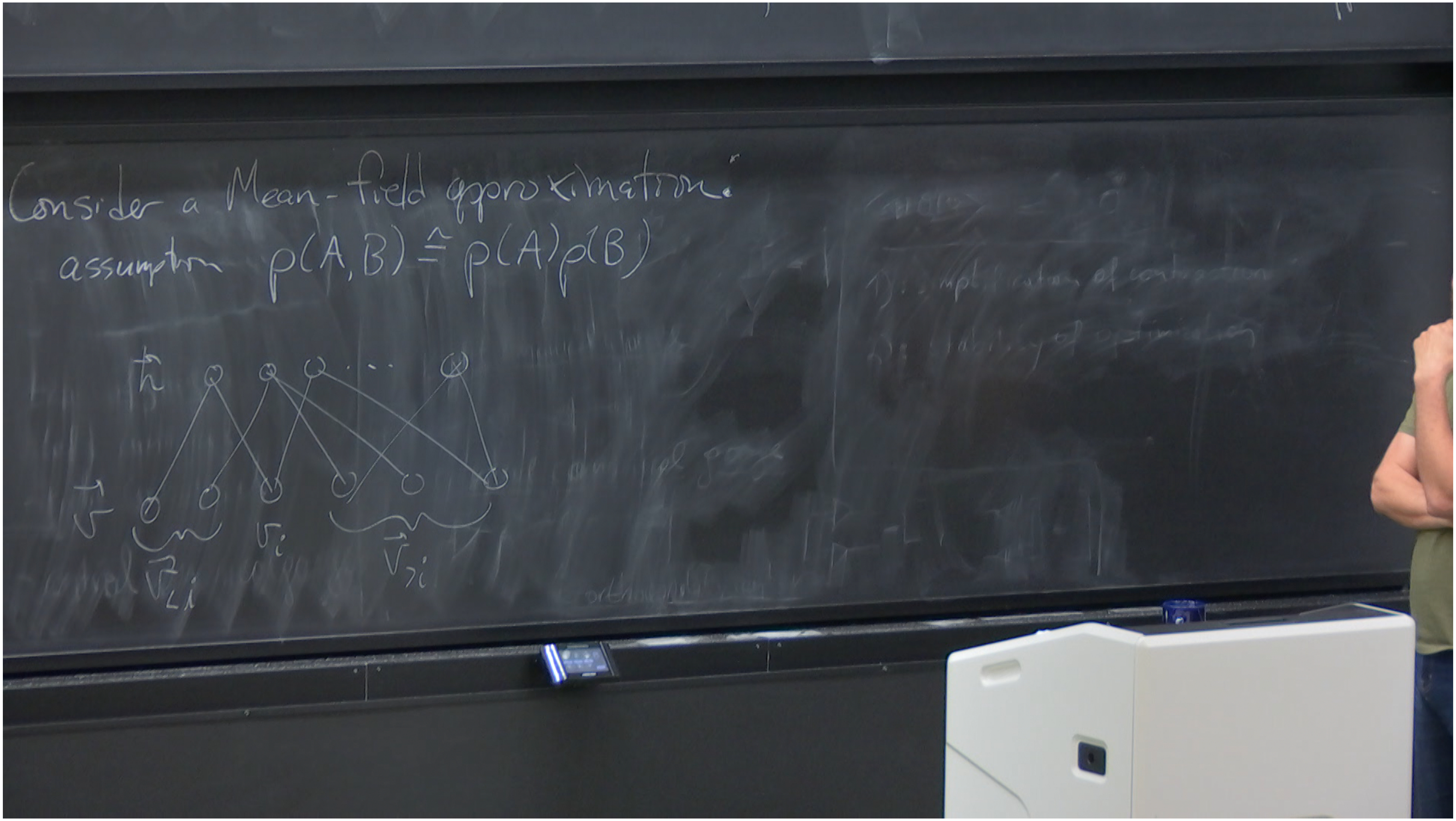
qubit projective measurement data distributed according to Born rule, $p(\mathbf{x})$

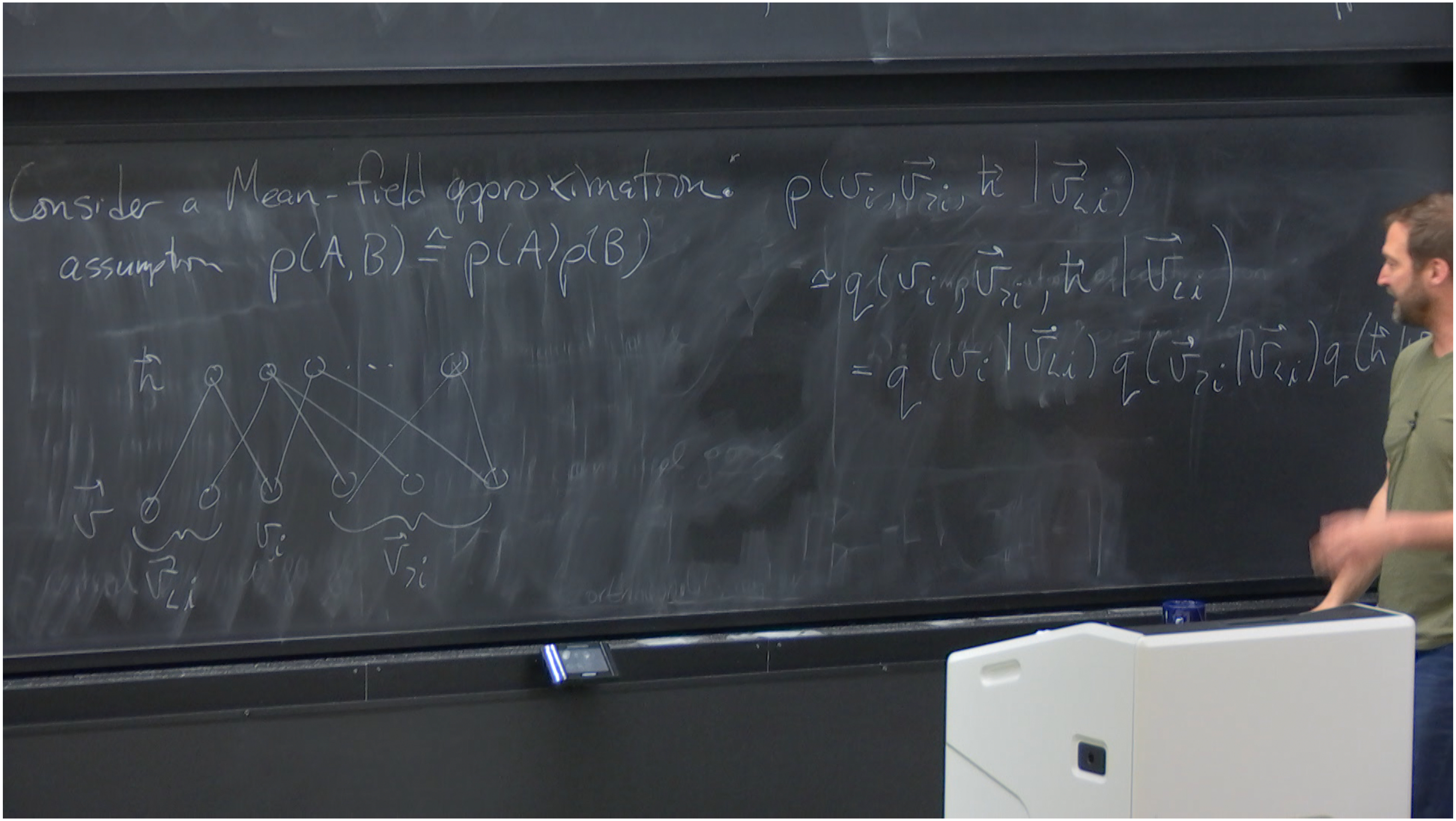
$$\left.\begin{aligned}\mathbf{x}_1 &= (1,0,0,1,1,1,0,0,0,0,\cdots,1)\\ \mathbf{x}_2 &= (1,1,1,0,1,1,0,1,1,1,\cdots,1)\\ \mathbf{x}_3 &= (0,1,1,0,0,1,0,1,0,1,\cdots,0)\\ &\vdots\end{aligned}\right\}\mathcal{D}$$

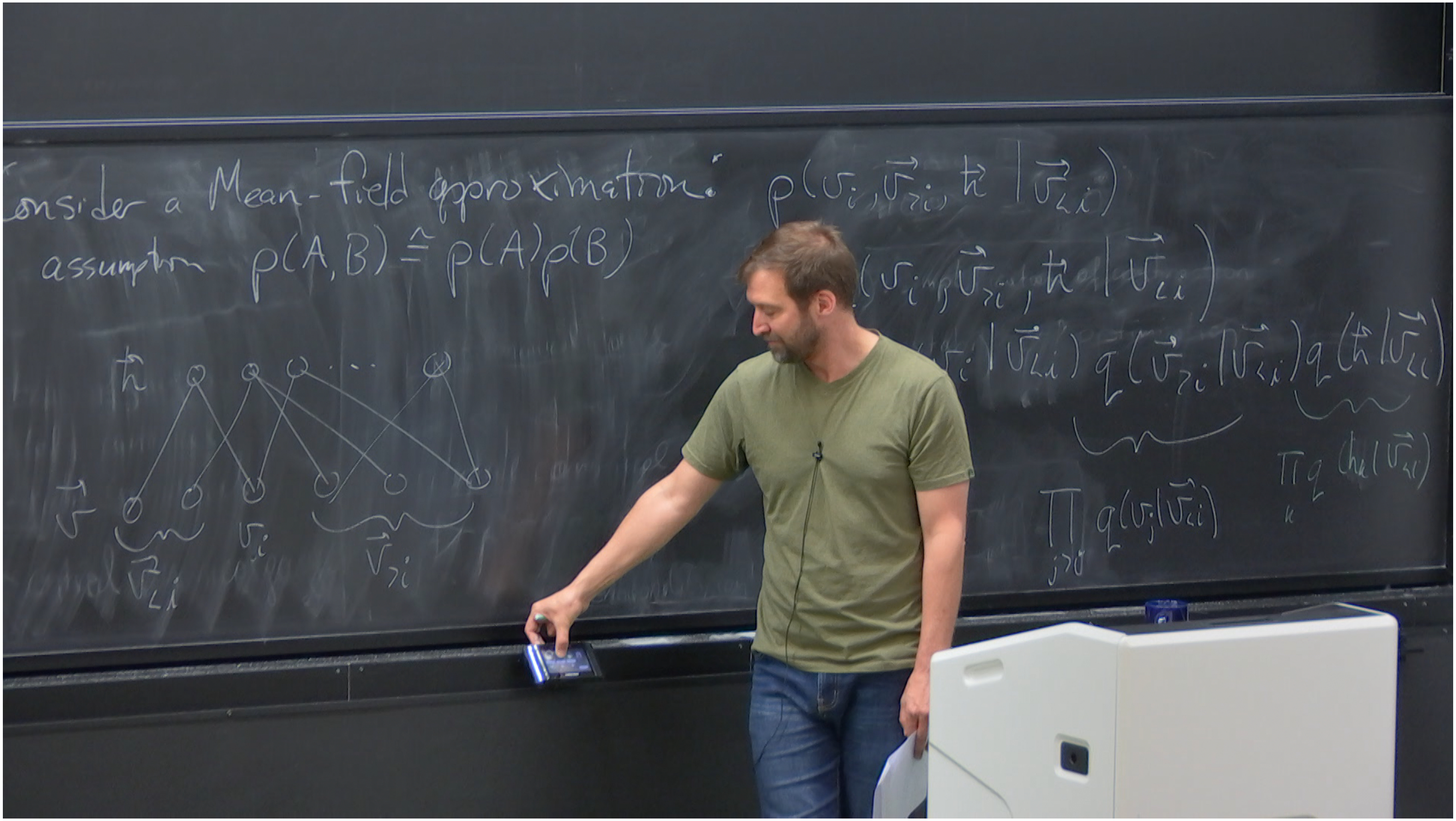
Goal: use available data to reconstruct the quantum state using a generative model

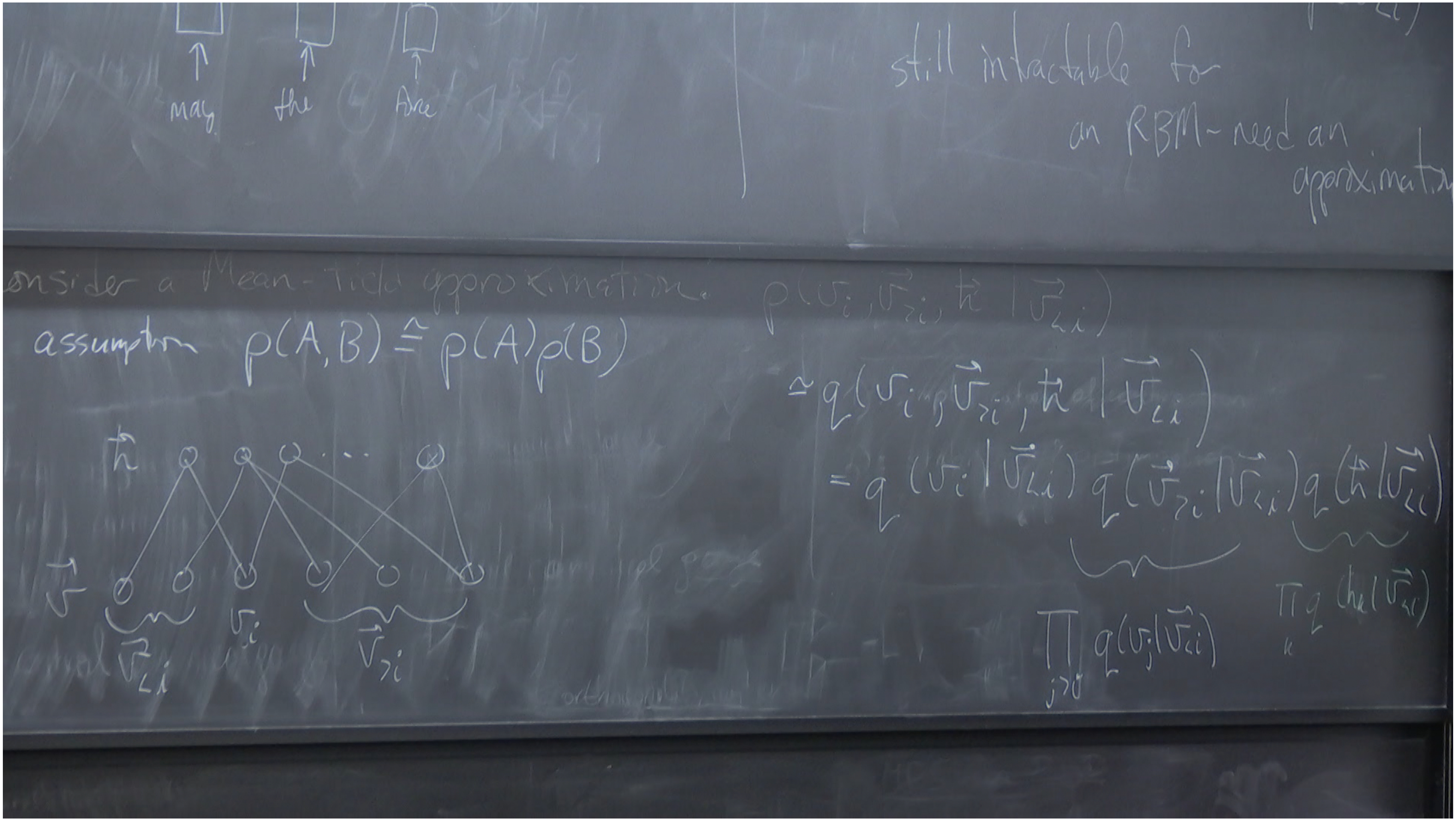



Autoregressive models & NADE
$p(\vec{v}) = p(v_1)\,p(v_2|v_1)\,p(v_3|v_2\,v_1)\cdots$
$p(word_2|word_1)$
$p(w_3|\cdots)$
may
the
fire
think of the RBM
$\prod_i p(v_i|\vec{v}_{<i})$

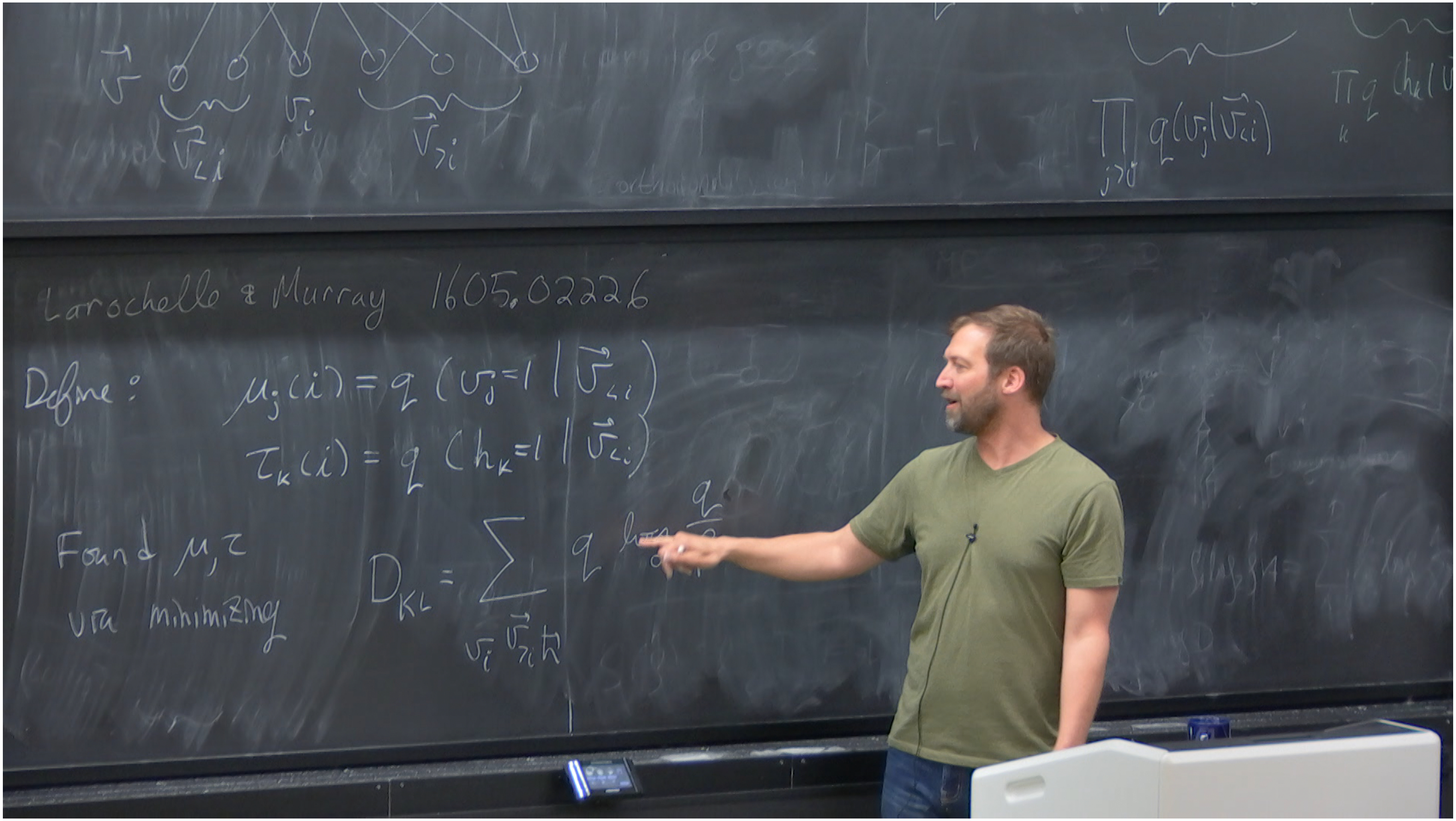

Consider a Mean-field approximation:
assumption ρ(A,B) ≏ ρ(A)ρ(B)
h
v
v_i
V_{<i}
V_{>i}

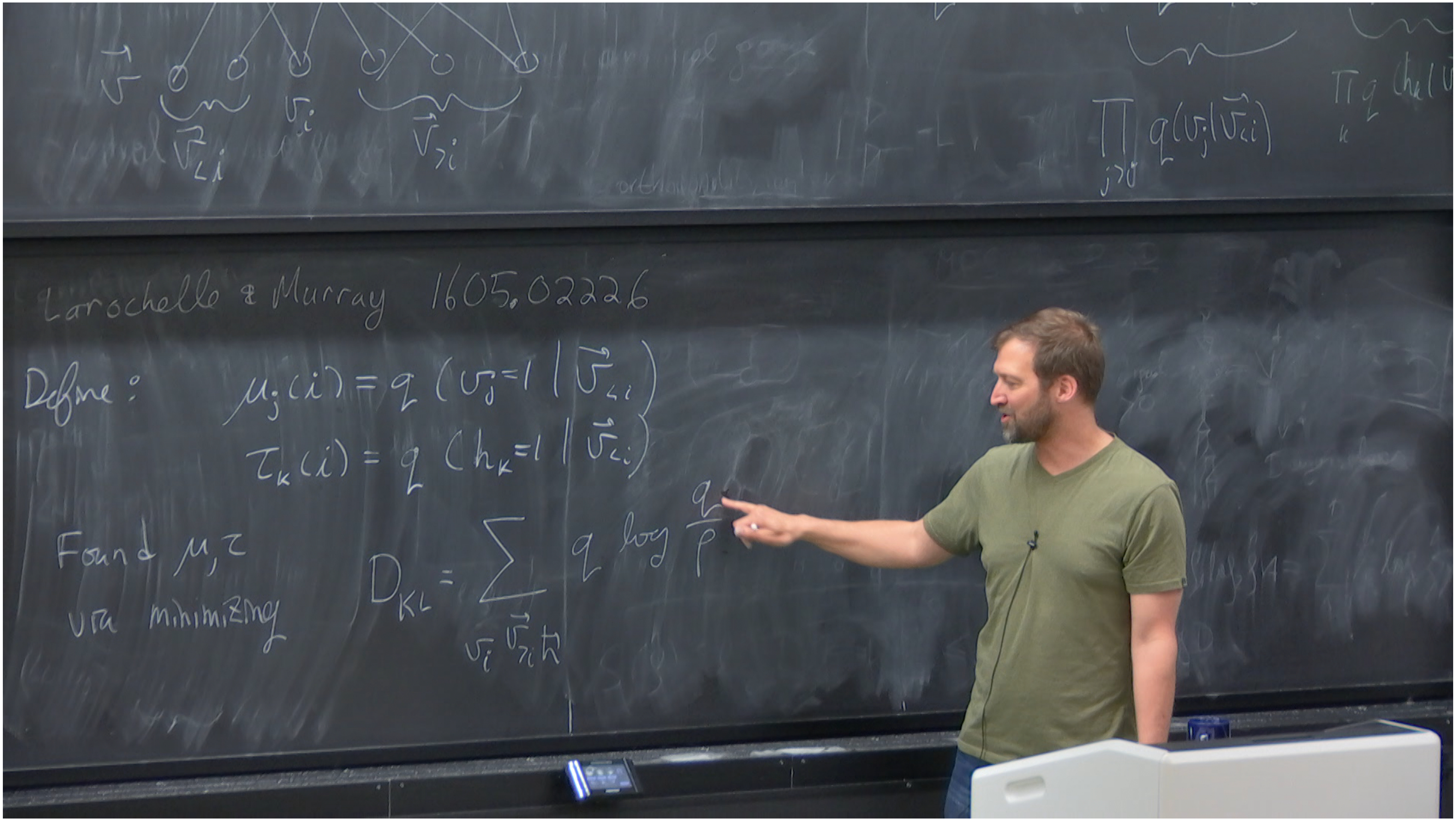

Consider a Mean-field approximation:
assumption $p(A,B) \simeq p(A)p(B)$
$p(v_i, \vec{v}_{>i}, \vec{h} \mid \vec{v}_{<i})$
$\simeq q(v_i, \vec{v}_{>i}, \vec{h} \mid \vec{v}_{<i})$
$= q(v_i \mid \vec{v}_{<i})\, q(\vec{v}_{>i} \mid \vec{v}_{<i})\, q(\vec{h} \mid$
$\vec{h}$
$\vec{v}$
$\vec{v}_{<i}$
v_i
$\vec{v}_{>i}$

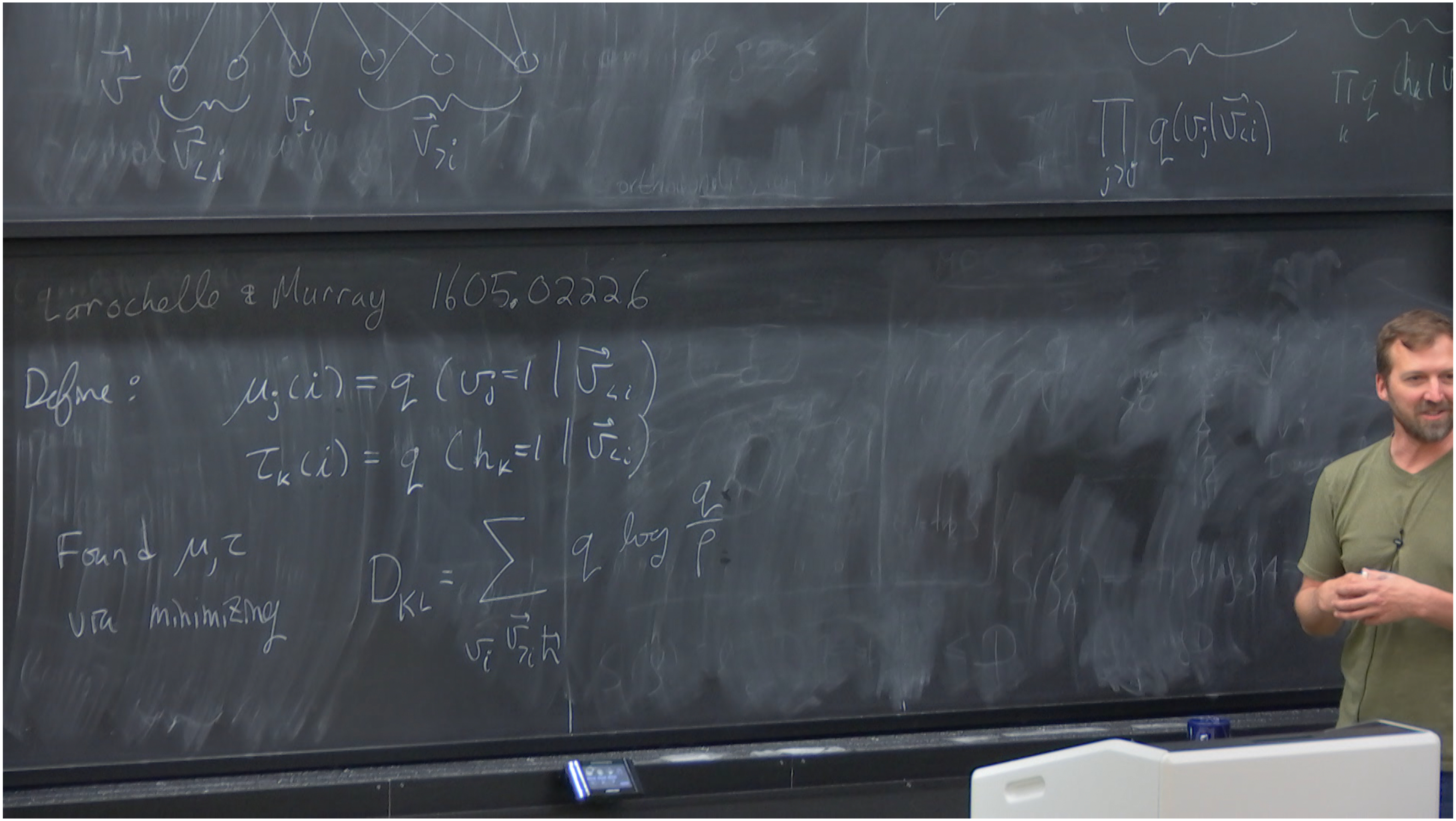

onsider a Mean-field approximation:
assumption p(A,B) ≙ p(A)p(B)
p(v_i, v_{>i}, h | v_{<i})

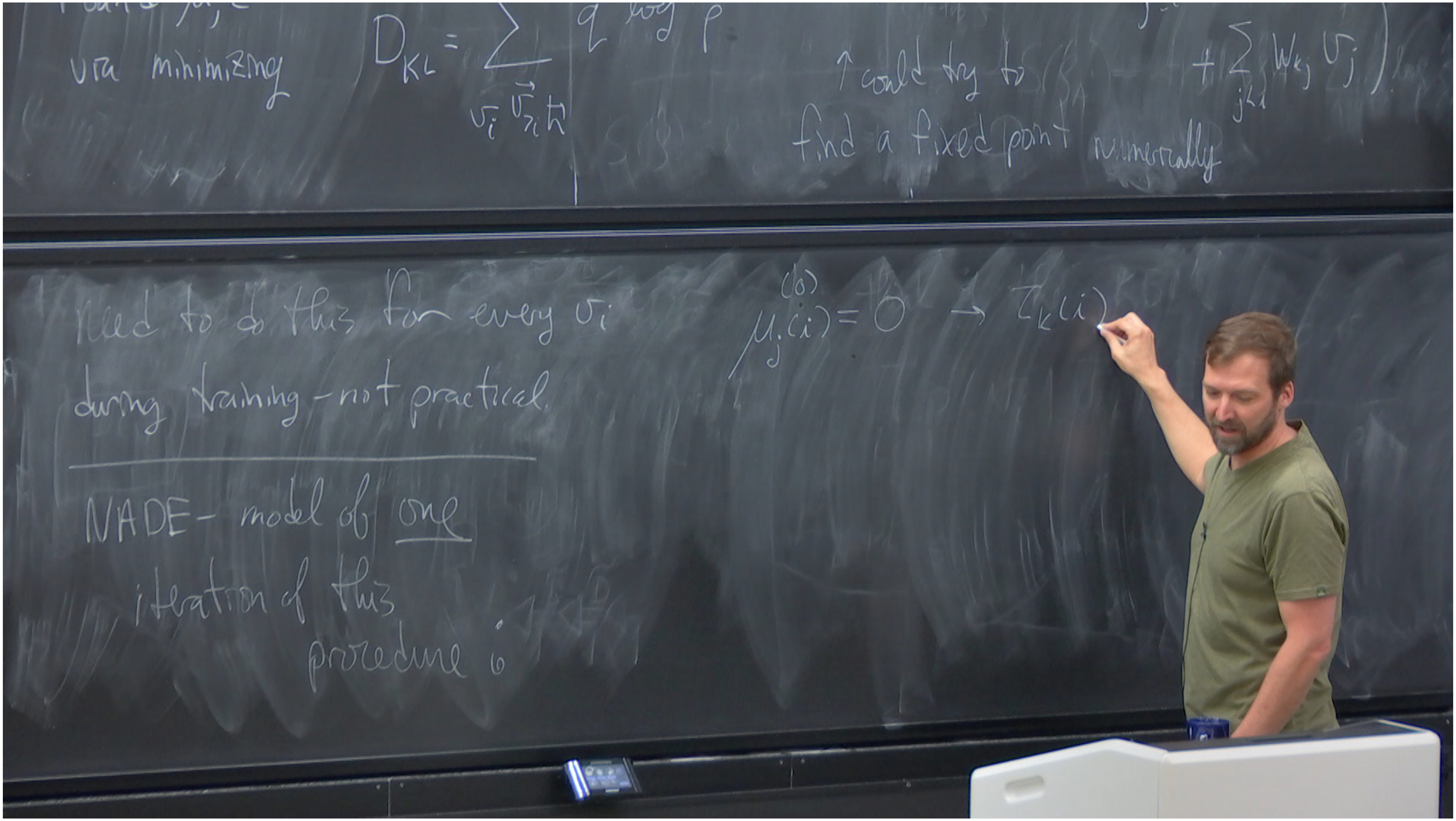


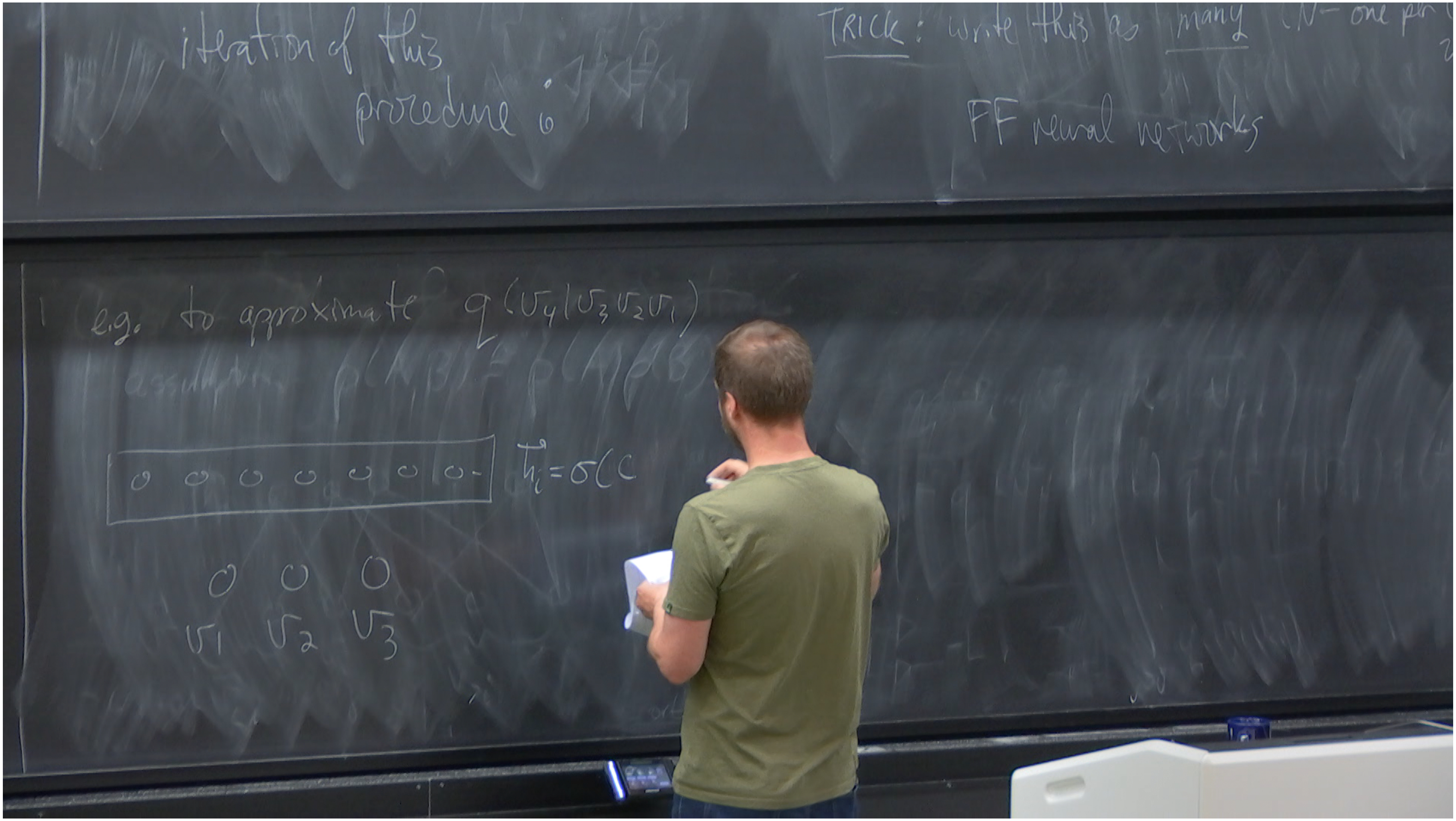
may
the
Are
still intractable for
an RBM - need an
approximation
onsider a Mean-field approximation.
assumption $p(A,B) \simeq p(A)p(B)$
$p(v_i, \vec{v}_{>i}, \vec{h} \mid \vec{v}_{<i})$
$\simeq q(v_i, \vec{v}_{>i}, \vec{h} \mid \vec{v}_{<i})$
$= q(v_i \mid \vec{v}_{<i})\, q(\vec{v}_{>i} \mid \vec{v}_{<i})\, q(\vec{h} \mid \vec{v}_{<i})$
$\prod_{j>i} q(v_j \mid \vec{v}_{<i})$
$\prod_k q(h_k \mid \vec{v}_{<i})$
$\vec{h}$
$\vec{v}$
$\vec{v}_{<i}$
v_i
$\vec{v}_{>i}$

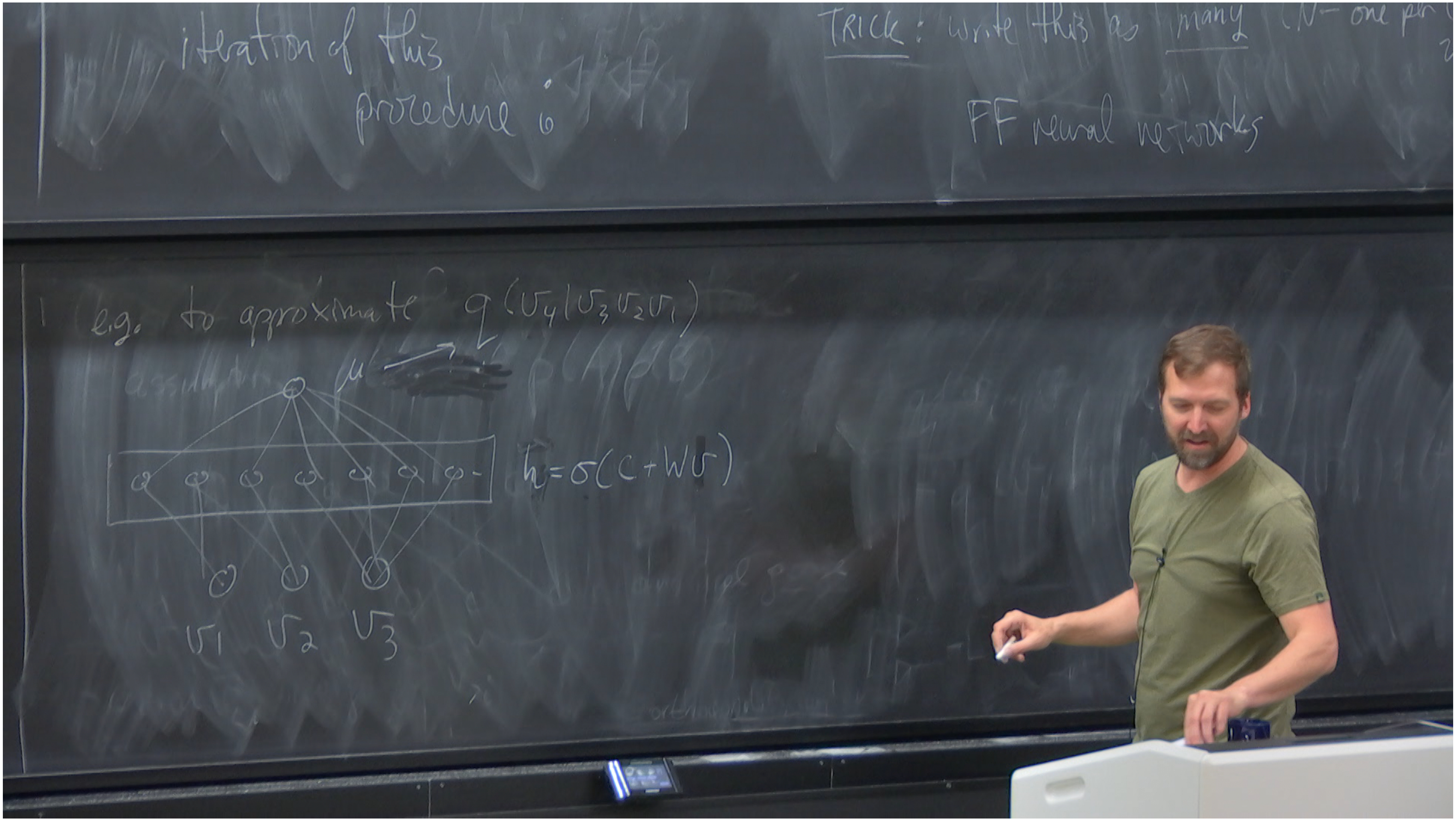

Larochelle & Murray 1605.02226
Define: $\mu_j(i) = q(v_j = 1 | \vec{v}_{<i})$
$\tau_k(i) = q(h_k = 1 | \vec{v}_{<i})$
Found μ, τ via minimizing
$D_{KL} = \sum_{v_i \vec{v}_{>i} \vec{h}} q \log \frac{q}{p}$
$\prod_{j>i} q(v_j | \vec{v}_{<i})$

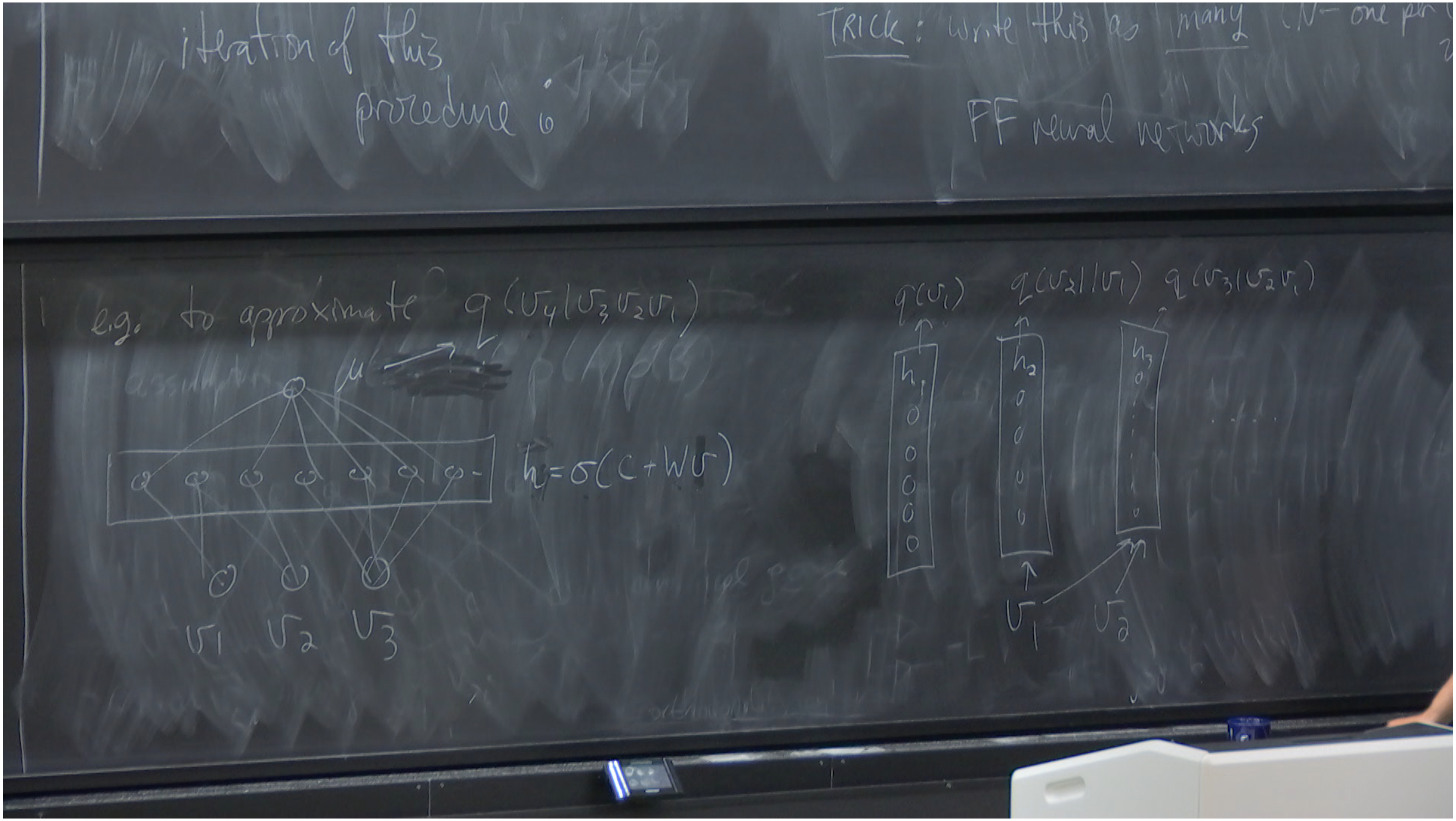

Larochelle & Murray 1605.02226
Define:
μ_j(i) = q(v_j=1 | v_<i)
τ_k(i) = q(h_k=1 | v_<i)
Found μ, τ
via minimizing
D_KL = Σ q log q/p

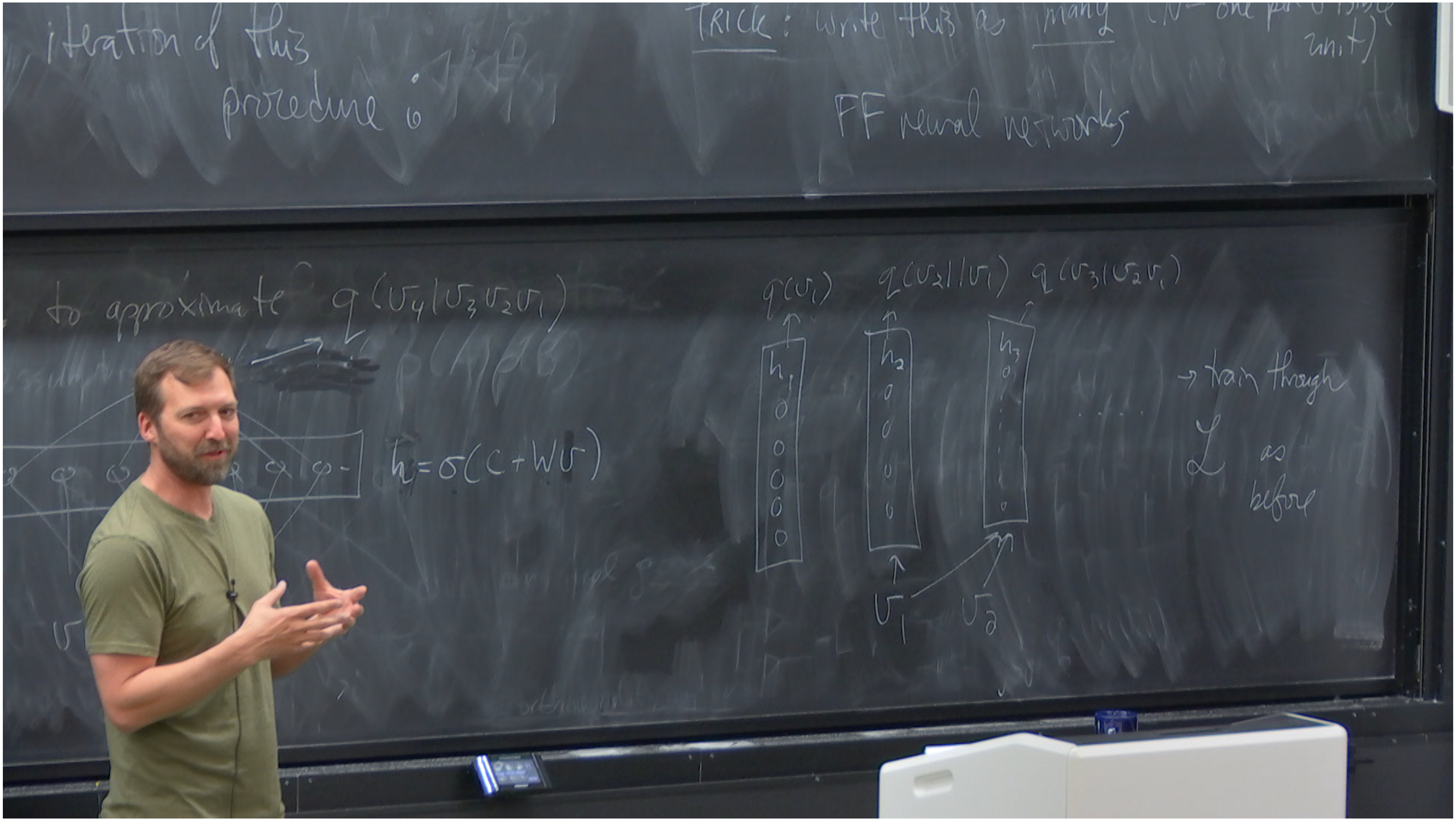

Larochelle & Murray 1605.02226
Define:
Found μ, τ
via minimizing

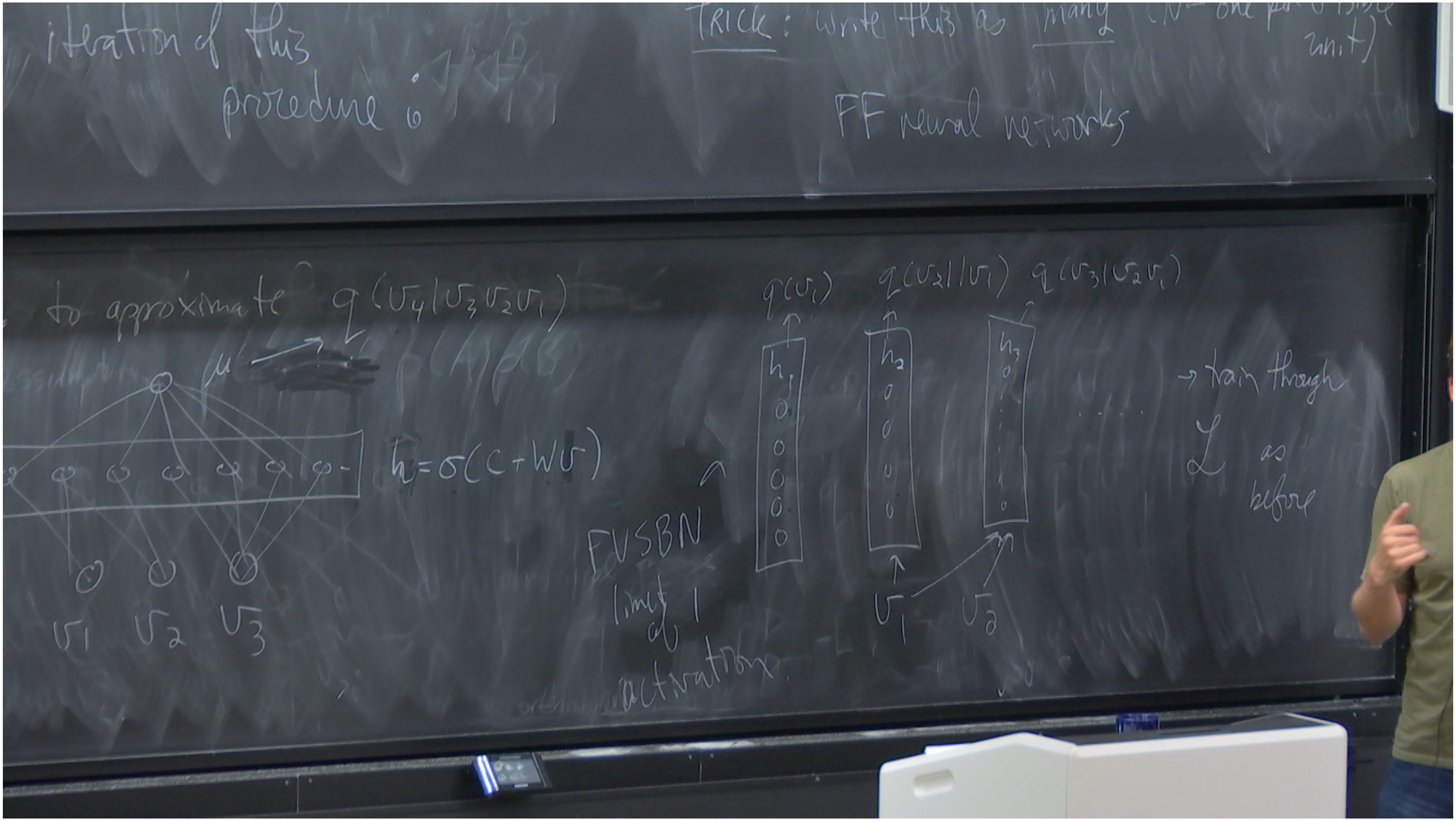

via minimizing
D_KL = ∑ q log p
↑ could try to
find a fixed point numerically
need to do this for every v_i
during training - not practical.
NADE - model of one
iteration of this
procedure ☺


iteration of this procedure :
TRICK : write this as many
FF neural networks
e.g. to approximate q(v4|v3v2v1)
hi = σ(C
v1 v2 v3


iteration of this procedure:
TRICK: write this as many (N – one per
FF neural networks
e.g. to approximate q(v4|v3v2v1)
μ
h = σ(c + Wv)
v1 v2 v3


iteration of this
procedure :
TRICK : write this as many
FF neural networks
e.g. to approximate q(v4|v3 v2 v1)
h = σ(c + Wv)
v1 v2 v3
q(v1) q(v2|v1) q(v3|v2 v1)
h1 h2 h3
v1 v2


iteration of this
procedure :
TRICK : write this as many
unit)
FF neural networks
to approximate q(v4|v3 v2 v1)
h = σ(c + Wv)
q(v1)
q(v2|v1)
q(v3|v2v1)
h1
h2
h3
v1
v2
→ train through
L as before



iteration of this procedure :
TRICK : write this as many
unit)
FF neural networks
to approximate q(v4 | v3 v2 v1)
h = σ(c + Wv)
v1 v2 v3
FVSBN
limit
activation
q(v1)
q(v2 | v1)
q(v3 | v2 v1)
h1
h2
h3
→ train through
L as before